440EP

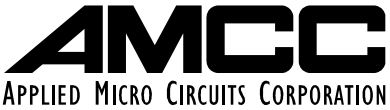
PPC440EP Embedded Processor

Preliminary User's Manual

PPC440EP Embedded Processor

User's Manual

PowerPC®



Applied Micro Circuits Corporation
215 Moffett Park Drive, Sunnyvale, CA 94089
Phone: (858) 450-9333 — (800) 755-2622 — Fax: (858) 450-9885
http://www.amcc.com

Contents

Figures 23

Tables 37

About This Book 43

Part I. Introduction 47

1. Overview 49

1.1 PPC440EP Features 49
1.1.1 PPC440EP Processor Core Features 49
1.1.2 PPC440EP Floating Point Unit Features 49
1.1.3 PPC440EP Peripheral Features 50
1.1.4 PPC440EP CoreConnect Features 53
1.2 PPC440EP Floating Point Unit (FPU) 55
1.3 PPC440EP Internal Buses 55
1.4 PPC440EP Peripheral Functions and Interfaces 55
1.4.1 Reset Interface 55
1.4.2 JTAG Interface 56
1.4.3 Bootstrap Controller 56
1.4.4 Clock and Power Management 56
1.4.5 DDR_SDRAM Memory Controller 56
1.4.6 PCI Bridge Controller 56
1.4.7 External Bus Controller 57
1.4.8 Nand Flash Controller 57
1.4.9 Direct Memory Access Controller 57
1.4.10 Memory Access Layer 57
1.4.11 EMAC to PHY Bridge 58
1.4.12 Ethernet Media Access Controller 58
1.4.13 General Purpose Timers 58
1.4.14 Universal Interrupt Controllers 58
1.4.15 Serial Port Operations 59
1.4.16 Serial Peripheral Interface 59
1.4.17 Inter-Integrated Circuit Bus 59
1.4.18 GPIO Operations 59
1.4.19 Universal Serial Bus Interface 59
1.5 PPC440EP Supported Configurability 60
1.5.1 PPC440EP Addressing Modes 60
1.5.2 PPC440EP Register Set 60
1.5.2.1 General Purpose Registers 61
1.5.2.2 Special Purpose Registers 61
1.5.2.3 Time Base Registers 61
1.5.2.4 Machine State Register 61
1.5.2.5 Condition Register 61
1.5.2.6 Device Control Registers 61
1.5.2.7 Memory-Mapped I/O Registers 62
1.6 PPC440EP Signals 62
1.7 Development Tool Support 62

2. On-Chip Buses 63

2.1 Processor Local Bus 63

2.1.1 PLB Features 63
2.1.2 PLB Master and Slave Assignments 64
2.1.3 PLB Master Priority Assignment 64
2.1.3.1 Alternate PLB4 Master Priority Register 0 (SDR0_AMP0) 66
2.1.3.2 Alternate PLB3 Master Priority Register 1 (SDR0_AMP1) 67
2.1.3.3 PPC440 CPU Control Register (SDR0_CP440) 68
2.1.3.4 PLB4 Master Interrupt Request Register 0 (SDR0_MIRQ0) 69
2.1.3.5 PLB Slave Address Pipeline Register (SDR0_SLPIPE0) 71
2.1.4 PLB4 Arbiter 71
2.1.4.1 PLB4 Arbiters 0 and 1 Registers 73
2.1.4.2 PLB4 Arbiter Revision ID Register (PLB4A0_REVID) 73
2.1.4.3 PLB4 Arbiter Control Register (PLB4An_ACR) 73
2.1.4.4 PLB4 Error Status Register Low (PLB4An_ESRL) 77
2.1.4.5 PLB4 Error Address Register Low (PLB4An_EARL) 78
2.1.4.6 PLB4 Error Address Register High (PLB4An_EARH) 78
2.1.4.7 PLB4 Crossbar Control Register (PLB4A0_CCR) 79
2.1.5 PLB4 to PLB3 Bridge Registers (Bridge Out 0) 80
2.1.5.1 PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL) 80
2.1.5.2 PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH) 80
2.1.5.3 PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0) 81
2.1.5.4 PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1) 83
2.1.5.5 PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG) 84
2.1.5.6 PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR) 85
2.1.5.7 PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR) 86
2.1.5.8 PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID) 86
2.1.6 PLB3 to PLB4 Bridge Registers (Bridge In 0) 86
2.1.6.1 PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL) 87
2.1.6.2 PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH) 87
2.1.6.3 PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0) 87
2.1.6.4 PLB3 to PLB4 Bridge Error Status Register 1 (P3P43BI0_BESR1) 90
2.1.6.5 PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG) 91
2.1.6.6 PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR) 92
2.1.6.7 PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR) 93
2.1.6.8 PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID) 93
2.1.7 PLB4 to OPB Bridge Registers 94
2.1.7.1 PLB4 to OPB Bridge Error Status Register 0 (PLB42OPB0_BESR0) 94
2.1.7.2 PLB4 to OPB Bridge Error Address Register Low (PLB42OPB0_BEARL) 96
2.1.7.3 PLB4 to OPB Bridge Error Address Register High (PLB42OPB0_BEARH) 96
2.1.7.4 PLB4 to OPB Bridge Error Status Register 1(PLB42OPB0_BESR1) 96
2.1.7.5 PLB4 to OPB Bridge Configuration Register (PLB42OPB0_CFG) 97
2.1.7.6 PLB4 to OPB Bridge Burst Latency Timer (PLB42OPB0_LATENCY) 98
2.1.7.7 PLB4 to OPB Bridge Revision ID (PLB42OPB0_REVID) 98
2.1.8 PLB3 Arbiter Registers 98
2.1.8.1 PLB3 Arbiter Revision ID Register (PLB3A0_REVID) 98
2.1.8.2 PLB3 Error Status Register (PLB3A0_BESR) 99
2.1.8.3 PLB3 Error Address Register (PLB3A0_BEAR) 100
2.1.8.4 PLB3 Arbiter 0 Control Register (PLB3A0_ACR) 101
2.1.9 PLB3 to OPB Bridge Registers 102
2.1.9.1 PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0) 102
2.1.9.2 PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR) 104
2.1.9.3 PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID) 104
2.1.9.4 PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1) 104
2.2 On-Chip Peripheral Bus 105

2.2.1 OPB Features 105
2.2.2 OPB Master Assignments 105
2.2.3 OPB Arbiter Registers 106
2.2.3.1 OPB Arbiter Priority Register (OPBAx_PR) 106
2.2.3.2 OPB Arbiter Control Register (OPBAx_CR) 107
2.2.4 OPB to PLB3 Bridge Registers 107
2.2.4.1 OPB to PLB3 Bridge Control Register (OPB2PLB30_BCTRL) 108
2.2.4.2 OPB to PLB3 Bridge Status Register (OPB2PLB30_BSTAT) 108
2.2.4.3 OPB to PLB3 Bridge Revision ID Register (OPB2PLB30_REVID) 108
2.3 Device Control Register (DCR) Bus 109

3. PLB Performance Monitor 111
3.1 PPM Features 111
3.2 PPM Functional Description 112
3.2.1 Master Event Counter Logic 113
3.2.2 Slave Event Counter Logic 113
3.2.3 Generic Pipeline Event Counter Logic 114
3.2.4 Duration Counter Logic 115
3.3 Monitoring of PLB Events 116
3.3.1 PLB Event-Occurrence Qualification 117
3.3.2 PLB Event-Duration Measurements 118
3.4 PLB Performance Monitor Registers 119
3.4.1 PPM Configuration Address Register (PPM0_CFGADDR) 121
3.4.2 PPM Configuration Data Register (PPM0_CFGDATA) 121
3.4.3 Interrupt Status Register (PPM0_ISR) 121
3.4.4 Control Register (PPM0_CR) 123
3.4.5 Cycle Counter Register (PPM0_CCR) 124
3.4.6 Upper Address Register (PPM0_UAR) 125
3.4.7 Lower Address Register (PPM0_LAR) 125
3.4.8 Upper Address Mask Register (PPM0_UAMR) 125
3.4.9 Lower Address Mask Register (PPM0_LAMR) 126
3.4.10 Revision ID Register (PPM0_RIDR) 126
3.4.11 Master Event Counter Selection Register 0:3 (PPM0_MCSR0:PPM0_MCSR3) 126
3.4.12 Slave Event Counter Selection Register 0:7 (PPM0_SCSR0:PPM0_SCSR3) 128
3.4.13 Generic Event Counter Selection Register 0:3 (PPM0_GCSR0:PPM0_GCSR3) 130
3.4.14 Master Event Counter Register 0:3 (PPM0_MCR0:PPM0_MCR3) 130
3.4.15 Slave Event Counter Register 0:3 (PPM0_SCR0:PPM0_SCR3) 130
3.4.16 Generic Pipeline Event Counter Register 0:3 (PPM0_GCR0:PPM0_GCR3) 131
3.4.17 Duration Counter Selection Register 0:1 (PPM0_DCSR0:PPM0_DCSR1) 131
3.4.18 Duration Counter Maximum Register 0:1 (PPM0_DCMXR0:PPM0_DCMXR1) 132
3.4.19 Duration Counter Minimum Register 0:1 (PPM0_DCMNR0:PPM0_DCMNR1) 132
3.4.20 Duration Counter Total Value Register 0:1 (PPM0_DCTVR0:PPM0_DCTVR1) 133
3.4.21 Duration Counter Occurrence Total Register 0:1 (PPM0_DCOTR0:PPM0_DCOTR1) 133

Part II. PPC440EP RISC Processor 135

4. Programming Model 137
4.1 Storage Addressing 137
4.1.1 Device Control Registers 139
4.1.2 Memory Mapped Registers 149
4.2 Instruction Classes 158

5. FPU Programming Model 159

5.1 Storage Addressing 159
5.1.1 Storage Operands 159
5.1.2 Effective Address Calculation 160
5.1.3 Data Storage Addressing Modes 160
5.2 Floating-Point Exceptions 161
5.3 Floating-Point Registers 161
5.3.1 Register Types 162
5.3.1.1 Floating-Point Registers (FPR0:FPR31) 162
5.3.1.2 Floating-Point Status and Control Register (FPSCR) 162
5.4 Floating-Point Data Formats 165
5.4.1 Value Representation 166
5.4.2 Binary Floating-Point Numbers 167
5.4.2.1 Normalized Numbers 167
5.4.2.2 Denormalized Numbers 167
5.4.2.3 Zero Values 167
5.4.3 Infinities 167
5.4.3.1 Not a Numbers 168
5.4.4 Sign of Result 168
5.4.5 Normalization and Denormalization 169
5.4.6 Data Handling and Precision 169
5.4.7 Rounding 170
5.5 Floating-Point Execution Models 171
5.5.1 Execution Model for IEEE Operations 172
5.5.2 Execution Model for Multiply-Add Type Instructions 174
5.6 Floating-Point Instructions 175
5.6.1 Instructions By Category 176
5.6.2 Load and Store Instructions 176
5.6.3 Floating-Point Store Instructions 178
5.6.4 Floating-Point Move Instructions 179
5.6.5 Floating-Point Arithmetic Instructions 180
5.6.5.1 Floating-Point Multiply-Add Instructions 180
5.6.6 Floating-Point Rounding and Conversion Instructions 181
5.6.7 Floating-Point Compare Instructions 181
5.6.8 Floating-Point Status and Control Register Instructions 181

6. Instruction and Data Caches 183

7. Memory Management 185

Part III. PPC440EP System Operations 187

8. Reset and Initialization 189
8.1 Reset Signals 189
8.2 Reset Types 189
8.2.1 CPU Reset 189
8.2.2 Chip Reset 189
8.2.3 System Reset 190
8.3 Processor Core State After Reset 191
8.4 PPC440EP Chip Initialization 195
8.4.1 Initial Configuration Register (CPR0_ICFG) 196
8.4.2 System Device Control Registers 196
8.4.2.1 Electronic Chip ID Register 0 (SDR0_ECID0) 197
8.4.2.2 Electronic Chip ID Register 1 (SDR0_ECID1) 197

8.4.2.3 Electronic Chip ID Register 2 (SDR0_ECID2) 197
8.4.2.4 Electronic Chip ID Register 3 (SDR0_ECID3) 197
8.4.2.5 Pin Function Control Register 0 (SDR0_PFC0) 197
8.4.2.6 Pin Function Control Register 1 (SDR0_PFC1) 198
8.4.2.7 Slave Address Pipeline Register (SDR0_SLPIPE0) 199
8.4.2.8 Soft Reset Registers (SDR0_SRST0 and SDR0_SRST1) 199
8.4.2.9 Miscellaneous Function Register (SDR0_MFR) 200
8.4.2.10 Hardware Self Refresh Register (SDR0_HSF) 201
8.5 Initialization Software Requirements 201

9. Bootstrap Controller 207
9.1 Bootstrap Configuration 207
9.2 Bootstrap Options 208
9.3 IIC Bootstrap Operations 213
9.3.1 Performance 214
9.4 IIC Bootstrap Registers 216
9.4.1 Pin Strapping Register (SDR0_PINSTP) 217
9.4.2 Serial Device Controller Settings Register (SDR0_SDCS0) 217
9.4.3 Serial Device Strap Register 0 (SDR0_SDSTP0) 218
9.4.4 Serial Device Strap Register 1 (SDR0_SDSTP1) 219
9.4.5 Read-Only Equivalent of SDR0_CUST0 (SDR0_SDSTP2) 220
9.4.6 Read-Only Version of SDR0_CUST1 (SDR0_SDSTP3) 221
9.4.7 Customer Configuration Register 0 (SDR0_CUST0) 221
9.4.8 Custom Configuration Register 1 (SDR0_CUST1) 222
9.4.9 EBC Configuration Register (SDR0_EBC0) 222

10. Universal Interrupt Controller 223
10.1 UIC Overview 223
10.2 UIC Features 223
10.3 UIC Interrupt Assignments 224
10.4 Interrupt Programmability 226
10.5 UIC Registers 226
10.5.1 UIC0 Status Register (UIC0_SR) 226
10.5.2 UIC1 Status Register (UIC1_SR) 228
10.5.3 UIC0 Enable Register (UIC0_ER) 230
10.5.4 UIC1 Enable Register (UIC1_ER) 233
10.5.5 UIC0 Critical Register (UIC0_CR) 235
10.5.6 UIC1 Critical Register (UIC1_CR) 237
10.5.7 UIC0 Polarity Register (UIC0_PR) 239
10.5.8 UIC1 Polarity Register (UIC1_PR) 241
10.5.9 UIC0 Trigger Register (UIC0_TR) 244
10.5.10 UIC1 Trigger Register (UIC1_TR) 246
10.5.11 UIC0 Masked Status Register (UIC0_MSR) 248
10.5.12 UIC1 Masked Status Register (UIC1_MSR) 250
10.5.13 UIC0 Vector Configuration Register (UIC0_VCR) 253
10.5.14 UIC1 Vector Configuration Register (UIC1_VCR) 253
10.5.15 UIC0 Vector Register (UIC0_VR) 254
10.5.15.1 Using the Value in UIC0_VR as a Vector Address or Entry Table Lookup 255
10.5.15.2 Vector Generation Scenarios 255
10.5.16 UIC1 Vector Register (UIC1_VR) 255
10.5.16.1 Using the Value in UIC1_VR as a Vector Address or Entry Table Lookup 256
10.5.16.2 Vector Generation Scenarios 256

11. Interrupts and Exceptions 257

12. Floating Point Unit Interrupts and Exceptions 259
12.1 Floating-Point Exceptions 259
12.2 Exceptions List 260
12.3 Floating-Point Interrupts 262
12.3.1 Floating-Point Unavailable Interrupt 262
12.4 Floating-Point Exception Behavior 263
12.4.1 Invalid Operation Exception 263
12.4.1.1 Action 264
12.4.2 Zero Divide Exception 265
12.4.2.1 Action 265
12.4.3 Overflow Exception 265
12.4.3.1 Action 265
12.4.4 Underflow Exception 266
12.4.4.1 Action 266
12.4.5 Inexact Exception 267
12.4.5.1 Action 267
12.5 Exception Priorities for Floating-Point Load and Store Instructions 268
12.6 Exception Priorities for other Floating-Point Instructions 268
12.7 QNaN 269
12.8 Updating FPRs on Exceptions 269
12.9 Floating-Point Status and Control Register 269
12.10 Updating the CR 270
12.10.1 CR Fields 270
12.10.2 Updating CR Fields 270
12.10.3 Generation of QNaN Results 271

13. Timer Facilities 273

14. General Purpose Timers 275
14.1 GPT Features 275
14.2 Time Base Counter 275
14.3 Compare Timers 275
14.3.1 Compare Timer Interrupt 276
14.4 GPT Registers 276
14.4.1 GPT Time Base Counter Register (GPT0_TBC) 277
14.4.2 GPT Interrupt Mask Register (GPT0_IM) 278
14.4.3 GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC) 278
14.4.4 GPT Interrupt Enable Register (GPT0_IE) 279
14.4.5 GPT Compare Timer Registers (GPT0_COMP0:GPT0_COMP6) 280
14.4.6 GPT Compare Mask Registers (GPT0_MASK0:GPT0_MASK6) 280
14.4.7 GPT Down Count Timer (GPT0_DCT0) 280
14.4.8 GPT Down Count Timer Interrupt Status (GPT0_DCIS) 281

15. Clocking 283
15.1 System Clocking 284
15.1.1 Feedback Selection 285
15.1.2 VCO Frequency and 'M' Value for SYS PLL 285
15.1.3 SYS PLL TUNE Setting 287
15.1.4 IIC Bootstrap Controller Clocking 287
15.1.5 Clocks For Off Chip Use 287
15.1.6 SYS PLL Strapping 288

15.1.7 PLL Bypass (Emulation Mode) ... 288
15.1.8 CPU / PLB Frequency N:1 Setting ... 288
15.1.9 Choosing System Clock Ratios ... 289
15.1.10 System Clock Ratio Examples ... 289
15.1.10.1 PLL Local Feedback Example ... 289
15.1.10.2 CPU Feedback Example ... 290
15.1.10.3 PerClk Feedback Example ... 291
15.2 PCI Clocking ... 292
15.2.1 PCI Adapter Applications ... 293
15.3 Clocking Registers ... 294
15.3.1 Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR) ... 294
15.3.2 Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA) ... 294
15.3.3 Clocking Update Register (CPR0_CLKUPD) ... 295
15.3.4 PLL Control Register (CPR0_PLLC0) ... 295
15.3.5 PLL Divisor Register (CPR0_PLLD0) ... 296
15.3.6 Primary A Divisor Register (CPR0_PRIMAD0) ... 297
15.3.7 Primary B Divisor Register (CPR0_PRIMBD0) ... 298
15.3.8 OPB Clock Divisor Register (CPR0_OPBD0) ... 298
15.3.9 Peripheral Clock Divisor Register (CPR0_PERD0) ... 298
15.3.10 MAL Clock Divisor Register (CPR0_MALD) ... 299
15.3.11 Sync PCI Clock Divisor Register (CPR0_SPCID) ... 299

16. Clock and Power Management ... 301
16.1 Overview ... 301
16.2 CPM Registers ... 301
16.2.1 CPM Enable Register (CPM0_ER) ... 301
16.2.2 CPM Force Register (CPM0_FR) ... 303
16.2.3 CPM Status Register (CPM0_SR) ... 304

17. Debug Facilities ... 307

Part IV. PPC440EP Peripheral Functions and Interfaces ... 309

18. DDR SDRAM Controller ... 311
18.1 Interface Signals ... 311
18.2 DDR SDRAM Registers ... 313
18.2.1 Device Configuration ... 314
18.2.1.1 Initial Configuration Following Power-On-Reset ... 316
18.2.1.2 Re-Configuration Following Initial Configuration ... 316
18.2.2 PPC440EP PLB3 and PLB4 Masters ... 316
18.2.3 Bus Error Syndrome Register 0 (SDRAM0_BESR0) ... 317
18.2.4 Bus Error Syndrome Register 1 (SDRAM0_BESR1) ... 318
18.2.5 Master Bus Error Address Register (SDRAM0_BEAR) ... 319
18.2.6 Master Write Interrupt (SDRAM0_MIRQ) ... 319
18.2.7 PLB Slave Interface Options (SDRAM0_SLIO) ... 320
18.2.8 Memory Controller Options 0 (SDRAM0_CFG0) ... 321
18.2.9 Memory Controller Options 1 (SDRAM0_CFG1) ... 322
18.2.10 DDR SDRAM Device Options (SDRAM0_DEVOPT) ... 323
18.2.11 Memory Controller Status (SDRAM0_MCSTS) ... 323
18.2.12 Refresh Timer Register (SDRAM0_RTR) ... 324
18.2.13 Power Management Idle Timer (SDRAM0_PMIT) ... 324
18.2.14 PLB UABus Base Address (SDRAM0_UABBA) ... 324
18.2.15 Memory 0–3 Configuration (SDRAM0_B0CR:SDRAM0_B3CR) ... 325
18.2.16 SDRAM Timing Register 0 (SDRAM0_TR0) ... 326

18.2.17 SDRAM Timing Register 1 (SDRAM0_TR1) 328
18.2.17.1 SDRAM0_TR1 [RDSL]/[RDCD]/[RDCT] 329
18.2.17.2 SDRAM0_TR1 [RDSS] 330
18.2.17.3 CAS Latency 2 Devices 330
18.2.17.4 CAS Latency 2.5 Devices 331
18.2.17.5 CAS Latency 3 Devices 332
18.2.17.6 SDRAM0_TR1 Recommendations/Considerations 333
18.2.18 DDR SDRAM Clock Timing Register (SDRAM0_CLKTR) 334
18.2.18.1 CLKTR Phase Advance and Delay 334
18.2.18.2 SDRAM0_CLKTR Phase Advance and Delay Considerations 336
18.2.18.3 DDR Delay Line Register (SDR0_DDRDL0) 336
18.2.19 Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR) 337
18.2.19.1 WDDCTR Phase Advance and Delay 337
18.2.19.2 WDDCTR/CLKTR Relationships 337
18.2.20 Delay Line Calibration Register (SDRAM0_DLYCAL) 339
18.2.21 ECC Error Status Register (SDRAM0_ECCESR) 340
18.2.22 Controller ID Register (SDRAM0_CID) 341
18.2.23 Revision ID Register (SDRAM0_RID) 341
18.3 Initialization 342
18.4 DDR to Memory Address Decode 342
18.5 DDR Slave Interface Options 343
18.5.1 Read Rearbitration 343
18.5.2 Write Rearbitration 343
18.5.3 Read Around Write 343
18.6 Basic DDR SDRAM Memory Requirements 344
18.7 Page Management 344
18.7.1 PMU Enabled 344
18.7.2 PMU Disabled 345
18.7.3 Physical Address to Memory Address Mapping 346
18.8 SDRAM Commands and Operations 347
18.8.1 DDR SDRAM Timing Parameters 347
18.8.2 Precharge Command 349
18.8.3 Refresh 349
18.8.4 Self-Refresh Operation 350
18.8.4.1 Software-Initiated Self-Refresh Operation 350
18.8.4.2 Hardware-Initiated Refresh Operation 352
18.8.5 Mode Register Write Commands 354
18.9 Registered DIMM Support 355
18.10 Error Checking and Correction (ECC) 355
18.10.1 Read-Modify-Writes 355
18.10.2 ECC Timing 356
18.10.3 ECC Configuration Register 356
18.10.4 ECC Error Register 356
18.10.5 Error Detection and Error Handling 356
18.10.6 Memory Read and Write Errors 356
18.10.7 Uncorrectable ECC Error on Memory Read 356
18.10.8 Uncorrectable ECC Error on Memory Partial Write 357
18.10.9 Correctable ECC Error on Memory Read 357
18.10.10 Correctable ECC Error on PLB Partial SDRAM Memory Write 357
18.10.11 ECC Diagnostics 357
18.10.12 ECC Code Matrix 358
18.11 Power Management 360

18.11.1 Sleep Mode Entry ... 360
18.11.2 Sleep Mode ... 360
18.11.3 Sleep Mode Exit ... 360
18.12 System Memory Clock Concerns ... 361

19. Peripheral Component Interconnect (PCI) Interface ... 363
19.1 Features ... 363
19.2 Byte Ordering ... 364
19.3 PCI Bridge Functional Blocks ... 365
19.3.1 PLB-to-PCI Half-Bridge ... 365
19.3.2 PCI-to-PLB Half-Bridge ... 366
19.3.3 PCI Arbiter ... 366
19.4 PCI Bridge Address Mapping ... 367
19.4.1 PLB-to-PCI Address Mapping ... 367
19.4.2 PCI-to-PLB Address Mapping ... 369
19.4.3 PCI Target Map Configuration ... 370
19.5 PCI Bridge Transaction Handling ... 370
19.5.1 PLB-to-PCI Transaction Handling ... 371
19.5.1.1 PCI Master Commands ... 371
19.5.1.2 PLB Slave Read Handling ... 372
19.5.1.3 Prefetching ... 373
19.5.1.4 PLB Slave Write Handling ... 373
19.5.1.4 PLB Slave Write Post Buffer ... 373
19.5.1.5 Aborted PLB Requests ... 373
19.5.1.6 Retried PCI Reads ... 374
19.5.2 PCI-to-PLB Transaction Handling ... 374
19.5.2.1 PLB Master Commands ... 375
19.5.2.2 Handling of Reads from PCI Masters ... 375
19.5.2.3 Miscellaneous ... 377
19.5.3 Completion Ordering ... 377
19.5.3.1 PCI Producer-Consumer Model ... 378
19.5.4 Collision Resolution ... 378
19.5.5 PCI Control Register (SDR0_PCI0) ... 378
19.6 PCI Bridge Configuration Registers ... 379
19.6.1 PCI Bridge Register Summary ... 379
19.6.2 PCI Bridge Local Configuration Registers ... 381
19.6.2.1 PMM 0 Local Address Register (PCIL0_PMM0LA) ... 381
19.6.2.2 PMM 0 Mask/Attribute Register (PCIL0_PMM0MA) ... 381
19.6.2.3 PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA) ... 382
19.6.2.4 PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA) ... 382
19.6.2.5 PMM 1 Local Address Register (PCIL0_PMM1LA) ... 382
19.6.2.6 PMM 1 Mask/Attribute Register (PCIL0_PMM1MA) ... 382
19.6.2.7 PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA) ... 383
19.6.2.8 PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA) ... 383
19.6.2.9 PMM 2 Local Address Register (PCIL0_PMM2LA) ... 383
19.6.2.10 PMM 2 Mask/Attribute Register (PCIL0_PMM2MA) ... 384
19.6.2.11 PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA) ... 384
19.6.2.12 PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA) ... 384
19.6.2.13 PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS) ... 385
19.6.2.14 PTM 1 Local Address Register (PCIL0_PTM1LA) ... 385
19.6.2.15 PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS) ... 385
19.6.2.16 PTM 2 Local Address Register (PCIL0_PTM2LA) ... 386
19.6.3 PCI Configuration Registers ... 386

19.6.3.1 PCI Configuration Address Register (PCIC0_CFGADDR) 386
19.6.3.2 PCI Configuration Data Register (PCIC0_CFGDATA) 387
19.6.3.3 PCI Vendor ID Register (PCIC0_VENDID) 387
19.6.3.4 PCI Device ID Register (PCIC0_DEVID) 387
19.6.3.5 PCI Command Register (PCIC0_CMD) 388
19.6.3.6 PCI Status Register (PCIC0_STATUS) 388
19.6.3.7 PCI Revision ID Register (PCIC0_REVID) 390
19.6.3.8 PCI Class Register (PCIC0_CLS) 390
19.6.3.9 PCI Cache Line Size Register (PCIC0_CACHELS) 390
19.6.3.10 PCI Latency Timer Register (PCIC0_LATTIM) 390
19.6.3.11 PCI Header Type Register (PCIC0_HDTYPE) 391
19.6.3.12 PCI Built-In Self Test (BIST) Control Register (PCIC0_BIST) 391
19.6.3.13 Unused PCI Base Address Register Space 391
19.6.3.14 PCI PTM 1 BAR (PCIC0_PTM1BAR) 391
19.6.3.15 PCI PTM 2 BAR (PCIC0_PTM2BAR) 392
19.6.3.16 PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID) 393
19.6.3.17 PCI Subsystem ID Register (PCIC0_SBSYSID) 393
19.6.3.18 PCI Capabilities Pointer (PCIC0_CAP) 393
19.6.3.19 PCI Interrupt Line Register (PCIC0_INTLN) 393
19.6.3.20 PCI Interrupt Pin Register (PCIC0_INTPN) 393
19.6.3.21 PCI Minimum Grant Register (PCIC0_MINGNT) 394
19.6.3.22 PCI Maximum Latency Register (PCIC0_MAXLTNCY) 394
19.6.3.23 PCI Interrupt Control/Status Register (PCIC0_ICS) 394
19.6.3.24 Error Enable Register (PCIC0_ERREN) 394
19.6.3.25 Error Status Register (PCIC0_ERRSTS) 395
19.6.3.26 Bridge Options 1 Register (PCIC0_BRDGOPT1) 396
19.6.3.27 PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0) 396
19.6.3.28 PLB Slave Error Syndrome Register 1 (PCIC0_PLBBESR1) 398
19.6.3.29 PLB Slave Error Address Register (PCIC0_PLBBEAR) 399
19.6.3.30 Capability Identifier (PCIC0_CAPID) 399
19.6.3.31 Next Item Pointer (PCIC0_NEXTIPTR) 400
19.6.3.32 Power Management Capabilities (PCIC0_PMC) 400
19.6.3.33 Power Management Control/Status Register (PCIC0_PMCSR) 400
19.6.3.34 PMCSR PCI-to-PCI Bridge Support Extensions (PCIC0_PMCSRBSE) 401
19.6.3.35 PCI Data Register (PCIC0_DATA) 401
19.6.3.36 Bridge Options 2 Register (PCIC0_BRDGOPT2) 401
19.6.3.37 Power Management State Change Request Register (PCIC0_PMSCRR) 402
19.7 Error Handling 403
19.7.1 PLB Unsupported Transfer Type 404
19.7.2 PCI Master Abort 404
19.7.3 Bridge PCI Master Receives Target Abort While PCI Bus Master 404
19.7.4 PCI Target Data Bus Parity Error Detection 405
19.7.5 PCI Master Data Bus Parity Error Detection 405
19.7.6 PCI Address Bus Parity Error While PCI Target 406
19.7.7 PLB Master Bus Error Detection 406
19.8 PCI Power Management Interface 407
19.8.1 Capabilities and Power Management Status and Control Registers 407
19.8.2 Power State Control 407
19.8.3 Changing Power States 407
19.9 PCI Bridge Reset and Initialization 408
19.9.1 Address Map Initialization 408
19.9.2 Other Configuration Register Initialization 410
19.9.3 Target Bridge Initialization 410

19.9.4 Local Processor Boot from PCI Memory 411
19.9.5 Type 0 Configuration Cycles for Other Devices 411
19.10 Timing Diagrams 411
19.10.1 PCI Timing Diagram Descriptions 412
19.10.1.1 PCI Master Burst Read From SDRAM 412
19.10.1.2 PCI Master Burst Write To SDRAM 412
19.10.1.3 CPU Read From PCI Memory Slave, Nonprefetching 412
19.10.1.4 CPU Read From PCI Memory Slave, Prefetching 412
19.10.1.5 CPU Write To PCI Memory Slave 412
19.10.1.6 PCI Memory To SDRAM DMA Transfer 413
19.10.1.7 SDRAM To PCI Memory DMA Transfer 413
19.10.2 Asynchronous 413
19.10.3 Synchronous 434

20. External Bus Controller 465
20.1 Interface Signals 465
20.1.1 Interfacing to Byte and Half-Word Devices 466
20.1.2 Driver Enables 467
20.2 Non-Burst Peripheral Bus Transactions 467
20.2.1 Single Read Transfer 468
20.2.2 Single Write Transfer 469
20.3 Burst Transactions 469
20.3.1 Burst Read Transfer 470
20.3.2 Burst Write Transfer 471
20.4 Device-Paced Transfers 472
20.4.1 Device-Paced Single Read Transfer 473
20.4.2 Device-Paced Single Write Transfer 474
20.4.3 Device-Paced Burst Read Transfer 475
20.4.4 Device-Paced Burst Write Transfer 476
20.5 External Bus Master Interface 477
20.5.1 Arbitration 478
20.5.2 Transaction Overview 479
20.5.3 Single Read and Single Write Transfers 480
20.5.4 Burst Read Transfer 481
20.5.5 Burst Write Transfer 482
20.5.6 External Master Error Interrupts 483
20.6 EBC Registers 483
20.6.1 EBC Configuration Register (EBC0_CFG) 485
20.6.2 Peripheral Bank Configuration Registers (EBC0_B0CR:EBC0_B5CR) 486
20.6.3 Peripheral Bank Access Parameters (EBC0_B0AP:B5AP) 487
20.7 Error Reporting 488
20.7.1 Peripheral Bus Error Address Register (EBC0_BEAR) 489
20.7.2 Peripheral Bus Error Status Register 0 (EBC0_BESR0) 489
20.7.3 Peripheral Bus Error Status Register 1 (EBC0_BESR1) 490

21. NAND Flash Controller 493
21.1 Overview 493
21.2 NDFC Signal Multiplexing 493
21.3 Resetting the Controller 494
21.4 Configuring EBC to Support the NAND Flash Controller 494
21.4.1 EBC0_BxAP 494
21.4.2 EBC0_CFG 494

21.5 Enabling the NAND Flash Controller 494
21.6 Booting from NDFC 494
21.6.1 AutoRead Mode 496
21.8 NDFC Registers 497
21.8.1 Memory Map 498
21.8.2 NDFC Address Register (NDFC0_ADDR) 499
21.8.3 NDFC Command Register (NDFC0_CMD) 499
21.8.4 NDFC Data Register (NDFC0_DATA) 500
21.8.5 NAND Flash ECC Calculation Registers (NDFC0_ECC0:NDFC0_ECC7) 501
21.8.6 NDFC Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR) 502
21.8.7 NDFC Configuration Register (NDFC0_CR) 503
21.8.8 NDFC Status Register (NDFC0_SR) 505
21.8.9 NAND Flash Direct Hardware Control Register (NDFC0_HWCTL) 505
21.8.10 NDFC Revision ID Register (NDFC0_REVID) 506
21.8.11 Linear Data Access Region 507

22. Direct Memory Access Controllers 509
22.1 DMA to PLB4 Controller (DMA2P40) 509
22.1.1 DMA Transfers 509
22.1.1.1 Memory-to-Memory Transfers 510
22.1.1.1 Device-Paced Memory Mode Transfers (Hardware Initiated) 510
22.1.1.1 Software-Initiated Memory-to-Memory Transfers 510
22.1.1.2 Scatter/Gather Transfers 511
22.1.2 Configuration and Status Registers 511
22.1.2.1 DMA to PLB 4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3) 513
22.1.2.2 DMA to PLB 4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3)) 514
22.1.2.3 DMA to PLB 4 Source Address Registers 515
22.1.2.4 DMA to PLB 4 Destination Address Registers 516
22.1.2.5 DMA to PLB 4 Scatter/Gather Descriptor Address Registers 516
22.1.2.6 DMA to PLB 4 Status Register (DMA2P40_SR) 517
22.1.2.7 DMA to PLB 4 Scatter/Gather Command Register (DMA2P40_SGC) 518
22.1.2.8 DMA to PLB 4 Sleep Mode Register (DMA2P40_SLP) 518
22.1.2.9 DMA to PLB 4 Polarity Configuration Register (DMA2P40_POL) 519
22.1.3 Channel Priorities 519
22.1.4 Data Parity During DMA Peripheral Transfers 519
22.1.5 Peripheral and Device-Paced Memory Bursts 519
22.1.6 Errors 520
22.1.6.1 Address Alignment Error 520
22.1.6.2 Burst Count Error 520
22.1.6.3 Burst Prefetch Error 521
22.1.6.4 PLB or OPB Timeout 521
22.1.6.5 Slave Transfer Errors 521
22.1.7 DMA to PLB4 Interrupts 521
22.1.8 Scatter/Gather Transfers 522
22.1.9 Programming the DMA2P40 Controller 523
22.1.9.1 Memory-to-Memory Transfers 523
22.2 DMA to PLB3 Controller (DMA2P30) 525
22.2.1 External Interface Signals 526
22.2.2 Functional Overview 527
22.2.3 Peripheral Mode Transfers 527
22.2.4 Memory-to-Memory Transfers 528
22.2.5 Scatter/Gather Transfers 528
22.2.6 Configuration and Status Registers 529

22.2.6.1 DMA to PLB 3 Polarity Configuration Register (DMA2P30_POL) 530
22.2.6.2 DMA to PLB 3 Sleep Mode Register (DMA2P30_SLP) 531
22.2.6.3 DMA to PLB 3 Status Register (DMA2P30_SR) 531
22.2.6.4 DMA to PLB 3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3) 532
22.2.6.5 DMA to PLB 3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3) 534
22.2.6.6 DMA to PLB 3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3) ... 534
22.2.6.7 DMA to PLB 3 Count Registers (DMA2P30_CT0–DMA2P30_CT3) 534
22.2.6.8 DMA to PLB 3 Scatter/Gather Descriptor Address Registers (DMA2P30_SG0–DMA2P30_SG3) 535
22.2.6.9 DMA to PLB 3 Scatter/Gather Command Register (DMA2P30_SGC) 535
22.2.6.10 DMA to PLB 3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3) 536
22.2.6.11 DMA to PLB 3 Address Decode Register (DMA2P30_ADR) 536
22.2.7 Channel Priorities 536
22.2.8 Data Parity During DMA Peripheral Transfers 537
22.2.9 Errors 537
22.2.10 Address Alignment Error 537
22.2.11 PLB Timeout 538
22.2.12 Slave Errors 538
22.2.13 DMA Interrupts 538
22.2.14 Scatter/Gather Transfers 538
22.2.15 Programming the DMA to PLB3 Controller 540
22.2.16 Peripheral Mode Transfers 540
22.2.16.1 Peripheral-to-Memory Transfer 542
22.2.16.2 Memory-to-Peripheral Transfer 543
22.2.17 Memory-to-Memory Transfers 543
22.2.17.1 Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers 543
22.2.17.2 Software-Initiated Memory-to-Memory Transfers (Non-Device Paced) 544

23. Memory Access Layer 545
23.1 MAL Features 545
23.1.1 MAL Internal Structure 546
23.1.1.1 PLB Master 546
23.1.1.2 OPB Master 546
23.1.1.3 TX Channel Handler 546
23.1.1.4 RX Channel Handler 546
23.1.1.5 TX Channel Arbiter 547
23.1.1.6 RX Channel Arbiter 547
23.1.1.7 TX Common Channel Logic 547
23.1.1.8 RX Common Channel Logic 547
23.1.1.9 Register Map File 547
23.2 MAL0 Interfaces and Channel Assignments 547
23.3 Transmit and Receive Operations 548
23.3.1 Transmit Operations 548
23.3.2 Receive Operations 549
23.4 Buffer Descriptor 550
23.5 Transmit Software Interface 552
23.5.1 Wrapping the BD Table for Transmit 553
23.5.2 Continuous Mode for Transmit 553
23.5.3 Back Up a Packet for Transmit 553
23.5.4 Descriptor Not Valid for Transmit 553
23.5.5 Scroll Descriptors for Transmit 554
23.6 Receive Software Interface 555
23.6.1 Wrapping the BD Table for Receive 555

23.6.2 Continuous Mode for Receive 555
23.6.3 Descriptor Not Valid for Receive 555
23.6.4 Buffer Length for Receive 556
23.7 Buffer Descriptor Status/Control Fields 556
23.7.1 Information from a Software Device Driver Directed to MAL and COMMAC 556
23.7.2 Information from MAL and COMMAC Directed to Software 557
23.7.3 Status/Control Field Handling 557
23.7.4 Status/Control Field Format 557
23.7.5 TX Status/Control Field Format 557
23.7.6 RX Status/Control Field Format 558
23.8 MAL Programming Notes 559
23.8.1 MAL Initialization 559
23.8.2 Interrupts 560
23.8.3 Error Handling 560
23.8.3.1 Error Detection 561
23.8.3.2 Indicated Errors 561
23.8.3.3 Error Handling Registers 562
23.8.3.4 Operational Error Modes 563
23.8.3.5 Resolution of an Error Situation 563
23.8.3.6 Interrupts To Software 564
23.9 MAL Serial Device Control Registers 566
23.9.1 SDR0_MALRBL Register 566
23.9.2 SDR0_MALRBS Register 566
23.9.3 SDR0_MALTBL Register 566
23.9.4 SDR0_MALTBS Register 567
23.10 MAL Registers 567
23.10.1 MAL Configuration Register (MAL0_CFG) 567
23.10.2 Channel Active Set and Reset Registers 569
23.10.3 End of Buffer Interrupt Status Registers 570
23.10.4 MAL Error Status Register (MAL0_ESR) 570
23.10.5 MAL Interrupt Enable Register (MAL0_IER) 572
23.10.6 Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR) 572
23.10.7 Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR) 573
23.10.8 Receive Channel Buffer Size Register (MAL0_RCBSx) 574

24. EMAC to PHY Interface Bridge 575
24.1 ZMII Features 575
24.2 ZMII Bridge Interface Signals 576
24.3 EMAC-ZMII Bridge Interfaces 577
24.3.1 MII Interface 577
24.3.2 RMII Interface 578
24.3.2.1 SMII Interface 578
24.4 ZMII Bridge Registers 579
24.4.1 ZMII Function Enable Register (ZMII0_FER) 579
24.4.2 ZMII Speed Select Register (ZMII0_SSR) 580
24.4.3 ZMII SMII Status Register (ZMII0_SMIISR) 581

25. Ethernet Media Access Controllers 583
25.1 EMAC Features 584
25.2 EMAC Operation 585
25.2.1 MAL Slave Logic 586
25.2.2 OPB Slave Logic 587
25.2.3 FIFO Management Logic 587

25.2.4 Ethernet Address Match Logic 587
25.2.5 Configuration and Status Registers 587
25.2.6 Wake On LAN Logic 587
25.2.7 Ethernet MAC 587
25.2.8 EMAC Loopback Modes 588
25.2.9 Packet Rejection 588
25.2.9.1 EMACx RX Status Register 589
25.2.9.2 EMACx TX Status Register 591
25.2.9.3 EMACx RX Packet Reject Counter 592
25.3 EMAC Transmit Operation 593
25.3.1 Arbitration Between TX Channels 593
25.3.2 Independent Mode 593
25.3.3 Dependent Mode 593
25.3.3.1 MAL TX Descriptor Control/Status Field 594
25.3.3.2 Early Packet Termination during Transmit 595
25.3.3.3 Empty Packets 596
25.3.3.4 Automatic Retransmission of Colliding Packets 596
25.3.3.5 Inter-Packet Gap (IPG) Tuning 596
25.3.3.6 Full-Duplex Operation 596
25.3.3.7 Packet Content Configuration Options 597
25.4 EMAC Receive Operation 599
25.4.1 EMAC – MAL RX Packet Transfer Flow 599
25.4.2 MAL RX Descriptor Status 599
25.4.3 Early Packet Termination during Receive 600
25.4.4 Discarding Packets During Receive 600
25.4.5 Wakeup On Lan (WOL) Support 600
25.4.5.1 EMAC WOL Support 601
25.5 Flow Control 601
25.5.1 MAC Control Packet 602
25.5.2 Control Packet Transmission 602
25.5.3 Integrated Flow Control 603
25.5.4 Control Packet Reception 603
25.6 VLAN Support 604
25.6.1 VLAN Tagged Packet Transmission 605
25.6.2 VLAN Tagged Packet Reception 605
25.6.3 Address Match Mechanism 606
25.6.3.1 Non-WOL Mode 606
25.6.3.2 WOL Mode 608
25.7 EMAC Registers 609
25.7.1 Mode Register 0 (EMACx_MR0) 611
25.7.2 Mode Register 1 (EMACx_MR1) 612
25.7.3 Transmit Mode Register 0 (EMACx_TMR0) 613
25.7.4 Transmit Mode Register 1 (EMACx_TMR1) 614
25.7.4.1 Low-Priority Requests 614
25.7.4.2 Urgent-Priority Requests 614
25.7.5 Receive Mode Register (EMACx_RMR) 615
25.7.6 Interrupt Status Register (EMACx_ISR) 616
25.7.7 Interrupt Status Enable Register (EMACx_ISER) 618
25.7.8 Individual Address High (EMACx_IAHR) 619
25.7.9 Individual Address Low (EMACx_IALR) 620
25.7.10 VLAN TPID Register (EMACx_VTPID) 620
25.7.11 VLAN TCI Register (EMACx_VTCI) 620

25.7.12 Pause Timer Register (EMACx_PTR) 621
25.7.13 Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4) 621
25.7.14 Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4) 621
25.7.15 Last Source Address High (EMACx_LSAH) 621
25.7.16 Last Source Address Low (EMACx_LSAL) 622
25.7.17 Inter-Packet Gap Value Register (EMACx_IPGVR) 622
25.7.18 STA Control Register (EMACx_STACR) 622
25.7.19 Transmit Request Threshold Register (EMACx_TRTR) 623
25.7.20 Receive Low/High Water Mark Register (EMACx_RWMR) 624
25.7.21 Transmitted Octects (EMACx_OCTX) 625
25.7.22 Received Octects Register (EMACx_OCRX) 625
25.8 MII Interface 625
25.8.1 MII Station Management Interface 625
25.9 MAL—EMAC Packet Transfer Flow 625
25.10 Programming Notes 626
25.10.1 Reset and Initialization 626
25.10.1.1 Scenario 1 627
25.10.1.2 Scenario 2 627
25.10.1.3 Scenario 3 627

26. Serial Port Operations 629
26.1 Functional Description 629
26.2 Serial Port Clocking 630
26.3 UART Registers 632
26.3.1 Receiver Buffer Registers (UARTx_RBR) 633
26.3.2 Transmitter Holding Registers (UARTx_THR) 633
26.3.3 Interrupt Enable Registers (UARTx_IER) 633
26.3.4 Interrupt Identification Registers (UARTx_IIR) 634
26.3.5 FIFO Control Registers (UARTx_FCR) 635
26.3.6 Line Control Registers (UARTx_LCR) 635
26.3.7 Modem Control Registers (UARTx_MCR) 636
26.3.8 Line Status Registers (UARTx_LSR) 637
26.3.9 Modem Status Registers (UARTx_MSR) 638
26.3.10 Scratchpad Registers (UARTx_SCR) 639
26.3.11 Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM) 639
26.3.12 UART Configuration Register 0 (SDR0_UART0)) 640
26.3.13 UART Configuration Register 1 (SDR0_UART1)) 641
26.3.14 UART Configuration Register 2 (SDR0_UART2)) 642
26.3.15 UART Configuration Register 3 (SDR0_UART3)) 642
26.4 FIFO Operation 643
26.4.1 Interrupt Mode 643
26.4.1.1 Receiver 643
26.4.1.2 Transmitter 644
26.4.2 Polled Mode 644
26.4.3 UART and Sleep Mode 645
26.5 DMA Operation 645
26.5.1 Transmitter DMA Mode 646
26.5.2 Receiver DMA Mode 649

27. Serial Peripheral Interface 653
27.1 Functional Description 653
27.1.1 SPI Interrupt Operation 654
27.1.2 Loopback 654

27.1.3 Typical Operation (Exchange of Data) 655
27.2 SPI Registers 655
27.2.1 SPI Receive Data Register (SPI0_RxD) 656
27.2.2 SPI Transmit Data Register (SPI0_TxD) 656
27.2.3 SPI Control Register (SPI0_CR) 656
27.2.4 SPI Clock Divisor Modulus Register (SPI0_CDM) 657
27.2.5 SPI Status Register (SPI0_SR) 657
27.2.6 SPI Mode Register (SPI0_MODE) 658

28. IIC Bus Interface 659
28.1 Addressing 659
28.1.1 Addressing Modes 659
28.1.2 Seven-Bit Addresses 659
28.1.3 Ten-Bit Addresses 660
28.2 IIC Registers 661
28.3 IIC Register Descriptions 662
28.3.1 IICx Master Data Buffer (IICx_MDBUF) 662
28.3.2 IICx Slave Data Buffer (IICx_SDBUF) 662
28.3.3 IICx Low Master Address Register (IICx_LMADR) 663
28.3.4 IICx High Master Address Register (IICx_HMADR) 664
28.3.5 IICx Control Register (IICx_CNTL) 664
28.3.6 IICx Mode Control Register (IICx_MDCNTL) 665
28.3.7 IICx Status Register (IICx_STS) 666
28.3.8 IICx Extended Status Register (IICx_EXTSTS) 668
28.3.9 IICx Low Slave Address Register (IICx_LSADR) 669
28.3.10 IICx High Slave Address Register (IICx_HSADR) 670
28.3.11 IICx Clock Divide Register (IICx_CLKDIV) 670
28.3.12 IICx Interrupt Mask Register (IICx_INTRMSK) 671
28.3.13 IICx Transfer Count Register (IICx_XFRCNT) 672
28.3.14 IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) 673
28.3.15 IICx Direct Control Register (IICx_DIRECTCNTL) 674
28.3.16 IICx Interrupt Register (IICx_INTR) 675
28.4 Interrupt Handling 676
28.5 General Considerations 676

29. GPIO Operations 679
29.1 Overview 679
29.2 Features 681
29.3 Clock and Power Management 681
29.4 GPIO Registers 681
29.5 GPIO Register Reset Values 683
29.6 Detailed Register Descriptions 683
29.6.1 GPIO Output Register (GPIOx_OR) 683
29.6.2 GPIO Three-State Control Register (GPIOx_TCR) 683
29.6.3 GPIO Output Select Registers (GPIOx_OSRH, GPIOx_OSRL) 683
29.6.4 GPIO Three-State Select Registers (GPIOx_TSRH, GPIOx_TSRL) 684
29.6.5 GPIO Open Drain Register (GPIOx_ODR) 684
29.6.6 GPIO Input Register (GPIOx_IR) 685
29.6.7 GPIO Input Select Registers (GPIOx_ISRnH, GPIOx_ISRnL) 685
29.6.8 GPIO Receive Registers (GPIOx_RR1, GPIOx_RR3) 686
29.6.9 Control of GPIO Signals 49–63 686
29.7 GPIO Signal Multiplexing 687

30. Universal Serial Bus Interfaces 693
30.1 Attributions 693
30.2 USB Overview 693
30.2.1 PPC440EP USB Implementation 694
30.2.2 USB Configuration Register (SDR0_USB0) 695
30.3 USB 1.1 Host Interface 695
30.3.1 Features 696
30.3.2 References 697
30.3.3 USB OHCI Registers 697
30.3.4.1 HcRevision Register 700
30.3.5.1 HcHCCA Register 707
30.3.6.1 HcFmInterval Register 714
30.3.7.1 HcRhDescriptorA Register 719
30.3.8.1 OPBMC Register 725
30.3.9 Programming Notes 726
30.4 USB 2.0 Device Interface 727
30.4.1 Features 727
30.4.2 References 729
30.4.3 Software Interface 729
30.4.4 Memory Map Definitions 730
30.4.5 Common Registers 730
30.4.5.1 USB2D0_INTRIN 731
30.4.5.2 USB2D0_POWER 731
30.4.5.3 USB2D0_FADDR 733
30.4.5.4 USB2D0_INTRINE 734
30.4.5.5 USB2D0_INTROUT 734
30.4.5.6 USB2D0_INTRUSBE 735
30.4.5.7 USB2D0_INTRUSB 736
30.4.5.8 USB2D0_INTROUTE 737
30.4.5.9 USB2D0_TESTMODE 738
30.4.5.10 USB2D0_INDEX 739
30.4.5.11 USB2D0_FRAME 739
30.4.6 Indexed Registers 740
30.4.6.1 USB2D0_INCSR0 741
30.4.6.2 USB2D0_INCSR 742
30.4.6.3 USB2D0_INMAXP 744
30.4.6.4 USB2D0_OUTCSR 745
30.4.6.5 USB2D0_OUTMAXP 747
30.4.6.6 USB2D0_OUTCOUNT0 748
30.4.6.7 USB2D0_OUTCOUNT 748
30.4.7 FIFOs 749

Part V. Reference 751

31. Instruction Set 753

32. Floating Point Instruction Set 755
32.1 Instruction Set Portability 755
32.2 Instruction Formats 755
32.3 Pseudocode 756
32.3.1 Operator Precedence 758
32.4 Register Usage 758
32.5 Floating-Point Instructions 758

32.6 Alphabetical Instruction Listing 758

33. Register Summary 811
33.1 Reserved Fields 811
33.2 Device Control Registers 811
33.3 Memory Mapped Registers 811
33.4 Alphabetical Listing of Chip Control and Peripheral Core Registers 811

34. Signal Summary 829
34.1 Signals Listed Alphabetically 829
34.2 Signals by Interface Category 829

Appendix A. Floating Point Instruction Summary 831
A.1 Instruction Set – Alphabetical 831
A.2 Instructions Sorted by Opcode 835
A.3 Instruction Formats 836
A.3.1 Instruction Fields 837
A.3.2 Instruction Format Diagrams 838
A.3.2.1 I-Form 839
A.3.2.2 B-Form 839
A.3.2.3 SC-Form 839
A.3.2.4 D-Form 839
A.3.2.5 X-Form 840
A.3.2.6 XL-Form 840
A.3.2.7 XFL-Form 841
A.3.2.8 XFX-Form 841
A.3.2.9 X0-Form 841
A.3.2.10 M-Form 841

Index 843

Revision Log 855

Figures

Figure 2-1. Alternate PLB4 Master Priority Register (SDR0_AMP0) 66
Figure 2-2. Alternate PLB3 Master Priority Register 1 (SDR0_AMP1) 67
Figure 2-3. PPC440 CPU Register (SDR0_CP440) 68
Figure 2-4. Master Interrupt Request Register 0 (SDR0_MIRQ0) 70
Figure 2-5. Slave Address Pipeline Register (SDR0_SLPIPE0) 71
Figure 2-6. PLB Crossbar Arbiter Interconnection 72
Figure 2-7. PLB4A0 Arbiter 0 Revision ID Register (PLB4A0_REVID) 73
Figure 2-8. PLB4 Arbiter Control Register (PLB4An_ACR) 74
Figure 2-9. PLB4 Error Status Register Low (PLB4An_ESRL) 77
Figure 2-10. PLB4 Error Address Register Low (PLB4An_EARL) 78
Figure 2-11. PLB4 Error Address Register High (PLB4An_EARH) 78
Figure 2-12. PLB4 Crossbar Control Register (PLB4A0_CCR) 79
Figure 2-13. PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL) 80
Figure 2-14. PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH) 80
Figure 2-15. PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0) 81
Figure 2-16. PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1) 83
Figure 2-17. PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG) 84
Figure 2-18. PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR) 85
Figure 2-19. PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR) 86
Figure 2-20. PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID) 86
Figure 2-21. PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL) 87
Figure 2-22. PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH) 87
Figure 2-23. PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0) 88
Figure 2-24. PLB3 to PLB4 Bridge Error Status Register 1 (P3P4BI0_BESR1) 90
Figure 2-25. PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG) 91
Figure 2-26. PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR) 93
Figure 2-27. PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR) 93
Figure 2-28. PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID) 93
Figure 2-29. PLB4 to OPB Bridge Error Status Register 0 (PLB42OPB0_BESR0) 94
Figure 2-30. PLB4 to OPB Bridge Error Address Register Low (PLB42OPB0_BEARL) 96
Figure 2-31. PLB4 to OPB Bridge Error Address Register High (PLB42OPB0_BEARH) 96
Figure 2-32. PLB4 to OPB Bridge Error Status Register 1 (PLB42OPB0_BESR1) 96
Figure 2-33. PLB4 to OPB Bridge Configuration Register (PLB42OPB0_CFG) 97
Figure 2-34. PLB4 to OPB Bridge Burst Latency Timer Register (PLB42OPB0_LATENCY) 98
Figure 2-35. PLB4 to OPB Bridge Revision ID Register (PLB42OPB0_REVID) 98
Figure 2-36. PLB3 Arbiter 0 Revision ID Register (PLB3A0_REVID) 99
Figure 2-37. PLB3 Error Status Register (PLB3A0_BESR) 99
Figure 2-38. PLB3 Error Address Register (PLB3A0_BEAR) 101
Figure 2-39. PLB3 Arbiter Control Register (PLB3A0_ACR) 101
Figure 2-40. PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0) 103

Figure 2-41. PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR) 104
Figure 2-42. PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID) 104
Figure 2-43. PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1) 104
Figure 2-44. OPB Arbiter Priority Register (OPBAx_PR) 106
Figure 2-45. OPB Arbiter Control Register (OPBAx_CR) 107
Figure 2-46. OPB to PLB3 Bridge Control Register (OPB2PLB30_BCTRL) 108
Figure 2-47. OPB to PLB3 Bridge Status Register (OPB2PLB30_BSTAT) 108
Figure 2-48. OPB to PLB3 Bridge Revision ID Register (OPB2PLB30_REVID) 108
Figure 3-1. Logical Organization of PLB Performance Monitor 112
Figure 3-2. PPM Master Event Counter Logic 113
Figure 3-3. PPM Slave Event Counter Logic 114
Figure 3-4. PPM Generic Counter Logic 115
Figure 3-5. PPM Duration Counter Logic 116
Figure 3-6. PPM Configuration Address Register (PPM0_CFGADDR) 121
Figure 3-7. PPM Configuration Data Register (PPM0_CFGDATA) 121
Figure 3-8. Interrupt Status Register (PPM0_ISR) 122
Figure 3-9. Control Register (PPM0_CR) 123
Figure 3-10. Cycle Counter Register (PPM0_CCR) 125
Figure 3-11. Upper Address Register (PPM0_UAR) 125
Figure 3-12. Lower Address Register (PPM0_LAR) 125
Figure 3-13. Upper Address Mask Register (PPM0_UAMR) 126
Figure 3-14. Lower Address Mask Register (PPM0_LAMR) 126
Figure 3-15. Revision ID Register (PPM0_RIDR) 126
Figure 3-16. Master Event Counter Selection Register 0:7 (PPM0_MCSR0:PPM0_MCSR7) 127
Figure 3-17. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0:PPM0_SCSR3) 128
Figure 3-18. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0:PPM0_GCSR3) 130
Figure 3-19. Master Event Counter Registers 0:3 (PPM0_MCR0:PPM0_MCR3) 130
Figure 3-20. Slave Event Counter Registers 0:3 (PPM0_SCR0:PPM0_SCR3) 131
Figure 3-21. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0:PPM0_GCR3) 131
Figure 3-22. Duration Counter Selection Registers 0:1 (PPM0_DCSR0:PPM0_DCSR1) 131
Figure 3-23. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0:PPM0_DCMXR1) 132
Figure 3-24. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0:PPM0_DCMNR1) 132
Figure 3-25. Duration Total Value Register 0:1 (PPMx_DCTVR0:PPM0_DCTVR1) 133
Figure 3-26. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0:PPM0_DCOTR1) 133
Figure 5-1. Floating-Point Registers (FPR0:FPR31) 162
Figure 5-2. Floating-Point Status and Control Register (FPSCR) 162
Figure 5-3. Approximation to Real Numbers 166
Figure 5-4. Selection of z1 and z2 171
Figure 8-1. PPC440EP Power-on Reset Process 190
Figure 8-2. PPC440EP Reset 191
Figure 8-3. Initial Configuration Register (CPR0_ICFG) 196

Figure 8-4. SDR Configuration Address Register (SDR0_CFGADDR) 196
Figure 8-5. SDR Configuration Data Register (SDR0_CFGDATA) 196
Figure 8-6. Electronic Chip ID Register 0 (SDR0_ECID0) 197
Figure 8-7. Electronic Chip ID Register 1 (SDR0_ECID1) 197
Figure 8-8. Electronic Chip ID Register 1 (SDR0_ECID2) 197
Figure 8-9. Electronic Chip ID Register 3 (SDR0_ECID3) 197
Figure 8-10. Pin Function Control Register 0 (SDR0_PFC0) 197
Figure 8-11. Pin Function Control Register 1 (SDR0_PFC1) 198
Figure 8-12. Slave Address Pipeline Register (SDR0_SLPIPE0) 199
Figure 8-13. Soft Reset Register 0 (SDR0_SRST0) 199
Figure 8-14. Soft Reset Register 1(SDR0_SRST1) 200
Figure 8-15. Miscellaneous Function Register (SDR0_MFR) 200
Figure 8-16. Hardware Self Refresh Register (SDR0_HSF) 201
Figure 9-1. IIC Bootstrap Controller Flow 214
Figure 9-2. Pin Strapping Register (SDR0_PINSTP) 217
Figure 9-3. Serial Device Controller Settings Register (SDR0_SDCS0) 217
Figure 9-4. Serial Device Strap Register 0 (SDR0_SDSTP0) 218
Figure 9-5. Serial Device Strap Register 1 (SDR0_SDSTP1) 219
Figure 9-6. Serial Device Strap Register 2 (SDR0_SDSTP2) 220
Figure 9-7. Serial Device Strap Register 3 (SDR0_SDSTP3) 221
Figure 9-8. Serial Device Strap Register 0 (SDR0_CUST0) 221
Figure 9-9. Custom Configuration Register 1 (SDR0_CUST1) 222
Figure 9-10. EBC Configuration Register (SDR0_EBC0) 222
Figure 10-1. Cascaded UIC Organization 223
Figure 10-2. UIC0 Status Register (UIC0_SR) 227
Figure 10-3. UIC1 Status Register (UIC1_SR) 229
Figure 10-4. UIC0 Enable Register (UIC0_ER) 231
Figure 10-5. UIC1 Enable Register (UIC1_ER) 233
Figure 10-6. UIC0 Critical Register (UIC0_CR) 235
Figure 10-7. UIC1 Critical Register (UIC1_CR) 237
Figure 10-8. UIC0 Polarity Register (UIC0_PR) 239
Figure 10-9. UIC1 Polarity Register (UIC1_PR) 241
Figure 10-10. UIC0 Trigger Register (UIC0_TR) 244
Figure 10-11. UIC1 Trigger Register (UIC1_TR) 246
Figure 10-12. UIC0 Masked Status Register (UIC0_MSR) 248
Figure 10-13. UIC1 Masked Status Register (UIC1_MSR) 251
Figure 10-14. UIC0 Vector Configuration Register (UIC0_VCR) 253
Figure 10-15. UIC1 Vector Configuration Register (UIC1_VCR) 254
Figure 10-16. UIC Vector Register (UIC0_VR) 254
Figure 10-17. UIC1 Vector Register (UIC1_VR) 256
Figure 12-1. Condition Register (CR) 270

Figure 14-1. Time Base Counter and Compare Register 276
Figure 14-2. GPT Time Base Counter Register (GPT0_TBC) 277
Figure 14-3. GPT Interrupt Mask Register (GPT0_IM) 278
Figure 14-4. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC) 279
Figure 14-5. GPT Interrupt Enable Register (GPT0_IE) 279
Figure 14-6. GPT Compare Timer Register (GPT0_COMP0 - GPT0_COMP6) 280
Figure 14-7. GPT Compare Mask Register (GPT0_MASK0:GPT0_MASK6) 280
Figure 14-8. Down Count Timer Register (GPT0_DCT0) 280
Figure 14-9. Down Count Timer Register (GPT0_DCIS) 281
Figure 15-1. PPC440EP System Clocking 284
Figure 15-2. Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR) 294
Figure 15-3. Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA) 294
Figure 15-4. Clocking Update Register (CPR0_CLKUPD) 295
Figure 15-5. PLL Control Register (CPR0_PLLC0) 295
Figure 15-6. PLL Divisor Register (CPR0_PLLD) 296
Figure 15-7. Primary A Divisor Register (CPR0_PRIMAD0) 297
Figure 15-8. Primary B Divisor Register (CPR0_PRIMBD0) 298
Figure 15-9. OPB Clock Divisor Register (CPR0_OPBD0) 298
Figure 15-10. Peripheral Clock Divisor Register (CPR0_PERD0) 299
Figure 15-11. MAL Clock Divisor Register (CPR0_MALD) 299
Figure 15-12. Sync PCI Clock Divisor Register (CPR0_SPCID) 299
Figure 16-1. CPM Enable Register (CPM0_ER) 302
Figure 16-2. CPM Force Register (CPM0_FR) 303
Figure 16-3. CPM Force Register (CPM0_SR) 304
Figure 18-1. DDR SDRAM Controller Signals 312
Figure 18-2. SDRAM Configuration Address Register (SDRAM0_CFGADDR) 313
Figure 18-3. SDRAM Configuration Data Register (SDRAM0_CFGDATA) 313
Figure 18-4. Bus Error Syndrome Register 0 (SDRAM0_BESR0) 317
Figure 18-5. Bus Error Syndrome Register 1 (SDRAM0_BESR1) 318
Figure 18-6. Bus Error Address Register (SDRAM0_BEAR) 319
Figure 18-7. Master Write Interrupt (SDRAM0_MIRQ) 320
Figure 18-8. PLB Slave Interface Options (SDRAM0_SLIO) 321
Figure 18-9. Memory Controller Options 0 (SDRAM0_CFG0) 321
Figure 18-10. Memory Controller Options 1 (SDRAM0_CFG1) 323
Figure 18-11. DDR_SDRAM Device Options (SDRAM0_DEVOPT) 323
Figure 18-12. Memory Controller Status (SDRAM0_MCSTS) 323
Figure 18-13. Refresh Timing Register (SDRAM0_RTR) 324
Figure 18-14. Power Management Idle Timer (SDRAM0_PMIT) 324
Figure 18-15. PLB UABus Base Address (SDRAM0_UABBA) 325
Figure 18-16. Memory 0-3 Configuration (SDRAM0_B0CR:SDRAM0_B3CR) 325
Figure 18-17. SDRAM Timing Register 0 (SDRAM0_TR0) 326

Figure 18-18. DDR SDRAM Controller Read Data Path Structure 328
Figure 18-19. Read Sample Cycle: CL = 2 330
Figure 18-20. Read Sample Cycle: CL = 2.5 332
Figure 18-21. Read Sample Cycle: CL = 3 333
Figure 18-22. SDRAM Timing Register 1 (SDRAM0_TR1) 333
Figure 18-23. CLKTR Phase Advance and Delay 335
Figure 18-24. DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR) 336
Figure 18-25. DDR Delay Line Register (SDR0_DDRDL0) 336
Figure 18-26. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR) 337
Figure 18-27. CLKTR 0 Degree Phase Advance and WDDCTR 0 Degree Phase Advance 338
Figure 18-28. CLKTR 90 Degree Phase Advance and WDDCTR 90/0 Degree Phase Advance 338
Figure 18-29. CLKTR 180 Degree Phase Advance and WDDCTR 180/90 Degree Phase Advance 339
Figure 18-30. SDRAM0_WDDCTR Delay 339
Figure 18-31. Delay Line Calibration Register (SDRAM0_DLYCAL) 340
Figure 18-32. ECC Error Status Register (SDRAM0_ECCESR) 340
Figure 18-33. Controller ID Register (SDRAM0_CID) 341
Figure 18-34. Revision ID Register (SDRAM0_RID) 341
Figure 18-35. Activate - Read - Precharge - Activate 348
Figure 18-36. Activate - Write - Precharge - Activate 348
Figure 18-37. Precharge All - Activate 349
Figure 18-38. Auto (CBR) Refresh 350
Figure 18-39. Self-Refresh Entry 351
Figure 18-40. Self-Refresh Exit 352
Figure 19-1. PCI Bridge Block Diagram 364
Figure 19-2. PLB-to-PCI Half-Bridge Block Diagram 365
Figure 19-3. PCI-to-PLB Half-Bridge Block Diagram 366
Figure 19-4. Arbitration Structure 366
Figure 19-5. PMM Register Sets Map PLB Address Space to PCI Address Space 369
Figure 19-6. PTM Register Sets Map PCI Address Space to PLB Address Space 370
Figure 19-7. PCI Control Register (SDR0_PCI0) 378
Figure 19-8. PMM 0 Local Address Register (PCIL0_PMM0LA) 381
Figure 19-9. PMM 0 Mask/Attribute Register (PCIL0_PMM0MA) 381
Figure 19-10. PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA) 382
Figure 19-11. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA) 382
Figure 19-12. PMM 1 Local Address Register (PCIL0_PMM1LA) 382
Figure 19-13. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA) 383
Figure 19-14. PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA) 383
Figure 19-15. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA) 383
Figure 19-16. PMM 2 Local Address Register (PCIL0_PMM2LA) 384
Figure 19-17. PMM 2 Mask/Attribute Register (PCIL0_PMM2MA) 384
Figure 19-18. PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA) 384

Figure 19-19. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA) 385
Figure 19-20. PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS) 385
Figure 19-21. PTM 2 Local Address Register (PCIL0_PTM1LA) 385
Figure 19-22. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS) 385
Figure 19-23. PTM 2 Local Address Register (PCIL0_PTM2LA) 386
Figure 19-24. PCI Configuration Address Register (PCIC0_CFGADDR) 386
Figure 19-25. PCI Configuration Data Register (PCIC0_CFGDATA) 387
Figure 19-26. PCI Vendor ID Register (PCIC0_VENDID) 387
Figure 19-27. PCI Device ID Register (PCIC0_DEVID) 387
Figure 19-28. PCI Command Register (PCIC0_CMD) 388
Figure 19-29. PCI Status Register (PCIC0_STATUS) 389
Figure 19-30. PCI Revision ID Register (PCIC0_REVID) 390
Figure 19-31. PCI Class Register (PCIC0_CLS) 390
Figure 19-32. PCI Cache Line Size Register (PCIC0_CACHELS) 390
Figure 19-33. PCI Latency Timer Register (PCIC0_LATTIM) 390
Figure 19-34. PCI Header Type Register (PCIC0_HDTYPE) 391
Figure 19-35. PCI Built-in Self Test Control Register (PCIC0_BIST) 391
Figure 19-36. PCI PTM 1 BAR Register (PCIC0_PTM1BAR) 392
Figure 19-37. PCI PTM 2 BAR Register (PCIC0_PTM2BAR) 392
Figure 19-38. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID) 393
Figure 19-39. PCI Subsystem ID Register (PCIC0_SBSYSID) 393
Figure 19-40. PCI Capabilities Pointer (PCIC0_CAP) 393
Figure 19-41. PCI Interrupt Line Register (PCIC0_INTLN) 393
Figure 19-42. PCI Interrupt Pin Register (PCIC0_INTPN) 393
Figure 19-43. PCI Minimum Grant Register (PCIC0_MINGNT) 394
Figure 19-44. PCI Maximum Latency Register (PCIC0_MAXLTNCY) 394
Figure 19-45. PCI Interrupt Control/Status Register 394
Figure 19-46. PCI Error Enable Register (PCIC0_ERREN) 394
Figure 19-47. PCI Error Status Register (PCIC0_ERRSTS) 395
Figure 19-48. PCI Bridge Options 1 Register (PCIC0_BRDGOPT1) 396
Figure 19-49. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0) 397
Figure 19-50. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1) 398
Figure 19-51. PLB Slave Error Address Register (PCIC0_PLBBEAR) 399
Figure 19-52. PCI Capability Identifier (PCIC0_CAPID) 399
Figure 19-53. PCI Next Item Pointer (PCIC0_NEXTIPTR) 400
Figure 19-54. Power Management Capabilities Register (PCIC0_PMC) 400
Figure 19-55. Power Management Control/Status Register (PCIC0_PMCSR) 401
Figure 19-56. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBSE) 401
Figure 19-57. PCI Data Register (PCIC0_DATA) 401
Figure 19-58. PCI Bridge Options 2 Register (PCIC0_BRDGOPT2) 402
Figure 19-59. Power Management State Change Request Register (PCIC0_PMSCRR) 402

Figure 19-60. Example Address Map 409
Figure 19-61. PCI Master Burst Read From SDRAM 413
Figure 19-62. PCI Master Burst Read From SDRAM (Continued) 414
Figure 19-63. PCI Master Burst Read From SDRAM (Continued) 415
Figure 19-64. PCI Master Burst Read From SDRAM (Continued) 416
Figure 19-65. PCI Master Burst Write To SDRAM 417
Figure 19-66. PCI Master Burst Write To SDRAM (Continued) 418
Figure 19-67. PCI Master Burst Write To SDRAM (Continued) 419
Figure 19-68. PCI Master Burst Write To SDRAM (Continued) 420
Figure 19-69. CPU Read From PCI Memory Slave, Nonprefetching 421
Figure 19-70. CPU Read From PCI Memory Slave, Nonprefetching (Continued) 422
Figure 19-71. CPU Read From PCI Memory Slave, Prefetching 423
Figure 19-72. CPU Read From PCI Memory Slave, Prefetching (Continued) 424
Figure 19-73. CPU Read From PCI Memory Slave, Prefetching (Continued) 425
Figure 19-74. CPU Read From PCI Memory Slave, Prefetching (Continued) 426
Figure 19-75. CPU Write To PCI Memory Slave 427
Figure 19-76. CPU Write To PCI Memory Slave (Continued) 428
Figure 19-77. CPU Write To PCI Memory Slave (Continued) 429
Figure 19-78. PCI Memory To SDRAM DMA Transfer 430
Figure 19-79. PCI Memory To SDRAM DMA Transfer (Continued) 431
Figure 19-80. SDRAM To PCI Memory DMA Transfer 432
Figure 19-81. SDRAM To PCI Memory DMA Transfer (Continued) 433
Figure 19-82. SDRAM To PCI Memory DMA Transfer (Continued) 434
Figure 19-83. PCI Master Burst Read From SDRAM 435
Figure 19-84. PCI Master Burst Read From SDRAM (Continued) 436
Figure 19-85. PCI Master Burst Read From SDRAM (Continued) 437
Figure 19-86. PCI Master Burst Read From SDRAM (Continued) 438
Figure 19-87. PCI Master Burst Read From SDRAM (Continued) 439
Figure 19-88. PCI Master Burst Read From SDRAM (Continued) 440
Figure 19-89. PCI Master Burst Read From SDRAM (Continued) 441
Figure 19-90. PCI Master Burst Write To SDRAM 442
Figure 19-91. PCI Master Burst Write To SDRAM (Continued) 443
Figure 19-92. PCI Master Burst Write To SDRAM (Continued) 444
Figure 19-93. PCI Master Burst Write To SDRAM (Continued) 445
Figure 19-94. PCI Master Burst Write To SDRAM (Continued) 446
Figure 19-95. PCI Master Burst Write To SDRAM (Continued) 447
Figure 19-96. CPU Read From PCI Memory Slave, Nonprefetching 448
Figure 19-97. CPU Read From PCI Memory Slave, Nonprefetching (Continued) 449
Figure 19-98. CPU Read From PCI Memory Slave, Prefetching 450
Figure 19-99. CPU Read From PCI Memory Slave, Prefetching (Continued) 451
Figure 19-100. CPU Read From PCI Memory Slave, Prefetching (Continued) 452

Figure 19-101. CPU Read From PCI Memory Slave, Prefetching (Continued) 453
Figure 19-102. CPU Write To PCI Memory Slave 454
Figure 19-103. CPU Write To PCI Memory Slave (Continued) 455
Figure 19-104. CPU Write To PCI Memory Slave (Continued) 456
Figure 19-105. CPU Write To PCI Memory Slave (Continued) 457
Figure 19-106. PCI Memory To SDRAM DMA Transfer 458
Figure 19-107. PCI Memory To SDRAM DMA Transfer (Continued) 459
Figure 19-108. PCI Memory To SDRAM DMA Transfer (Continued) 460
Figure 19-109. SDRAM To PCI Memory DMA Transfer 461
Figure 19-110. SDRAM To PCI Memory DMA Transfer (Continued) 462
Figure 19-111. SDRAM To PCI Memory DMA Transfer (Continued) 463
Figure 20-1. EBC Signals 465
Figure 20-2. Attachment of Devices of Various Widths to the Peripheral Data Bus 466
Figure 20-3. Single Read Transfer 468
Figure 20-4. Single Write Transfer 469
Figure 20-5. Burst Read Transfer 471
Figure 20-6. Burst Write Transfer 472
Figure 20-7. Device-Paced Single Read Transfer 473
Figure 20-8. Device-Paced Single Write Transfer 474
Figure 20-9. Device-Paced Burst Read Transfer 475
Figure 20-10. Device-Paced Burst Write Transfer 477
Figure 20-11. Sample External Bus Master System 478
Figure 20-12. External Master Arbitration, Single Read and Single Write 481
Figure 20-13. External Master Burst Read 482
Figure 20-14. External Master Burst Write 483
Figure 20-15. EBC Configuration Address Register (EBC0_CFGADDR) 483
Figure 20-16. EBC Configuration Data Register (EBC0_CFGDATA) 484
Figure 20-17. EBC Configuration Register (EBC0_CFG) 485
Figure 20-18. Peripheral Bank Configuration Registers (EBC0_B0CR–EBC0_B5CR) 486
Figure 20-19. Peripheral Bank Access Parameters (EBC0_B0AP–EBC0_B5AP) 487
Figure 20-20. Peripheral Bus Error Address Register (EBC0_BEAR) 489
Figure 20-21. Peripheral Bus Error Status Register 0 (EBC0_BESR0) 489
Figure 20-22. Peripheral Bus Error Status Register 1 (EBC0_BESR1) 491
Figure 21-1. NAND Flash Address Register (NDFC0_ADDR) 499
Figure 21-2. NAND Flash Command Register (NDFC0_CMD) 499
Figure 21-3. NAND Flash Data Register (NDFC0_DATA) 500
Figure 21-4. NAND Flash ECC Calculation Register (NDFC0_ECC0:NDFC0_ECC7) 501
Figure 21-5. NAND Flash Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR) 502
Figure 21-6. NAND Flash Configuration Register (NDFC0_CR) 503
Figure 21-7. NAND Flash Status Register (NDFC0_SR) 505
Figure 21-8. NAND Flash Direct Hardware Control Register (NDFC0_HWCTL) 506

Figure 21-9. NAND Flash Revision ID Register (NDFC0_REVID) 506
Figure 22-1. DMA to PLB4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3) 513
Figure 22-2. DMA to PLB4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3) 515
Figure 22-3. DMA to PLB4 Source Address High Registers (DMA2P40-SAH0-DMA2P40-SAH3) 516
Figure 22-4. DMA to PLB4 Source Address Low Registers (DMA2P40-SAL0-DMA2P40-SAL3) 516
Figure 22-5. DMA to PLB4 Destination Address High Registers (DMA2P40_DAH0-DMA2P40_DAH3) 516
Figure 22-6. DMA to PLB4 Destination Address Low Registers (DMA2P40_DAL0-DMA2P40_DAL3) ..516
Figure 22-7. DMA to PLB4 Scatter/Gather Desc Adr High Reg (DMA2P40_SGH0-DMA2P40_SGH3) .517
Figure 22-8. DMA to PLB4 Scatter/Gather Desc Adr Low Reg (DMA2P40_SGL0-DMA2P40_SGL3) ...517
Figure 22-9. DMA to PLB4 Status Register (DMA2P40_SR) 517
Figure 22-10. DMA to PLB4 Scatter/Gather Command Register (DMA2P40_SGC) 518
Figure 22-11. DMA to PLB4 Sleep Mode Register (DMA2P40_SLP) 519
Figure 22-12. Scatter/Gather Descriptor Table 522
Figure 22-13. DMA to PLB 3 External Bus Controller Signals 527
Figure 22-14. DMA to PLB3 Polarity Configuration Register (DMA2P30_POL) 530
Figure 22-15. DMA to PLB3 Sleep Mode Register (DMA2P30_SLP) 531
Figure 22-16. DMA to PLB3 Status Register (DMA2P30_SR) 532
Figure 22-17. DMA to PLB3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3) 532
Figure 22-18. DMA to PLB3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3) 534
Figure 22-19. DMA to PLB3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3) 534
Figure 22-20. DMA to PLB3 Count Registers (DMA2P30_CT0–DMA2P30_CT3) 535
Figure 22-21. DMA to PLB3 Scatter/Gather Desc. Addr. Regs (DMA2P30_SG0–DMA2P30_SG3) 535
Figure 22-22. DMA to PLB3 Scatter/Gather Command Register (DMA2P30_SGC) 536
Figure 22-23. DMA to PLB3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3) 536
Figure 22-24. DMA to PLB3 Address Decode Register (DMA2P30_ADR) 536
Figure 22-25. Peripheral to Memory DMA Transfers 541
Figure 22-26. Memory to Peripheral DMA Transfers 542
Figure 23-1. MAL Internal Structure 546
Figure 23-2. Transmit Operations 548
Figure 23-3. Receive Operations 549
Figure 23-4. Buffer Descriptor Structure 551
Figure 23-5. Packet Memory Structure 552
Figure 23-6. TX Status Control Field 558
Figure 23-7. RX Status Control Field 559
Figure 23-8. Error Status Register Field 563
Figure 23-9. MAL Error Processing 565
Figure 23-10. MAL Receive Burst Length Register (SDR0_MALRBL) 566
Figure 23-11. MAL Transmit Burst Length Register (SDR0_MALTBL) 566
Figure 23-12. MAL Configuration Register (MAL0_CFG) 568
Figure 23-13. MAL Transmit Channel Active Set Register (MAL0_TXCASR) 569
Figure 23-14. MAL Transmit Channel Active Reset Register (MAL0_TXCARR) 569

Figure 23-15. MAL Receive Channel Active Set Register (MAL0_RXCASR) 569
Figure 23-16. MAL Receive Channel Active Reset Register (MAL0_RXCARR) 570
Figure 23-17. MAL Transmit End of Buffer Interrupt Status Register (MAL0_TXEOBISR) 570
Figure 23-18. MAL Receive End of Buffer Interrupt Status Register (MAL0_RXEOBISR) 570
Figure 23-19. MAL Error Status Register (MAL0_ESR) 571
Figure 23-20. MAL Interrupt Enable Register (MAL0_IER) 572
Figure 23-21. MAL Transmit Descriptor Error Interrupt Register (MAL0_TXDEIR) 573
Figure 23-22. MAL Receive Descriptor Error Interrupt Register (MAL0_RXDEIR) 573
Figure 23-23. MAL Transmit Channel Table Pointer Register (MAL0_TXCTPxR) 573
Figure 23-24. MAL Receive Channel Table Pointer Register (MAL0_RXCTPxR) 574
Figure 23-25. MAL Receive Channel Buffer Size Register (MAL0_RCBSx) 574
Figure 24-1. EMAC to PHY Using MII 577
Figure 24-2. EMAC to PHY Using RMII 578
Figure 24-3. EMAC to PHY Using SMII 579
Figure 24-4. ZMII Function Enable Register (ZMII0_FER) 579
Figure 24-5. ZMII Speed Selection Register (ZMII0_SSR) 580
Figure 24-6. ZMII SMII Status Register (ZMII0_SMIISR) 581
Figure 25-1. EMAC in a Typical Ethernet Application 584
Figure 25-2. Internal EMAC Structure 586
Figure 25-3. EMAC Loopback Modes 588
Figure 25-4. EMACx RX Status Register(SDR0_EMACxRXST) 589
Figure 25-5. EMACx TX Status Register(SDR0_EMACxTXST) 591
Figure 25-6. EMACx Reject Count Register (SDR0_EMACxREJCNT) 592
Figure 25-7. MAL TX Descriptor Control/Status Field 594
Figure 25-8. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets) 597
Figure 25-9. MAL RX Descriptor Control/Status Field 599
Figure 25-10. Wake-Up Packet Format 601
Figure 25-11. Control Packet Format 602
Figure 25-12. Integrated Flow Control Mechanism 603
Figure 25-13. Pause Operation State Machine 604
Figure 25-14. VLAN Tagged Packet Format 605
Figure 25-15. Tag Control Information Field Structure 605
Figure 25-16. Receive Address Recognition Flowchart 608
Figure 25-17. Ethernet Address Filter Operation 609
Figure 25-18. Mode Register 0 (EMACx_MR0) 611
Figure 25-19. Mode Register 1 (EMACx_MR1) 612
Figure 25-20. Transmit Mode Register 0 (EMACx_TMR0) 613
Figure 25-21. Transmit Mode Register 1 (EMACx_TMR1) 614
Figure 25-22. Receive Mode Register (EMACx_RMR) 615
Figure 25-23. Interrupt Status Register (EMACx_ISR) 616
Figure 25-24. Interrupt Status Enable Register (EMACx_ISER) 618

Figure 25-25. Individual Address High Register (EMACx_IAHR) 620
Figure 25-26. Individual Address Low Register (EMACx_IALR) 620
Figure 25-27. VLAN TPID Register (EMACx_VTPID) 620
Figure 25-28. VLAN TCI Register (EMACx_VTCI) 621
Figure 25-29. Pause Timer Register (EMACx_PTR) 621
Figure 25-30. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4) 621
Figure 25-31. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4) 621
Figure 25-32. Last Source Address High Register (EMACx_LSAH) 622
Figure 25-33. Last Source Address Low Register (EMACx_LSAL) 622
Figure 25-34. Inter-Packet Gap Value Register (EMACx_IPGVR) 622
Figure 25-35. STA Control Register (EMACx_STACR) 623
Figure 25-36. Transmit Request Threshold Register (EMACx_TRTR) 624
Figure 25-37. Receive Low/High Water Mark Register (EMACx_RWMR) 624
Figure 25-38. Number of Octets Transmitted (EMACx_OCTX) 625
Figure 25-39. Number of Octets Received (EMACx_OCRX) 625
Figure 25-40. EMAC-MAL Communication Phases 626
Figure 26-1. UART Receiver Buffer Registers (UARTx_RBR) 633
Figure 26-2. UART Transmitter Holding Registers (UARTx_THR) 633
Figure 26-3. UART Interrupt Enable Registers (UARTx_IER) 633
Figure 26-4. UART Interrupt Identification Registers (UARTx_IIR) 634
Figure 26-5. UART FIFO Control Registers (UARTx_FCR) 635
Figure 26-6. UART Line Control Registers (UARTx_LCR) 636
Figure 26-7. UART Modem Control Registers (UARTx_MCR) 636
Figure 26-8. UART Line Status Registers (UARTx_LSR) 637
Figure 26-9. UART Modem Status Registers (UARTx_MSR) 638
Figure 26-10. UART Scratchpad Registers (UARTx_SCR) 639
Figure 26-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM) 639
Figure 26-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL) 639
Figure 26-13. UART Configuration Register 0 (SDR0_UART0) 640
Figure 26-14. UART Configuration Register 1 (SDR0_UART1) 641
Figure 26-15. UART Configuration Register 2 (SDR0_UART2) 642
Figure 26-16. UART Configuration Register 3 (SDR0_UART3) 642
Figure 27-1. SPI Functional Diagram 654
Figure 27-2. SPI Receive Data Register (SPI0_RxD) 656
Figure 27-3. SPI Transmit Data Register (SPI0_TxD) 656
Figure 27-4. SPI Control Register (SPI0_CR) 657
Figure 27-5. SPI Clock Divisor Modulus Register (SPI0_CDM) 657
Figure 27-6. SPI Status Register (SPI0_SR) 658
Figure 27-7. SPI Mode Register (SPI0_MODE) 658
Figure 28-1. 7-Bit Addressing 660
Figure 28-2. 10-Bit Addressing 660

Figure 28-3. IICx Master Data Buffer (IICx_MDBUF) 662
Figure 28-4. IICx Slave Data Buffer (IICx_SDBUF) 663
Figure 28-5. IICx Low Master Address Register (IICx_LMADR) 663
Figure 28-6. IICx High Master Address Register (IICx_HMADR) 664
Figure 28-7. IICx Control Register (IICx_CNTL) 664
Figure 28-8. IICx Mode Control Register (IICx_MDCNTL) 666
Figure 28-9. IICx Status Register (IICx_STS) 667
Figure 28-10. IICx Extended Status Register (IICx_EXTSTS) 668
Figure 28-11. IICx Low Slave Address Register (IICx_LSADR) 670
Figure 28-12. IICx High Slave Address Register (IICx_HSADR) 670
Figure 28-13. IICx Clock Divide Register (IICx_CLKDIV) 671
Figure 28-14. IICx Interrupt Mask Register (IICx_INTRMSK) 672
Figure 28-15. IICx Transfer Count Register (IICx_XFRCNT) 672
Figure 28-16. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) 673
Figure 28-17. IICx Direct Control Register (IICx_DIRECTCNTL) 674
Figure 28-18. IICx Interrupt Register (IICx_INTR) 675
Figure 29-1. GPIO Dataflow and Configuration Registers 680
Figure 29-2. GPIO Output Register (GPIOx_OR) 683
Figure 29-3. GPIO Three-State Register (GPIOx_TCR) 683
Figure 29-4. GPIO Open Drain Register (GPIOx_ODR) 685
Figure 29-5. GPIO Input Register (GPIOx_IR) 685
Figure 30-1. USB Interface Block Diagram 694
Figure 30-2. USB Controller Data Flow 695
Figure 30-3. USB Configuration Register (SDR0_USB0) 695
Figure 30-4. USB Host Controller Hardware/Software Interface 696
Figure 30-5. HcRevision Register Format 700
Figure 30-6. HcControl Register Format 701
Figure 30-7. HcCommandStatus Register Format 703
Figure 30-8. HcInterruptStatus Register Format 704
Figure 30-9. HcInterruptEnable Register Format 705
Figure 30-10. HcInterruptDisable Register Format 706
Figure 30-11. HcHCCA Register Format 707
Figure 30-12. HcPeriodCurrentED Register Format 708
Figure 30-13. HcControlHeadED Register Format 709
Figure 30-14. HcControlCurrentED Register Format 710
Figure 30-15. HcBulkHeadED Register Format 711
Figure 30-16. HcBulkCurrentED Register Format 712
Figure 30-17. HcDoneHead Register Format 713
Figure 30-18. HcFmInterval Register Format 714
Figure 30-19. HcFmRemaining Register Format 715
Figure 30-20. HcFmNumber Register Format 716

Figure 30-21. HcPeriodicStart Register Format 717
Figure 30-22. HcLSThreshold Register Format 718
Figure 30-23. HcRhDescriptorA Register Format 719
Figure 30-24. HcRhDescriptorB Register Format 720
Figure 30-25. HcRhStatus Register Format 721
Figure 30-26. HcRhPortStatus[1:2] Register Format 723
Figure 30-27. OPBMC Register Format 725
Figure 30-28. Typical USB 2.0 Device System Application Block Diagram 728
Figure 30-29. USB 2.0 Device System Block Diagram 729
Figure 30-30. Interrupts for Endpoints 0 and IN 1–3 Register (USB2D0_INTRIN) 731
Figure 30-31. USB2D0_POWER Register 731
Figure 30-32. USB2D0_FADDR Register 733
Figure 30-33. USB2D0_INTRINE Register 734
Figure 30-34. USB2D0_INTROUT Register 734
Figure 30-35. USB2D0_INTRUSBE Register 735
Figure 30-36. USB2D0_INTRUSB Register 736
Figure 30-37. USB2D0_INTROUTE Register 737
Figure 30-38. USB2D0_TESTMODE Register 738
Figure 30-39. USB2D0_INDEX Register 739
Figure 30-40. USB2D0_FRAME Register 739
Figure 30-41. USB2D0_INCSR0 Register 741
Figure 30-42. USB2D0_INCSR Register 742
Figure 30-43. USB2D0_INMAXP Register 744
Figure 30-44. USB2D0_OUTCSR Register 745
Figure 30-45. USB2D0_OUTMAXP Register 747
Figure 30-46. USB2D0_OUTCOUNT0 Register 748
Figure 30-47. USB2D0_OUTCOUNT Register 748

Tables

Table 2-1. PPC440EP PLB4 Master and Slave Assignments 64
Table 2-2. PPC440EP PLB3 Master and Slave Assignments 64
Table 2-3. Registers Controlling PLB4 Master Priority Assignments 65
Table 2-4. Registers Controlling PLB3 Master Priority Assignments 65
Table 2-5. PLB4 Arbiters 0 and 1 Registers 73
Table 2-6. Master Priority Orders 75
Table 2-7. PLB Segment Access 79
Table 2-8. PLB4 to PLB3 Bridge Registers 80
Table 2-9. PLB3 to PLB4 Bridge Registers 86
Table 2-10. PLB4 to OPB Bridge Registers 94
Table 2-11. PLB3 Arbiter 0 Registers 98
Table 2-12. PLB3 to OPB Bridge Registers 102
Table 2-13. OPBA0 Master Assignments 105
Table 2-14. OPB Arbiter Registers 106
Table 2-15. OPB to PLB3 Bridge Registers 107
Table 3-1. PLB Event Occurrence 117
Table 3-2. PLB Event Duration Measurement 119
Table 3-3. PPM DCR Addresses 119
Table 3-4. PPM Indirectly Accessed Registers 119
Table 4-1. PPC440EP System Memory Address Map 138
Table 4-2. PPC440EP Device Configuration Register Address Map 139
Table 4-3. PPC440EP Device Control Registers 139
Table 4-4. PPC440EP Memory Mapped Registers 149
Table 5-1. Data Operand Definitions 160
Table 5-2. Alignment Effects for Storage Access Instructions 160
Table 5-3. Invalid Operation Exception Categories 161
Table 5-4. Floating-Point Single Format 165
Table 5-5. Floating-Point Double Format 165
Table 5-6. Format Fields 166
Table 5-7. IEEE 754 Floating-Point Fields 166
Table 5-8. Rounding Modes 171
Table 5-9. IEEE 64-bit Execution Model 172
Table 5-10. Interpretation of the G, R, and X Bits 173
Table 5-11. Location of the Guard, Round, and Sticky Bits in the IEEE Execution Model 173
Table 5-12. Multiply-Add 64-bit Execution Model 174
Table 5-13. Location of Guard, Round, and Sticky Bits in the Multiply-Add Execution Model 175
Table 5-14. Floating-Point Load Instructions 178
Table 5-15. Floating-Point Store Instructions 179
Table 5-16. Floating-Point Move Instructions 179
Table 5-17. Floating-Point Elementary Arithmetic Instructions 180

Table 5-18. Floating-Point Multiply-Add Instructions 180
Table 5-19. Floating-Point Rounding and Conversion Instructions 181
Table 5-20. Comparison Sets 181
Table 5-21. Floating-Point Compare and Select Instructions 181
Table 5-22. Floating-Point Status and Control Register Instructions 182
Table 8-1. Reset Values of Registers and Other 440EP Core Facilities 192
Table 8-2. SDR Configuration Registers 196
Table 9-1. Bootstrappable Register Bit Fields 207
Table 9-2. Bootstrap Pins 209
Table 9-3. Bootstrap Options and Default X SDR0 Register Settings 211
Table 9-4. Serial ROM Memory Map 215
Table 9-5. System DCR Register 216
Table 9-6. IIC Bootstrap Controller Registers 216
Table 10-1. UIC0 Interrupt Assignments 224
Table 10-2. UIC1 Interrupt Assignments 225
Table 10-3. UIC Device Control Registers 226
Table 12-1. Invalid Operation Exception Categories 260
Table 12-2. MSR[FE0, FE1] Modes 262
Table 12-3. Invalid Operation Exceptions 264
Table 12-4. QNaN result 269
Table 12-5. FPSCR[FPRF] Result Flags 269
Table 12-6. Bit Encodings for a CR Field 271
Table 14-1. GPT Registers 277
Table 15-1. PPC440EP Clocking Modes 285
Table 15-2. Clock Frequencies and M multiplier 286
Table 15-3. System PLL Configuration Using PLL Local Feedback (CPR0_PLLC0[SEL] = 000) 286
Table 15-4. System PLL Configuration Using CPU Feedback (CPR0_PLLC0[SEL] = 001) 286
Table 15-6. CPR0_PLLC0[TUNE] Bit Settings 287
Table 15-5. System PLL Configuration Using PerClk Feedback (CPR0_PLLC0[SEL] = 101) 287
Table 15-7. SDR0_CP440[Nto1] Settings 288
Table 15-8. System Clock Rules 289
Table 15-9. Equations to Determine VCO, CPU, PLB Frequency 289
Table 15-10. Clock Ratio Listing With PLL Local Feedback, SysClk = 33.3MHz 290
Table 15-11. Clock Ratio Listing With CPU Feedback, SysClk = 33.3MHz 291
Table 15-12. Clock Ratio Listing With PerClk feedback, SysClk = 33.3MHz 292
Table 15-13. CPU:PLB Ratio 292
Table 15-14. Example Synchronous PCI Clock Frequencies in Asynchronous Mode 293
Table 15-15. PPC440EP Clocking Control Register Access 294
Table 15-16. PPC440EP Clocking Control Registers 294
Table 16-1. CPM Registers 301
Table 18-1. DDR SDRAM Signal Usage and State During/Following Reset 312

Table 18-2. DDR SDRAM Controller DCR Addresses 313
Table 18-3. DDR SDRAM Registers 314
Table 18-4. DCRs with Dependencies on SDRAM0_CFG0[DCEN] 315
Table 18-5. DDR SDRAM Addressing Modes 326
Table 18-6. 32-Bit SDRAM Page Size 345
Table 18-7. 64-Bit SDRAM Page Size 345
Table 18-8. 32-Bit DDR SDRAM Addressing Modes 346
Table 18-9. 64-Bit DDR SDRAM Addressing Modes 346
Table 18-10. SDRAM Memory Timing Parameters 347
Table 18-11. Precharge Command 349
Table 18-12. SDRAM Controller Reset/Self-Refresh Response 353
Table 18-12. Mode Set Command Vectors 354
Table 18-13. ECC Features 355
Table 18-14. Effect of ECC on Timing 356
Table 18-15. 32-Bit Mode ECC Code Matrix for Word 0 358
Table 18-16. 32-Bit Mode ECC Code Matrix for Word 0 (continued) 358
Table 18-17. 32-Bit Mode ECC Code Matrix for Word 1 359
Table 18-18. 32-Bit Mode ECC Code Matrix for Word 1 (continued) 359
Table 18-19. 64-Bit Mode ECC Code Matrix 360
Table 19-1. PowerPC, CoreConnect PLB, and PCI Address Bit-Naming Conventions 364
Table 19-2. PowerPC, CoreConnect PLB, and PCI Data Bus Bit-Naming Conventions 365
Table 19-3. PLB Address Map 367
Table 19-4. PCI Memory Address Map 369
Table 19-5. Transaction Mapping: PLB —> PCI 371
Table 19-6. Transaction Mapping: PCI → PLB 374
Table 19-7. Collision Resolution 378
Table 19-8. Directly Accessed MMIO Registers 379
Table 19-9. PCI Configuration Address and Data Registers 379
Table 19-10. PCI Configuration Register Offsets 380
Table 19-11. PLB Unsupported Transfer Types 404
Table 19-12. Address Map Register Values 409
Table 20-1. EBC Signal Usage and State During and After Chip and System Resets 466
Table 20-2. Effect of Driver Enable Programming on EBC Signal States 467
Table 20-3. External Master Arbitration. 478
Table 20-4. Signal States During Hold Acknowledge (HoldAck=1) 479
Table 20-5. EBC DCR Addresses 483
Table 20-6. EBC Configuration and Status Registers 484
Table 21-1. Correspondence between SDR0_CUST0 Fields and NDFC Register Fields 495
Table 21-2. ECC Code Assignment Table 497
Table 21-3. EBC Transfer Sizes 497
Table 21-4. NAND Flash Controller Memory Map 498

Table 21-5. Data Input Ordering to Read Data on EBC 500
Table 21-6. Write Data to Output Ordering 501
Table 22-1. DMA to PLB 4 Channel Assignments 509
Table 22-2. DMA to PLB4 Controller Configuration and Status Registers 511
Table 22-3. DMA to PLB 4 Transfer Priorities 519
Table 22-4. Address Alignment Requirements 520
Table 22-5. DMA to PLB 4 Registers Loaded from Scatter/Gather Descriptor 522
Table 22-6. DMA to PLB 3 Channel Assignments 525
Table 22-7. DMA to PLB3 Controller External I/Os 526
Table 22-8. DMA to PLB 3 Controller Configuration and Status Registers 529
Table 22-9. DMA to PLB 3 Transfer Priorities 536
Table 22-10. Address Alignment Requirements 537
Table 22-11. Scatter/Gather Descriptor Table 538
Table 22-12. Bit Fields in the Scatter/Gather Descriptor Table 539
Table 22-13. DMA Registers Loaded from Scatter/Gather Descriptor Table 539
Table 23-1. MAL0 Channel Assignment 547
Table 23-2. MAL Register Summary 567
Table 24-1. ZMII Bridge PHY Interface 576
Table 24-2. Register Address List 579
Table 25-1. Packet Reject Registers 588
Table 25-2. FCS/SA Enable - Possible Configurations 598
Table 25-3. FCS/Pad Enable - Possible Configurations 598
Table 25-4. FCS/VLAN Tag Enable - Possible Configurations 598
Table 25-5. In Range Length Error Behavior for Various Packet Lengths 600
Table 25-6. EMAC0 Register Summary 609
Table 25-7. EMAC1 Register Summary 610
Table 26-1. Baud Rate Settings 631
Table 26-2. UART Configuration Registers 632
Table 26-3. Interrupt Priority Level 634
Table 26-4. Divisor Latch Settings for Certain Baud Rates 640
Table 26-5. DMA to PLB3 Channel Assignments 645
Table 26-6. UART 0 Transmitter DMA Mode Register Field Settings 647
Table 26-7. UART 1 Transmitter DMA Mode Register Field Settings 648
Table 26-8. UART2 Transmitter DMA Mode Register Field Settings 648
Table 26-9. UART3 Transmitter DMA Mode Register Field Settings 649
Table 26-10. UART0 Receiver DMA Mode Register Field Settings 650
Table 26-11. UART1 Receiver DMA Mode Register Field Settings 650
Table 26-12. UART2 Receiver DMA Mode Register Field Settings 651
Table 26-13. UART3 Receiver DMA Mode Register Field Settings 651
Table 27-1. SPI Registers 655
Table 28-1. IIC Registers 661

Table 28-2. IIC Response to IICx_CNTL Field Settings 665
Table 28-3. IICx_STS[ERR, PT] Decoding 667
Table 28-4. IICx Clock Divide Programming 671
Table 29-1. GPIO Register Summary 681
Table 29-2. GPIO Output Signal Selection 684
Table 29-3. GPIO Three-State Selection 684
Table 29-4. GPIO0_ODR Control Settings 685
Table 29-5. GPIO Alternate Input Signal Selection 686
Table 29-6. Alternate 1 Configuration for GPIO00–GPIO15 687
Table 29-7. Alternate 1 Configuration for GPIO16–GPIO31 688
Table 29-8. Alternate 1 Configuration for GPIO32–GPIO47 689
Table 29-9. Alternate 1 Configuration for GPIO48 689
Table 29-10. Alternate 2 Configuration for GPIO00–GPIO05 690
Table 29-11. Alternate 2 Configuration for GPIO26–GPIO31 690
Table 29-12. Alternate 2 Configuration for GPIO34–GPIO47 691
Table 29-13. Alternate 2 Configuration for GPIO48 691
Table 29-14. Alternate 3 Configuration for GPIO34–GPIO37 691
Table 30-1. PPC440EP Register Name to OHCI Register Name Cross-Reference 698
Table 30-2. OHCI Host Controller Operation Register Summary 699
Table 30-3. HcRevision Register Field Description 700
Table 30-4. HcControl Register Field Description 701
Table 30-5. HcCommandStatus Register Field Description 703
Table 30-6. HcInterruptStatus Register Field Description 704
Table 30-7. HcInterruptEnable Register Field Description 705
Table 30-8. HcInterruptDisable Register Field Description 706
Table 30-9. HcHCCA Register Field Description 707
Table 30-10. HcPeriodCurrentED Register Field Description 708
Table 30-11. HcControlHeadED Register Field Description 709
Table 30-12. HcControlCurrentED Register Field Description 710
Table 30-13. HcBulkHeadED Register Format 711
Table 30-14. HcBulkCurrentED Register Field Description 712
Table 30-15. HcDoneHead Register Field Description 713
Table 30-16. HcFmInterval Register Field Description 714
Table 30-17. HcFmRemaining Register Field Description 715
Table 30-18. HcFmNumber Register Field Description 716
Table 30-19. HcPeriodicStart Register Field Description 717
Table 30-20. HcLSThreshold Register Field Description 718
Table 30-21. HcRhDescriptorA Register Field Description 719
Table 30-22. HcRhDescriptorB Register Field Description 720
Table 30-23. HcRhStatus Register Field Description 721
Table 30-24. HcRhPortStatus[1:2] Register Field Description 723

Table 30-25. OPBMC Register Field Description 726
Table 30-26. Common Registers 730
Table 30-27. Indexed Registers 740
Table 30-28. FIFO Address Assignment 749
Table 32-1. Instruction Categories 755
Table 32-2. Operator Precedence 758
Table 32-3. fres Operation with Special Operand Values 781
Table 32-4. frsqrte Operation with Special Operand Values 783
Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers 812

About This Book

This user's manual provides the architectural overview, programming model, and detailed information about the registers, the instruction set, and operations of the AMCC™ PowerPC™ 440EP (PPC440EP™) 32-bit RISC embedded processor.

The PPC440EP RISC embedded processor features:

- Book-E Enhanced PowerPC Architecture™
- PPC440EP CPU processor core
- Selectable processor bus ratios
- CoreConnect bus architecture, PLB performance monitor and peripheral cores
- Floating point processor core
- Double data rate (DDR) synchronous DRAM (SDRAM) controller
- Peripheral component interconnect (PCI) bridge controller
- Two direct memory access (DMA) controllers
- Memory access layer (MAL) controller
- EMAC to PHY interface (ZMII)
- Two on-chip ethernet ports (EMACs)
- General purpose timer (GPT)
- Two universal interrupt controllers (UICs)
- Four universal asynchronous receiver/transmitters (UARTs)
- Serial peripheral interface (SPI)
- NAND flash controller (NDFC)
- Four universal serial bus (USB) interfaces
- Master and slave Inter-integrated circuit (IIC) controller
- General purpose I/O (GPIO) interface
- External bus controller integrated with external bus master interface (EBC)
- JTAG support for board level testing
- Internal processor local bus (PLB) running at SDRAM interface frequency
- Support for PowerPC processor boot from PCI memory
- Extensive development tool support

Who Should Use This Book

This book is for system hardware and software developers, and for application developers who need to understand the PPC440EP. The audience should understand embedded processor design, embedded system design, operating systems, RISC processing, and design for testability.

How to Use This Book

This book describes the PPC440EP device architecture (including instructions and registers), processor core functions, system operations, internal bus functions, and external interfaces. The book is organized as follows:

- Part I Introduction
  - *Overview* on page 49
  - *On-Chip Buses* on page 63
  - *PLB Performance Monitor* on page 111
- Part II PPC440EP RISC Processor
  - *Programming Model* on page 137
  - *FPU Programming Model* on page 159
  - *Instruction and Data Caches* on page 183
  - *Memory Management* on page 185
- Part III PPC440EP System Operations
  - *Reset and Initialization* on page 189
  - *Bootstrap Controller* on page 207
  - *Universal Interrupt Controller* on page 223
  - *Interrupts and Exceptions* on page 257
  - *Floating Point Unit Interrupts and Exceptions* on page 259
  - *Timer Facilities* on page 273
  - *General Purpose Timers* on page 275
  - *Clocking* on page 283
  - *Clock and Power Management* on page 301
  - *Debug Facilities* on page 307
- Part IV PPC440EP Peripheral Functions and Interfaces
  - *DDR SDRAM Controller* on page 311
  - *Peripheral Component Interconnect (PCI) Interface* on page 363
  - *External Bus Controller* on page 465
  - *NAND Flash Controller* on page 493
  - *Direct Memory Access Controllers* on page 509
  - *Memory Access Layer* on page 545
  - *EMAC to PHY Interface Bridge* on page 575
  - *Ethernet Media Access Controllers* on page 583
  - *Serial Port Operations* on page 629
  - *Serial Peripheral Interface* on page 653
  - *IIC Bus Interface* on page 659
  - *GPIO Operations* on page 679
  - *Universal Serial Bus Interfaces* on page 693
- Part V Reference
  - *Instruction Set* on page 753
  - *Register Summary* on page 811
  - *Signal Summary* on page 829

This book contains the following appendixes:

- *Floating Point Instruction Summary* on page 831

To help readers find material in these chapters, the book contains:
- *Contents* on page 3
- *Figures* on page 23
- *Tables* on page 37
- *Index* on page 843

Conventions

The following is a list of notational conventions frequently used in this manual.

$\overline{\text{ActiveLow}}$	An overbar indicates an active-low signal.
n	A decimal number
0x*n*	A hexadecimal number
0b*n*	A binary number
=	Assignment
$\wedge$	AND logical operator
$\neg$	NOT logical operator
$\vee$	OR logical operator
$\oplus$	Exclusive-OR (XOR) logical operator
+	Twos complement addition
–	Twos complement subtraction, unary minus
$\times$	Multiplication
$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
$\|\|$	Concatenation
=, $\neq$	Equal, not equal relations
<, >	Signed comparison relations
$\overset{u}{<}$, $\overset{u}{>}$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
do	Do loop. "to" and "by" clauses specify incrementing an iteration variable; "while" and "until" clauses specify terminating conditions. Indenting indicates the scope of a loop.
leave	Leave innermost do loop or do loop specified in a leave statement.
FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
REG_b	A bit in a named register
$REG_{b:b}$	A range of bits in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register

REG[FLD]	A field in a named register
REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, . . .
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
CR_{FLD}	The field in the condition register pointed to by a field of an instruction.
$c_{0:3}$	A 4-bit object used to store condition results in compare instructions.
^{n}b	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
CEIL(x)	Least integer $\ge$ x.
EXTS(x)	The result of extending *x* on the left with sign bits.
PC	Program counter.
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies a location in main storage.
EA_b	A bit in an effective address.
$EA_{b:b}$	A range of bits in an effective address.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.

Part I. Introduction

1. Overview

The AMCC™ PowerPC™ 440EP 32-bit reduced instruction set computer (RISC) embedded processor, referred to as the PPC440EP, is a system-on-a-chip (SOC) design that integrates the PowerPC 440 CPU processor with a mix of rich peripheral controllers by implementing IBM's high speed CoreConnect™ technology.

This chapter describes the following features with cross references for detailed information included in this book:

- PPC440EP embedded processor features as seen from processor core, peripheral cores and CoreConnect™ technology viewpoint.
- PowerPC Architecture that describes features accessed at user level and supervisor level.
- PPC440EP as a 32-bit implementation of Book-E Enhanced PowerPC Architecture™ optimized for embedded applications.
- A fully integrated PowerPC 440 CPU and FPU processor implemented in advanced copper technology with advanced CPU/FPU extensions and DSP instructions.
- Standard on-chip buses that host both high performance, high bandwidth as well as lower data rate peripherals.
- Multiple PPC440EP interfaces that provide a peripheral mix for a wide variety of applications.
- Chip level programming model that provides a high degree of user control over configuration and operation of all functional units.
- Powerful debug support for a wide range of hardware and software development tools.

Please refer to the *Power PC 440EP Embedded Processor Data Sheet* for a block diagram of this processor.

1.1 PPC440EP Features

The PPC440EP provides high performance and low power consumption with CPU processor executing at sustained speeds approaching two instructions per cycle. On-chip peripherals reduce chip count and design complexity in systems and improve system throughput. The CPU combined with diverse peripheral mix through the IBM CoreConnect™ technology provides an ideal foundation for systems incorporating system-on-a-chip (SOC) designs such as in PPC440EP. This section provides a list of features that are implemented in PPC440EP.

1.1.1 PPC440EP Processor Core Features

Refer to the *PPC440 Processor User's Manual* for a description of the PPC440 processor and its features.

1.1.2 PPC440EP Floating Point Unit Features

- Single-precision and double-precision operation in hardware
- Executes PowerPC floating point instruction set
- Masked exceptions handled in hardware
- Superpipelined; single-cycle throughput for most instructions
- Superscalar; independent arithmetic and load/store execution units
- Out-of-order dispatch, execution, and completion
- Dual instruction (arithmetic and load/store) decode and issue
- Thirty-two 64-bit Floating Point Registers (FPRs)
- Extensive built-in power management for maximum performance and power efficiency

- Interfaces to the PowerPC 440 embedded processor core and CPM via the Auxiliary Processor Unit (APU) interface
- 128-bit load/store interface

1.1.3 PPC440EP Peripheral Features

- Double data rate (DDR) synchronous DRAM (SDRAM) controller
  - 1.1GB peak data rate
  - 32-bit interface with optional ECC
  - Eight-bit or wider devices
  - DDR: 64, 128, 256MB; (4 bank devices)
  - 16MB to 256MB per bank , maximum 1GB total
  - Supports discrete devices
  - Page mode accesses w/bank interleaving
  - Configurable paging (supports 8 open pages)
  - Power management
  - SSTL logic interface
- Peripheral component interconnect (PCI) bridge controller
  - 32-bit address/data bus, PCI v2.2 compatible (no 5V support)
  - PCI bus frequency up to 66 MHz (asynchronous)
  - Asynchronous clocking between PLB and PCI buses (optional)
  - Supports 2:1, 3:1, and 4:1 clock ratios from PLB to PCI
  - Power management and buffering
  - Error tracking/status and PCI arbitration function
  - PCI target-side configuration and processor access to all PCI address spaces
- Two Direct Memory Access (DMA) controllers
  - DMA2P40 attached to 128-bit processor local bus
    - Supports memory to memory, peripheral to memory, and memory to peripheral transfers
    - Four independent DMA channels supporting internal USB 2.0 device endpoints 1 and 2
    - 128-byte buffer with programmable thresholds
    - Scatter/gather capability
  - DMA2P30 attached to 64-bit processor local bus
    - Supports memory to memory, peripheral to memory, and memory to peripheral transfers
    - Four independent DMA channels supporting internal and external transfers
    - Option to use upto four external or four internal channels (UART DMAs)
    - 128-byte buffer with programmable thresholds
    - Scatter/gather capability
- Memory access layer (MAL) controller
  - No restrictions on buffer alignment
  - Aligned bus accesses to enable burst operation with external memories

- Configurable receive buffer size (configurable per channel)
- No minimum transmit buffer size
- Maximum buffer sizes of 4095 bytes (TX) and 4080 bytes (RX)
- Up to 256 descriptors in the buffer descriptor table per channel
- Configures EMAC according to commands specified in the descriptor status/control field
- Updates the descriptor status/control field at the end of packet transfer according to the status received from EMAC
- Buffer-based interrupt capabilities for each channel
- Concurrent operation of RX and TX channels
- Configuration using Device Control Registers (DCRs)
- Programmable PLB arbitration priority
- PLB/OPB error detection

- Z media independent interface (ZMII)
  - Support for one MII PHY
  - Support for two RMII PHYs
    - 10Mbps and 100Mbps data rates
    - 50MHz reference clock sourced from EMAC, or an external source, to the PHY
    - Independent 2-bit transmit and receive data paths
  - Support for two SMII PHYs
    - 10Mbps and 100Mbps data rates
    - Half and full duplex operation
    - System clock sharing for multiple EMACs and PHYs
    - Independent 1-bit transmit and receive data paths
  - Programmable selection of any EMAC to drive MDI
  - Programmable selection of any EMACto drive MII
  - Programmable selection of either or both EMACs to drive RMII or SMII
- Two on-chip ethernet ports (EMAC0 and EMAC1)
  - Multi-speed capability for full or half duplex at 10/100 Mbps
  - 2KB transmit fifo, 4KB receive fifo
  - One MII or two RMII/SMII ports
  - Packet reject support
- General purpose timer (GPT)
  - Provides a separate time base counter and system timers in addition to those defined in the processor core
  - 32-bit Time Base Counter driven by the OPB bus Clock
  - Five 32-bit compare timers
- Two universal interrupt controllers (UICs)
  - Supports programmable interrupt handling from a variety of sources
  - Support for asynchronous level- or edge-sensitive interrupt types

  - Programmable polarity for all interrupt types
  - Support for 10 external and 64 internal interrupts
- Four universal asynchronous receiver/transmitters (UARTs)
  - Compatible with the NS 16750
  - 16-byte send FIFO, 16-byte receive FIFO
  - Full duplex operation
  - Programmable baud rate generator
  - Supports 5- to 8-bit word size, 1 or 2 stop bits, even, odd, or no parity
  - 1x 8-pin, or 2x 4-pin, or 1x 4-pin and 2x 2-pin, or 4x2-pin
- Serial peripheral interface (SPI)
  - Full-duplex, synchronous, character-oriented (byte) port
  - Allows exchange of data with other serial devices.
  - 4-wire serial port interface (receive, transmit, clock, and slave select)
  - Each serial transaction transmits and receives one byte synchronous to the port clock provided by the master
  - Programmable clock rate divider and clock inversion
  - Up to 66 MHz OPB clock frequency
- Two inter-integrated circuit (IIC) controllers
  - Complies with Phillips I2C specifications
  - IIC0 supports an integrated bootstrap controller
  - Two wire, bi-directional, open-drain, low-speed serial interface
  - 100-kHz and 400-kHz operation
  - 8-bit data transfers
  - 7-bit and 10-bit addressing
- General purpose I/O (GPIO) interface
  - Allows flexible control of up to 64 multiplexed I/Os with user-defined functions
  - Direct control of all functions from registers programmed via memory-mapped addresses
  - Outputs can be programmed to emulate an open drain driver
- External bus controller (EBC)
  - Provides direct attachment for most SRAM/Flash type memory and peripheral devices
  - Minimizes the amount of external glue logic needed to communicate with memory and peripheral devices
  - Supports device-paced transfers with optional bus-timeout
  - Support for 8-bit and 16-bit masters
  - Separate 30-bit address bus
- NAND Flash Controller
  - Direct interfacing to discrete NAND Flash devices (up to 4 devices) or SmartMedia Card socket
  - ECC generation: Hamming code, single bit correction, double bit detection (SEC/DED)
  - x8 wide command write, x8 wide address write, x8 and x16 bit wide data read/write
  - Automatically generates RE# and WE# strobes with configurable Strobe Pulse Width parameter

- Pin sharing interface to multiplex external data bus with external bus controller.
- Interrupt on device becoming Ready (after long page write or block erase operations).
- On-chip ROM controller: Alternate, extremely low latency boot source
- Interfaces to external bus controller to allow sharing of external I/O for data and chip select.
- Operates from a single external bus controller peripheral clock.
- Clock gating for low power applications

- Universal serial bus (USB) interfaces
  - USB 2.0 Device (UTMI), USB 1.1 Host (on-chip PHY), and USB 1.1 Device (on-chip PHY)"
  - Provides a common solution for interconnecting various personal computer peripherals with a single host computing device.
  - Half duplexed, packetized, single mastered, tiered tree structure, supporting up to 127 unique devices on the tree.
  - Device ports support 6 independent endpoints (3 IN and 3 OUT), each 1024 bytes
  - 1024Byte FIFO (double buffering of 512Byte Packets)

1.1.4 PPC440EP CoreConnect Features

- Dual processor local buses: 128-bit and 64-bit implementations
  - Fully synchronous high performance PLB4 133 MHz 64-bit address bus and 128-bit data bus for CPU and DDR SDRAM.
  - Fully synchronous 133MHz 64-bit data bus (PLB3) for PCI, EBC, and other high-performance cores
  - Provides a standard interface between the processor core and the on-chip peripheral cores
  - Decoupled address and data buses support split-bus transaction capability for improved bandwidth
  - Bus arbitration-locking mechanism allows for master-driven atomic operations
  - Byte-enable capability allows for unaligned transfers and odd-byte transfers
  - Overlapping of read and write transfers allows two data transfers per clock cycle for maximum bus utilization as well as 4.3 GB/s at 133 MHz
- Dual processor local bus arbiters: 128-bit and 64-bit implementations
  - Features bus arbitration control unit, a watchdog timer, address path, write data path, and read data path units
  - Programmable read and write address pipelining
  - Programmable high bus utilization feature
  - Fixed and rotating priority schemes via strapping or programming
  - High frequency 3 cycle acknowledge timing protocol
- Dual 32-bit on-chip peripheral buses (OPB)
  - Fully synchronous 66MHz 32-bit address bus and 32-bit data bus for lower-speed peripherals
  - Dynamic bus sizing; byte, halfword, and fullword transfers
  - Sequential address (Burst) protocol support
- Dual 32-bit on-chip peripheral bus arbiters
  - Internal 32-bit address bus and 32-bit data bus core for on-chip peripheral integration
  - Dynamic priority reordering mode, implementing a true least recently used (LRU) algorithm

    - Bus parking modes for reduced access latency
- 32-bit device control register (DCR) bus for device initialization
    - A simple but flexible interface with distributed multiplexer architecture
    - 10-bit address bus and 32-bit data bus
    - 2-cycle minimum read or write transfers extendable by slave or master
    - Handshake supports clocked asynchronous transfers
    - Slaves may be clocked either faster or slower than the master
- Dual PLB to PLB bridges enabling data between 128-bit and 64-bit PLB buses in both directions
    - External programmable address space via address decode pin for the slave port.
    - Supports word, doubleword and quadword reads and writes, including both fixed-length bursts, variable length bursts, and 4 and 8-word line reads/writes.
    - Three deep address FIFO structure for reads and writes.
    - Address pipelining for read and write transfers on slave side PLB.
    - Parity checking and interrupt forwarding feature
    - 133MHz PLB clock frequency, 1:1 operation for PLB4:PLB3.
- 64-bit PLB to OPB bridge
    - Enables transfers of data between the PLB and OPB under the direction of PLB master devices
    - External programmable address space via address decode pin
    - Supports word, double word and quad word burst reads and writes, including fixed-length bursts
    - Supports pipelining for read and write transfers
    - Supports programmable PLB rearbitrate feature
    - Supports PLB bus-speeds at 2x, 3x, 4x, or 5x the frequency of the OPB
- 64-bit OPB to PLB bridge
    - Enables transfers of data between the OPB and PLB under the direction of OPB master devices
    - PLB master interface supports quadword reads, and quadword, doubleword, word, halfword, and byte writes
    - Ability to be mapped to any OPB address space
    - Single-cycle response to OPB reads and writes
    - Supports PLB bus-speeds at 2x the frequency of the OPB
- 128-bit PLB performance monitor
    - Hardware for counting certain events associated with PLB transactions
    - Registers to allow selection of master, slave, and generic events to be counted.
    - Power Management Support (Class II).

1.2 PPC440EP Floating Point Unit (FPU)

The PowerPC 440 Floating-Point Unit (PPC440 FPU) is a high-performance core that interfaces to the PowerPC 440 processor core through the Auxiliary Processor Interface (API). The PPC440 FPU incorporates a five-stage arithmetic pipeline working in parallel with a four-stage load store pipeline. The pipelines enable two instructions (one load/store and one arithmetic) to be issued during each cycle. Floating-point instructions execute with three- to five-cycle latency and one-cycle throughput, except for operations on denormalized operands and division.

See *FPU Programming Model* on page 159.

1.3 PPC440EP Internal Buses

The PPC440EP includes 5 buses, PLB4, PLB3, OPB0 for USB 2.0, OPB1 for normal use, and DCR.

The PLB is a high performance bus used to access memory through bus interface units. Lower performance peripherals (such as serial ports, general purpose I/O interface and inter integrated circuit controllers) are attached to the OPB. A bridge between the PLB and OPB enables data transfers between PLB masters and OPB slaves.

PPC440EP implements two PLB and OPB buses: 128-bit processor local bus (referred to in this document as PLB4) and 64-bit processor local bus (referred to as PLB3). Transfer of data between these two buses is enabled via the PLB to PLB bridges. One the OPB buses interfaces with USB 2.0 and DMA, and the other interfaces with rest of the peripherals.

The DCR bus is used primarily to access status and control registers of the various PLB and OPB masters and slaves. The DCR bus offloads status and control read and write transfers from the PLB.

The PLB performance monitor provides hardware for counting certain events associated with the processor local bus transactions. The performance monitor consists of a set of counters whose contents may be read by software and used to analyze and enhance PLB performance, or used as a software debug mechanism.

See *On-Chip Buses* on page 63.

1.4 PPC440EP Peripheral Functions and Interfaces

A brief description of various PPC440EP interfaces (including high performance and low performance peripherals) used in this system-on-a-chip follows with a cross reference to more detailed information available in this book.

1.4.1 Reset Interface

Resetting the PPC440EP is accomplished by asserting of reset inputs into the PPC440EP processor core, which is not reset using a scan-flush mechanism. Attempts to reset the PPC440EP core using scan-flush result in indeterminate logic states in the PPC440EP processor core. However, a scan-flush mechanism can *initialize* the PPC440EP. Some customers may wish to *initialize* the PPC440EP out of its undefined state to remove Xs in simulation (a practice *not* endorsed by the PPC440EP design group, because this can mask code and hardware initialization problems).A customer implementing a scan-flush *initialization* must still properly reset the PPC440EP by the described methods after the scan-flush initialization.

See *Reset and Initialization* on page 189.

1.4.2 JTAG Interface

The PPC440EP JTAG port is enhanced to support the attachment of a debug tool such as the RISCWatch product. Through the JTAG test access port, and using the debug facilities designed into the PPC440EP, a debug workstation can single-step the processor and interrogate internal processor state to facilitate hardware and software debugging. The enhancements comply with the IEEE 1149.1 specification for vendor-specific extensions, and are therefore compatible with standard JTAG hardware for boundary-scan system testing.

1.4.3 Bootstrap Controller

In preparation for booting, the PPC440EP initializes several settings affecting system clocking, booting and other system features during system reset. The boot settings or bootstraps used in this process are provided by either the IIC bootstrap controller or by one of six hardwired bootstraps. The hardwired bootstraps configure the PPC440EP to operate between 333 MHz and 533MHz and for booting from either an EBC or PCI attached ROM. Using the IIC bootstrap controller all bootstraps are configurable. During system reset, the IIC bootstrap controller configures the PPC440EP for booting using data read from an IIC serial ROM.

See *Bootstrap Controller* on page 207.

1.4.4 Clock and Power Management

The PPC440EP provides a clock and power management (CPM) controller that reduces power dissipation by stopping clocks in unused or dormant functional units. Use of the CPM controller requires careful programming and special consideration to avoid compromising system and functional unit integrity.

See *Clock and Power Management* on page 301.

1.4.5 DDR_SDRAM Memory Controller

The integrated memory controller supports Double Data Rate (DDR) SDRAMs, consists of a PLB slave interface and a DCR interface, and provides 32-bit interface to SDRAM memory with optional Error Checking and Correction (ECC).

The controller supports page mode operation with bank interleaving and maintains up to four open pages. The controller supports up to four 256MB logical banks in limited configurations, providing memory up to 1 GB. Global memory timings, address and bank sizes, and memory addressing modes are programmable. System power can be reduced by placing the DDR_SDRAM controller in sleep and/or self-refresh mode.

See *DDR SDRAM Controller* on page 311.

1.4.6 PCI Bridge Controller

The peripheral component interconnect (PCI) interface and bridge (referred to as PCI bridge) provides a means for connecting PCI-compatible devices to the on-chip bus architecture of the PPC440EP chip. The 32-bit PCI bridge is compatible with he PCI Specification, Version 2.2 (it does not support 5V operation). The PCI bridge is bidirectional in that it allows PPC440EP PLB masters to access PCI targets off-chip. It also allows PCI masters to access PLB slave devices such as the SDRAM controller. The PCI bridge contains an arbiter which can optionally be used for host applications.

The PCI bridge can be used as the host bridge and is configurable by an external PCI agent, allowing it to be used in target adapter applications. The PCI bridge contains address mapping register sets to provide address mapping for both transaction directions.

See *Peripheral Component Interconnect (PCI) Interface* on page 363.

1.4.7 External Bus Controller

The 16-bit external bus controller (EBC) provides direct attachment for most SRAM and Flash memory and peripheral devices. The interface minimizes the amount of external glue logic needed to communicate with memory and peripheral devices, reducing embedded system device count, circuit board area, and cost.

See *External Bus Controller* on page 465.

1.4.8 Nand Flash Controller

The Nand flash controller (NDFC) provides a simple interface between the external bus controller (EBC) and a variety of Nand flash-based storage devices. It allows easy communication with up to 4 separate external Nand flash devices. It provides both direct command, address, and data access to the external device as well as providing a memory mapped 'linear' region that will generate data accesses. The linear region can be used as the source or destination of a memcpy() command or DMA memory-to-memory transfer function to simplify and speed the software device driver.

NDFC provides hardware support for the Hamming code ECC algorithm that provides single-bit error correction and double-bit error detection (per 256 bytes of data). It also supports a mechanism to allow the chip to boot directly from code stored in the external Nand flash device when the low-level boot loader follows a specific algorithm. Power consumption within the NDFC is reduced by gating the clock to all latches internally. Minimum power is achieved when no Nand flash requests or transfers are in progress and the EBC bus is idle.

See *NAND Flash Controller* on page 493.

1.4.9 Direct Memory Access Controller

PPC440EP implements two direct memory access controllers: one attached to 128-bit processor local bus and the other attached to 64-bit processor local bus. The former provides direct support to DDR for Endpoints 1 and 2 of the USB 2.0 Device interface."

The Direct Memory Access (DMA) controller is a Processor Local Bus (PLB) and On-chip Peripheral Bus (OPB) master which supports the autonomous transfer of data between memory and peripherals and from memory-to-memory. The controller provides four DMA channels, each of which has an independent set of configuration registers. Each channel has its own control, count and control, source address, destination address, and scatter/gather address registers. Once these registers are programmed by the PPC440EP processor core, the DMA controller performs the requested data transfer without the need for processor intervention. All four channels are accessible within the PPC440EP, as well as externally for peripherals on the External Bus Controller.

The four DMA channels also support scatter/gather transfers. During a scatter/gather transfer the configuration registers for a particular DMA channel are automatically loaded from a data structure in memory instead of being individually programmed. Since the scatter/gather address register is updated in this process, the channel can optionally reconfigure itself for another transfer when the current one completes.

See *Direct Memory Access Controllers* on page 509.

1.4.10 Memory Access Layer

The Memory Access Layer (MAL) is a hardware core that manages data transfers between packet-oriented communications cores, also known as COMMACs (communications media access controllers), and memory. To communicate with software device drivers, MAL utilizes a buffer descriptor ring structure in memory. A software device driver uses the buffer descriptor structure to inform MAL about buffer locations and packet or buffer status. MAL uses the buffer descriptors to convey packet transfer status from the COMMAC core back to the software device driver. Each MAL channel requires its own buffer descriptor table ring structure in memory.

The MAL manages the transfer of packets between the two Ethernet Media Access Controllers (EMAC0 and EMAC1) and the memory attached to the PPC440EP (DDR SDRAM). The primary function of MAL is to move packets directly between memory and a COMMAC core with minimal involvement of the processor core.

See *Section 23 Memory Access Layer* on page 545.

1.4.11 EMAC to PHY Bridge

The PPC440EP provides an EMAC to PHY Interface Bridge (ZMII) that connects the ethernet media access controllers (EMACs) to standard physical devices (PHYs). Media independent interface (MII) and management data interface (MDI) signals to and from an EMAC are passed to the ZMII bridge. The ZMII bridge passes MII and MDI signals through to the PHY, or formats the MII signals to support the reduced media independent interface (RMII) or serial media independent interface (SMII).

Depending on the ZMII configuration, ZMII provides one of the following off-chip interfaces: one MII, two RMII, or two SMII. The interfaces are mutually exclusive. Interface selection is controlled by the Function Enable Register (ZMII0_FER). RMII and SMII significantly reduce the external signal count required to support multiple EMAC ports. While the MII requires 16 additional pins for a single interface, RMII requires only seven additional pins (six data and control and a common clock) and SMII requires only four additional pins (two data, sync, and a clock). Configuring the ZMII bridge for SMII or RMII allows the use of both EMACs using a reduced number of external pins.

See *Section 24 EMAC to PHY Interface Bridge* on page 575.

1.4.12 Ethernet Media Access Controller

The PPC440EP provides two Ethernet media access controllers (EMACs) that are generic implementations of the Ethernet Media Access Control (MAC) protocol complying with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. EMAC supports both half-duplex (CSMA/CD) and full-duplex operation for 10-Mbps and 100-Mbps operations.

See *Section 25 Ethernet Media Access Controllers* on page 583.

1.4.13 General Purpose Timers

The General Purpose Timer (GPT) is a system timer with seven maskable compare registers and a 32-bit time base counter. Each compare register has a corresponding GPT interrupt to the UIC. GPT interrupts can be generated for a specific count by a match between the contents of a compare register and the time base counter. GPT interrupts may also be generated on a specific interval by masking individual bits of a compare register.

See *General Purpose Timers* on page 275.

1.4.14 Universal Interrupt Controllers

The PPC440EP contains two universal interrupt controllers (UIC0, and UIC1) that provide all necessary control, status, and communication between the various internal and external interrupt sources and the processor core.

The UICs support 64 internal interrupts and 10 external interrupts. Status reporting (using the UIC Status Register [UIC*x*_SR]) is provided to ensure that systems software can determine the current and interrupting state of the system and respond appropriately. Software can generate interrupts to simplify software development and for diagnostics.

See *Universal Interrupt Controller* on page 223.

1.4.15 Serial Port Operations

The PPC440EP contains two universal asynchronous receiver/transmitters (UARTs) which provide two full-duplex serial interfaces to support communications with serial peripheral devices. Each UART is compatible with the National Semiconductor (NS) 16750 chip, and includes a 16-byte send and a 16-byte receive FIFO.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions, such as parity, overrun, framing, and break interrupt.

See *Serial Port Operations* on page 629.

1.4.16 Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a full-duplex, synchronous, character-oriented (byte) port that enables the exchange of data with other serial devices. The SPI is master on the serial port supporting a 4-wire interface (receive, transmit, clock, and slave select). The SPI is also a slave device to the on-chip peripheral bus (OPB) for communication to the processor.

See *Serial Peripheral Interface* on page 653.

1.4.17 Inter-Integrated Circuit Bus

The PPC440EP provides two inter-integrated circuit (IIC) bus interfaces (IIC0 and IIC1). Each IIC bus has a two wire, bi-directional, open-drain, low-speed serial interface. The serial clock (IIC0SClk and IIC1SClk) and serial data (IIC0SDA and IIC1SDA) lines are bidirectional, to support multiple bus masters and to mix high- and low-speed devices on the same bus.

See *IIC Bus Interface* on page 659.

1.4.18 GPIO Operations

The GPIO Controller is an OPB macro that controls up to 64 bidirectional module I/O pins with user-programmable functions. To reduce the quantity of module I/O on the PPC440EP package there are no dedicated general purpose I/Os. Each of the GPIOs has been assigned a function pin such as an interrupt (refer to the PPC440EP Datasheet for pin assignments).

See *GPIO Operations* on page 679.

1.4.19 Universal Serial Bus Interface

PPC440EP implements USB 1.1 Host, USB 1.1 Device, USB 2.0, and USB 2.0 Device. The universal serial bus interface provides a common solution for interconnecting various consumer peripherals with a single host computing device. The USB consists of half duplexed, packetized, single mastered, tiered tree structure, supporting up to 127 unique devices on the tree. The master device on the USB tree is known as the USB host controller.

See *Universal Serial Bus Interfaces* on page 693.

1.5 PPC440EP Supported Configurability

The PowerPC instruction set and register set implemented in PPC440EP provide a high degree of user control over configuration and operation of the PPC440EP functional units.

1.5.1 PPC440EP Addressing Modes

As a 32-bit implementation of the Book-E Enhanced PowerPC Architecture, the PPC440EP implements a uniform 32-bit effective address (EA) space. Effective addresses are expanded into virtual addresses and then translated to 36-bit (64GB) real addresses by the memory management unit (see *Memory Management* on page 185 for more information on the translation process). The organization of the real address space into a physical address space is described in *Programming Model* on page 137.

The PPC440EP generates an effective address whenever it executes a storage access, branch, cache management, or translation lookaside buffer (TLB) management instruction, or when it fetches the next sequential instruction.

Bytes in storage are numbered consecutively starting with 0. Each number is the address of the corresponding byte. Data storage operands accessed by the integer load/store instructions may be bytes, halfwords, words, or—for load/store multiple and string instructions—a sequence of words or bytes, respectively. Data storage operands accessed by auxiliary processor unit (APU) load/store instructions can be bytes, halfwords, words, doublewords, or quadwords. The address of a storage operand is the address of its first byte (that is, of its lowest-numbered byte). Byte ordering can be either big-endian or little-endian, as controlled by the endian storage attribute (see *Byte Ordering* in the *PPC440 Processor User's Manual*; also see *Endian (E)* in the *PPC440 Processor User's Manual* for more information on the endian storage attribute).

The PPC440EP supports the following addressing modes, which enable efficient retrieval and storage of data in memory:

- Base plus displacement addressing
- Indexed addressing
- Base plus displacement addressing and indexed addressing, with update

In the base plus displacement addressing mode, an effective address (EA) is formed by adding a displacement to a base address contained in a GPR (or to an implied base of 0). The displacement is an immediate field in an instruction.

In the indexed addressing mode, the EA is formed by adding an index contained in a GPR to a base address contained in a GPR (or to an implied base of 0).

The base plus displacement and the indexed addressing modes also have a "with update" mode. In "with update" mode, the effective address calculated for the current operation is saved in the base GPR, and can be used as the base in the next operation. The "with update" mode relieves the processor from repeatedly loading a GPR with an address for each piece of data, regardless of the proximity of the data in memory.

1.5.2 PPC440EP Register Set

The PPC440EP registers can be grouped into basic categories based on function and access mode: general purpose registers (GPRs), special purpose registers (SPRs), time base registers (TBRs), machine state register (MSR), condition register (CR), device control registers (DCRs), and memory-mapped I/O (MMIO) registers.

1.5.2.1 General Purpose Registers

The PPC440EP provides general purpose registers (GPRs) each containing 32 bits. Data from the data cache or memory can be loaded into GPRs using integer load instructions; the contents of GPRs can be stored to the data cache or memory using integer store instructions. Most of the integer instructions reference GPRs. The GPRs are also used as targets and sources for most of the instructions which read and write the other register types.

1.5.2.2 Special Purpose Registers

Special Purpose Registers (SPRs) are directly accessed using the **mtspr** and **mfspr** instructions. In addition, certain SPRs may be updated as a side-effect of the execution of various instructions. For example, the Integer Exception Register (XER) is an SPR which is updated with arithmetic status (such as carry and overflow) upon execution of certain forms of integer arithmetic instructions.

SPRs control the use of the debug facilities, timers, interrupts, memory management, caches, and other architected processor resources. Refer to the *PPC440 Processor User's Manual* for dmore detail.

1.5.2.3 Time Base Registers

The PPC440EP provides a 64-bit time base as described in *Timer Facilities* on page 273. The time base is implemented as two 32-bit time base registers (TBRs). The low-order 32 bits of the time base are read from the TBL and high-order 32 bits are read from the TBU. User-mode access to the TBRs is read-only and there is no explicitly privileged read access to the time base.

The **mftb** instruction reads from TBL and TBU. (Writing the time base is accomplished by moving the contents of a GPR to a pair of SPRs, which are also called TBL and TBU, using the **mtspr** instruction.

1.5.2.4 Machine State Register

The Machine State Register (MSR) is a register of its own unique type that controls important chip functions, such as the enabling or disabling of various interrupt types.

The MSR can be written from a GPR using the mtmsr instruction. The contents of the MSR can be read into a GPR using the mfmsr instruction. The MSR[EE] bit (external interrupt enable) can be set or cleared atomically using the wrtee or wrteei instructions. The MSR contents are also automatically saved, altered, and restored by the interrupt-handling mechanism. Refer to the *PPC440 Processor User's Manual* for more detailed information on the MSR and the function of each of its bits.

1.5.2.5 Condition Register

The Condition Register (CR) is a 32-bit register of its own unique type and is divided up into eight, independent 4-bit fields (CR0–CR7). The CR may be used to record certain conditional results of various arithmetic and logical operations. Subsequently, conditional branch instructions may designate a bit of the CR as one of the branch conditions. Instructions are also provided for performing logical bit operations and for moving fields within the CR.

1.5.2.6 Device Control Registers

Device Control Registers (DCRs) are on-chip registers that control various PPC440EP system functions, such as the operation of PPC440EP buses, peripherals, and certain PPC440EP processor core behaviors. The DCR access instructions are mtdcr (move to device control register) and mfdcr (move from device control register), which move data between GPRs and the DCRs. See *Device Control Registers* on page 139.

1.5.2.7 Memory-Mapped I/O Registers

The memory-mapped I/O (MMIO) registers are associated with PPC440EP peripherals and are accessed using load and store instructions. MMIO registers are mapped into the system memory and are used to control, configure, and hold status for various PPC440EP functional units. See *Memory Mapped Registers* on page 149

1.6 PPC440EP Signals

See *Signal Summary* on page 829.

1.7 Development Tool Support

The PPC440EP provides powerful debug support for a wide range of hardware and software development tools.

The EPOS Open real-time operating system debugger is an example of an operating system-aware debugger, implemented using software traps.

RISCWatch is an example of a development tool that uses the external debug mode, debug events, and the JTAG port to support hardware and software development and debugging.

The RISCTrace™ feature of RISCWatch is an example of a development tool that uses the real-time trace capability of the PPC440EP.

The PPC440EP evaluation board kit provides hardware and software tools. Hardware tool includes the PowerPC 440GX Evaluation Board, power supply, line cord, and board interface cables. The software tool includes the PowerPC Initialization Boot Software (PIBS) resident in the flash memory on the board, PIBS source code, the Embedded PowerPC Operating System (EPOS), sample application programs, application development libraries and tools, a compiler/linker/assembler and associated binary utilities, and RISCWatch, a source-level debugger that runs on the host system.

2. On-Chip Buses

The on-chip bus structure, which consists of the processor local bus (PLB), on-chip peripheral bus (OPB), and device control register (DCR) bus, provides a connection among all the masters and slaves on the chip buses. PPC440EP implements two processor local buses (128-bit PLB4 and 64-bit PLB3) and two identical on-chip peripheral buses. The block diagram in the *Power PC 440EP Embedded Processor Data Sheet* illustrates the on-chip bus structure of the PPC440EP.

PLB4 and PLB3 are high performance buses used to access memory through bus interface units.Transfer of data between these two buses is enabled via PLB4 to PLB3 and PLB3 to PLB4 bridges. The PLB master and slave assignments for the PPC440EP are listed in *PLB Master and Slave Assignments* on page 64. While some high performance peripherals such as PLB4 to PLB3 bridge, DDR SDRAM and DMA controller are attached to PLB4, others such as PCI bridge, media access layer, PLB3 to PLB4 bridge and another DMA controller are attached to PLB3. See *Processor Local Bus* on page 63.

Lower performance peripherals (such as serial ports, ethernet controller, external bus controller, general purpose timer, USB interface, general purpose I/O interface and IIC controllers) are attached to the OPB. A bridge between the PLB and OPB enables data transfers between PLB masters and OPB slaves. See *On-Chip Peripheral Bus* on page 105.

The DCR bus is used primarily to access status and control registers of the various PLB and OPB masters and slaves. The DCR bus off-loads status and control read and write transfers from the PLB. PPC440EP also contains peripherals such as serial port, ethernet controllers and others that use memory mapped registers which are accessed using load/store instructions. See *Device Control Register (DCR) Bus* on page 109.

Note: The term *processor local bus* or PLB in this chapter is used to represent both PLB4 and PLB3 bus unless otherwise mentioned.

2.1 Processor Local Bus

The processor local bus is a high-performance on-chip bus. The PLB supports read and write data transfers between master and slave devices equipped with a PLB interface and connected through PLB signals.

Each PLB master is attached to the PLB through separate address, read data and write data buses, and transfer qualifier signals. PLB slaves are attached to the PLB through shared, but decoupled, address, read data and write data buses, and transfer control and status signals for each data bus.

Access to the PLB is granted through a central arbitration mechanism that enables masters to compete for bus ownership. This arbitration mechanism provides for fixed and fair priority schemes.

Timing for all PLB signals is provided by a clock source that is shared by all PLB masters and slaves.

2.1.1 PLB Features

- Overlapping of read and write transfers allows two concurrent data transfers for maximum bus utilization
- Decoupled address and data buses support split-bus transaction capability for improved bandwidth
- Address pipelining reduces overall bus latency by allowing the latency associated with a new request to be overlapped with an ongoing data transfer in the same direction
- Late master request abort capability reduces latency associated with aborted requests
- Four levels of request priority for each master allow PLB implementations with various arbitration schemes
- Byte-enable capability allows for unaligned transfers and odd-byte transfers.

- Support for fixed length burst transfers
- Guarded and unguarded memory transfers allow a slave device to enable or disable the prefetching of instructions or data
- DMA buffered, peripheral to memory, memory to peripheral, and DMA memory to memory operations are supported

2.1.2 PLB Master and Slave Assignments

Table 2-1 and *Table 2-2* list the PLB4 and the PLB3 masters and slaves provided in the PPC440EP.

Table 2-1. PPC440EP PLB4 Master and Slave Assignments

PLB Agent	PLB Masters and Slaves	Master/Slave No.
Processor core instruction cache unit (ICU)	Master	0
Processor core data cache read unit (DCU)	Master	1
Processor core data cache write unit (DCU)	Master	2
DMA2P40 controller	Master	3
PLB3 to PLB4 bridge	Master	4
Double date rate (DDR) SDRAM controller	Slave	Slave segment 0, 0
PLB4 to OPB bridge	Slave	Slave segment 0, 1
PLB4 to PLB3 bridge	Slave	Slave segment 1, 0

Table 2-2. PPC440EP PLB3 Master and Slave Assignments

PLB Agent	PLB Masters and Slaves	Master/Slave No.
PLB4 to PLB3 bridge	Master	0
Media access layer (MAL)	Master	1
PCI bridge	Master	2
DMA2P30 controller	Master	3
OPB to PLB bridge	Master	4
External bus controller	Master	5
PLB3 to OPB bridge	Slave	0
External bus controller	Slave	1
PCI bridge	Slave	2
PLB3 to PLB4 bridge	Slave	3

2.1.3 PLB Master Priority Assignment

Each PLB master can be programmed to use one of four priority levels during PLB transfers, enabling the system designer to tune PLB transfer priorities to the requirements of a particular application. For example, if an application always requires DMA PCI to SDRAM transfers to have the lowest latency, the DMA PCI master can be programmed to the highest PLB master priority. This causes the PLB arbiter to grant DMA PCI access requests before granting the access requests of any other master.

A register associated with each master controls the priority of that master. *Table 2-3* and *Table 2-4* list the PLB4 and PLB3 masters and the register fields controlling the priority of the masters. Priorities range from 0b00 (lowest) to 0b11 (highest).

Table 2-3. Registers Controlling PLB4 Master Priority Assignments

Master ID	Description	Register Field	Comments
0	Processor core instruction cache unit (ICU)	SDR0_AMP0[AICURP] SDR0_AMP0[ICURP] SDR0_CP440[IRF] SDR0_CP440[IRT] SDR0_CP440[IRS]	Alternate ICU Read Priority ICU Read Priority IcuRdFetch priority setting IcuRdTouch priority setting IcuRdSpec priority setting
1	Processor core data cache read unit (DCU)	SDR0_AMP0[ADCURP] SDR0_AMP0[DCURP] SDR0_CP440[DRU] SDR0_CP440[DRT] SDR0_CP440[DRNC] SDR0_CP440[DRLC]	Alternate DCU Read Priority DCU Read Priority DcuRdUrgent priority setting DcuRdTouch priority setting DcuRdNonCache priority DcuRdLdCache priority setting
2	Processor core data cache write unit (DCU)	SDR0_AMP0[ADCUWP] SDR0_AMP0[DCUWP] SDR0_CP440[DWF] SDR0_CP440[DWS] SDR0_CP440[DWU]	Alternate DCU Write Priority DCU Write Priority DcuWrFlush priority setting DcuWrStore priority setting DcuWrUrgent priority setting
3	DMA2P40 controller	DMA2P40_CR0,1,2,3[CP] SDR0_AMP0[ADMAP4P] SDR0_AMP0[DMAP4P]	 Alternate DMA to PLB4 Priority DMA to PLB4 Priority
4	PLB3 to PLB4 bridge	P3P4BI0[PRI] SDR0_AMP0[AP3P4P] SDR0_AMP0[P3P4P]	 Alternate PLB3 to PLB4 Priority PLB3 to PLB4 Priority
5:7	Reserved		

Table 2-4. Registers Controlling PLB3 Master Priority Assignments

Master ID	Description	Register Field	Comments
0	PLB4 to PLB3 bridge	P4P3BO0_CFG[PRI] SDR0_AMP1[AP4P3P] SDR0_AMP1[P4P3P]	 Alternate PLB4 to PLB3 bridge priority PLB4 to PLB3 bridge priority
1	Media access layer (MAL)	MAL0_CFG[PLBP] SDR0_AMP1[AMALP] SDR0_AMP1[MALP]	
2	PCI bridge	SDR0_AMP1[APCIP] SDR0_AMP1[PCIP]	
3	DMA2P30 controller	DMA2P30_CR0[CP] DMA2P30_CR1[CP] DMA2P30_CR2[CP] DMA2P30_CR3[CP] SDR0_AMP1[ADMA1P] SDR0_AMP1[DMA1P]	Unique priorities can be assigned to each DMA channel
4	OPB to PLB3 bridge	OPB0_BCTRL[PRI] SDR0_AMP1[AOP3P] SDR0_AMP1[OP3P]	

Table 2-4. Registers Controlling PLB3 Master Priority Assignments (continued)

Master ID	Description	Register Field	Comments
5	External bus controller	EBC0_CFG[EMPL] EBC0_CFG[EMPH] SDR0_AMP1[AEBCP] SDR0_AMP1[EBCP]	Which field sets external master priority depends upon the setting of the HoldPri signal.
6:7	Reserved		

Note: PLB master priority assignments are application-dependent, and must be considered carefully in order to prevent potential lockouts of lower priority masters. For most applications, assigning a priority of 0b10 to each master is a useful starting point. See *PLB4 Arbiter Control Register (PLB4An_ACR)* on page 73 and *PLB3 Arbiter 0 Control Register (PLB3A0_ACR)* on page 101 for information about programming the PLB4A0_ACR and PLB3A0_ACR to control PLB priority mode and priority order. The PLB0_ACR must be set to "fair" mode arbitration. This helps to prevent various lockout scenarios.

2.1.3.1 Alternate PLB4 Master Priority Register 0 (SDR0_AMP0)

SDR0_AMP0 is a 32-bit read/write register containing alternative PLB4 master priority setting.

Reset value = 0 for all fields.

Figure 2-1. Alternate PLB4 Master Priority Register (SDR0_AMP0)

0:1	AICURP	Alternate ICU Read Priority 00 Lowest 01 10 11 Highest	
2:3	ADCURP	Alternate DCU Read Priority 00 Lowest 01 10 11 Highest	
4:5	ADCUWP	Alternate DCU Write Priority 00 Lowest 01 10 11 Highest	
6:7	ADMAP4P	Alternate DMA to PLB4 Priority 00 Lowest 01 10 11 Highest	
8:9	AP3P4P	Alternate PLB3 to PLB4 Bridge Priority 00 Lowest 01 10 11 Highest	
10:15		Reserved	

16	ICURP	ICU Read Priority 0 ICU read controls own priority 1 ICU read uses alternate priority	
17	DCURP	DCU Read Priority 0 DCU read controls own priority 1 DCU read uses alternate priority	
18	DCUWP	DCU Write Priority 0 DCU write controls own priority 1 DCU write uses alternate priority	
19	DMAP4P	DMA to PLB4 Priority 0 DMA to PLB4 controls own priority 1 DMA to PLB4 uses alternate priority	
20	P3P4P	PLB3 to PLB4 Priority 0 PLB3 to PLB4 controls own priority 1 PLB3 to PLB4 uses alternate priority	
21:31		Reserved	

2.1.3.2 Alternate PLB3 Master Priority Register 1 (SDR0_AMP1)

SDR0_AMP1 is a a 32-bit read/write register containing alternative PLB3 master priority setting.

Reset value = 0 for all fields.

Figure 2-2. Alternate PLB3 Master Priority Register 1 (SDR0_AMP1)

0:1	AP4P3P	Alternate PLB4 to PLB3 Bridge Priority 00 Lowest 01 10 11 Highest	
2:3	AMALP	Alternate MAL Priority 00 Lowest 01 10 11 Highest	
4:5	APCIP	Alternate PCI Priority 00 Lowest 01 10 11 Highest	
6:7	ADMA1P	Alternate DMA 1 Priority 00 Lowest 01 10 11 Highest	
8:9	AOP3P	Alternate OPB to PLB3 Bridge Priority 00 Lowest 01 10 11 Highest	
10:11	AEBCP	Alternate EBC Priority 00 Lowest 01 10 11 Highest	

12:15		Reserved	
16	P4P3P	PLB4 to PLB3 Bridge Priority 0 PLB4 to PLB3 bridge controls own priority 1 PLB4 to PLB3 bridge uses alternate priority	
17	MALP	MAL Priority 0 MAL controls own priority 1 MAL uses alternate priority	
18	PCIP	PCI Priority 0 PCI controls own priority 1 PCI uses alternate priority	
19	DMA1P	DMA 1 Priority 0 DMA 1 controls own priority 1 DMA 1 uses alternate priority	
20	OP3P	OPB to PLB3 Bridge Priority 0 OPB to PLB3 bridge controls own priority 1 OPB to PLB3 bridge uses alternate priority	
21	EBCP	EBC Priority 0 EBC controls own priority 1 EBC uses alternate priority	
22:31		Reserved	

2.1.3.3 PPC440 CPU Control Register (SDR0_CP440)

SDR0_CP440 is a 32-bit read/write register that controls processor core priorities

Reset value = see individual field definitions.

Figure 2-3. PPC440 CPU Register (SDR0_CP440)			
0:1		Reserved	Reset value = 0b00.
2:3	RL	Boot ROM Location 00 EBC 01 PCI 10 NDFC 11 Reserved	Specifies the boot source. Reset value = SDR0_SDSTP1[RL]
4:5	DRU	DcuRdUrgent 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with an urgent state in which two or more read data cache operations are pending, waiting for the previous request to be serviced. PLB master priority is updated to this value when in urgent state regardless of instruction type. Reset value = 0b11.
6:7	DRT	DcuRdTouch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with dcbt instructions except when in urgent state. Reset value = 0b10.
8:9	DRNC	DcuRdNonCache 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with non-cacheable load instructions except when in urgent state. Reset value = 0b10.

10:11	DRLC	DcuRdLdCache 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with cacheable load instructions except when in urgent state. Reset value = 0b10.
12:13	DWF	DcuWrFlush 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with flush instructions except when in urgent state. Reset value = 0b10.
14:15	DWS	DcuWrStore 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with store instructions except when in urgent state. Reset value = 0b10.
16:17	DWU	DcuWrUrgent 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with an urgent state in which two or more write data cache operations are pending, waiting for the previous request to be serviced. PLB master priority is updated to this value when in urgent state regardless of instruction type. Reset value = 0b11.
18:19	IRF	IcuRdFetch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with non-speculative ICU accesses Reset value = 0b10.
20:21	IRT	IcuRdTouch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with icbt instructions. Reset value = 0b10.
22:23	IRS	IcuRdSpec 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with speculative ICU accesses. Reset value = 0b10.
24:29		Reserved	Reset value = 0b00.
30	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	Reset value = SDR0_SDSTP1[Nto1]
31		Reserved	Reset value = 0b00.

2.1.3.4 PLB4 Master Interrupt Request Register 0 (SDR0_MIRQ0)

SDR0_MIRQ0 is a 32-bit read/write register that contains status of interrupt request generated by PLB4 slaves. A PLB4 slave can assert an interrupt request to a PLB4 master whenever the slave encounters an event which it deems important to the master. This event can be because of an operation that was initiated by the master or not. Once the slave makes an interrupt request, the request remains asserted until cleared by a subsequent access to the slave. This access can be via the PLB or DCR interface.

Software can examine the error address register and error status register of the slave making the interrupt request to determine for which transfer the interrupt request applies.

Reset value = 0 for all fields.

Figure 2-4. Master Interrupt Request Register 0 (SDR0_MIRQ0)

0		Reserved	
1	M0DDR	ICU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
2	M0P4OP	ICU read interrupt request from PLB4 to OPB bridge 0 Interrupt request inactive 1 Interrupt request active	
3	M0P4P3	ICU read interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
4:5		Reserved	
6	M1DDR	ICU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
7	M1P4OP	ICU read interrupt request from PLB4 to OPB bridge 0 Interrupt request inactive 1 Interrupt request active	
8	M1P4P3	ICU read interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
9:10		Reserved	
11	M2DDR	ICU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
12	M2P4OP	ICU read interrupt request from PLB4 to OPB bridge 0 Interrupt request inactive 1 Interrupt request active	
13	M2P4P3	ICU read interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
14:15		Reserved	
16	M3DDR	ICU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
17	M3P4OP	ICU read interrupt request from PLB4 to OPB bridge 0 Interrupt request inactive 1 Interrupt request active	
18	M3P4P3	ICU read interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	

19:20		Reserved	
21	M4DDR	ICU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
22	M4P4OP	ICU read interrupt request from PLB4 to OPB bridge 0 Interrupt request inactive 1 Interrupt request active	
23	M4P4P3	ICU read interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
24:31		Reserved	

2.1.3.5 PLB Slave Address Pipeline Register (SDR0_SLPIPE0)

SDR0_SLPIPE0 is a 32-bit read/write register that controls PLB4 and PLB3 slave address pipeline.

Figure 2-5. Slave Address Pipeline Register (SDR0_SLPIPE0)

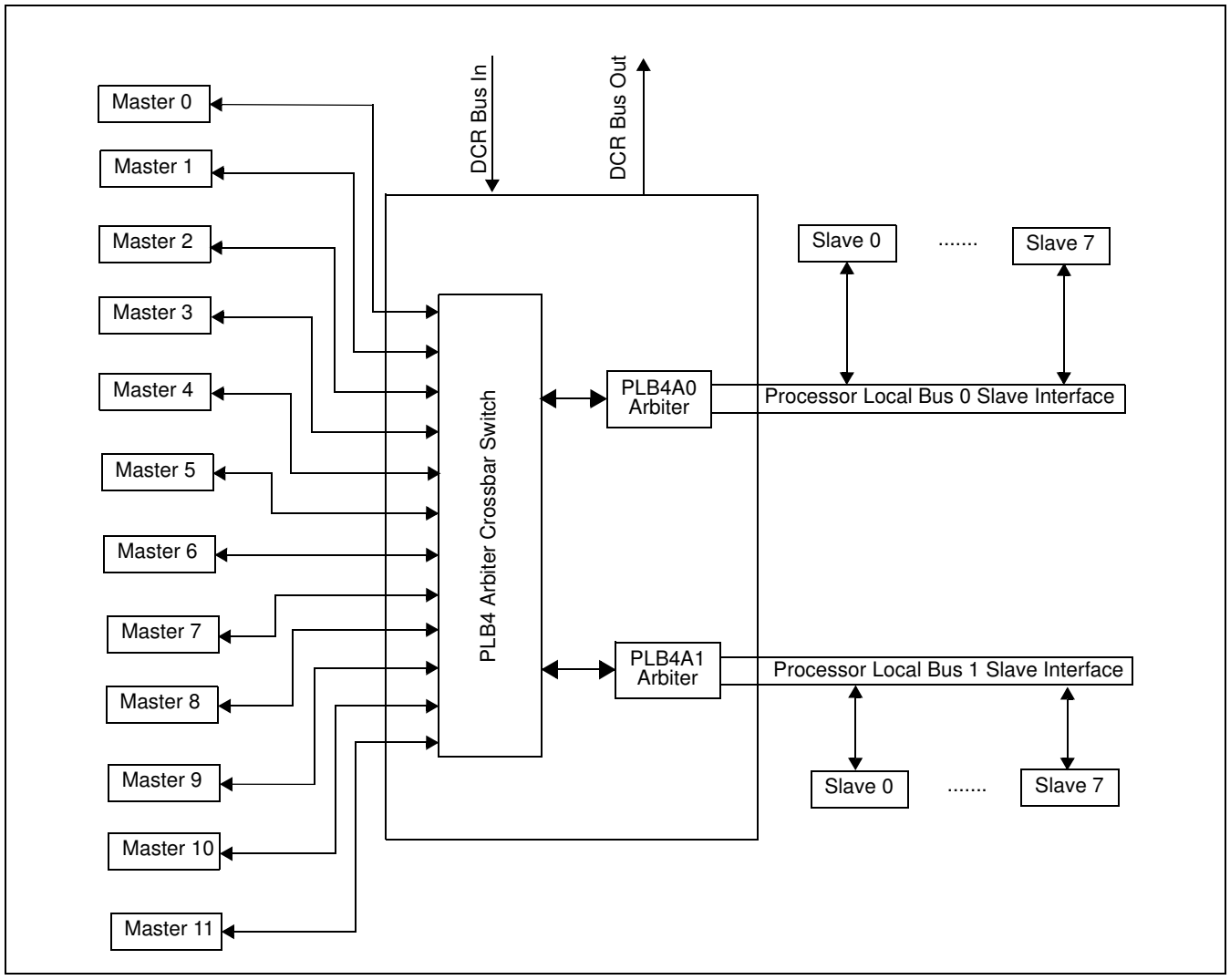
0		Reserved	
1	DAP	DDR Address Pipeline 0 DDR address pipeline disabled 1 DDR address pipeline enabled	
2	P4P3AP	PLB4 to PLB3 Address Pipeline 0 PLB4 to PLB3 address pipeline disabled 1 PLB4 to PLB3 address pipeline enabled	
3	P4OAP	PLB4 to OPB Address Pipeline 0 PLB4 to OPB address pipeline disabled 1 PLB4 to OPB address pipeline enabled	
4:27		Reserved	
28	P3OAP	PLB3 to OPB Address Pipeline 0 PLB3 to OPB address pipeline disabled 1 PLB3 to OPB address pipeline enabled	
29	EAP	EBC Address Pipeline 0 EBC address pipeline disabled 1 EBC address pipeline enabled	
30	PAP	PCI Address Pipeline 0 PCI address pipeline disabled 1 PCI address pipeline enabled	
31	P3P4AP	PLB3 to PLB4 Address Pipeline 0 PL3 to PLB4 address pipeline disabled 1 PLB3 to PLB4 address pipeline enabled	

2.1.4 PLB4 Arbiter

This is the high performance128-bit version crossbar switch supporting two processor local bus arbiters implementing the three cycle acknowledge timing protocol. It is a soft core consisting of a 12-master to 2-PLB slave segments crossbar switch including for each PLB segment: a bus arbitration control unit, a watchdog timer, address path, write data path, and read data path units.

Figure 2-6 shows how the PLB crossbar arbiter is interconnected.

Figure 2-6. PLB Crossbar Arbiter Interconnection



As shown in *Figure 2-6*, the on-chip bus structure provides a link between PLB bus masters such as the processor core, DMA controller, OPB to PLB bridge, and other PLB master devices and PLB bus slaves such as the memory controller, PLB to OPB bridge, and other PLB slave devices, residing on either the PLB0 slave interface or the PLB1 slave interface.

The processor local bus (PLB) is the high performance bus used to access memory through the bus interface units. This PLB is implemented with a crossbar switch allowing separate PLB master devices to independently access slave devices attached to PLB 0 slave interface and PLB1 slave Interface concurrently for improved PLB bus performance.

The device control register (DCR) bus is used primarily for accessing status and control registers. It is meant to off-load the PLB from the lower performance status and control read and write transfers. The DCR bus architecture allows data transfers among peripherals to occur independently from, and concurrent with, data transfers between the PLB masters and PLB slaves.

2.1.4.1 PLB4 Arbiters 0 and 1 Registers

PLB4 arbiter registers listed in *Table 2-5* are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-5. PLB4 Arbiters 0 and 1 Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB4A0_REVID	PLB4 Crossbar ID/Revision Register	0x0080	R	73
PLB4A0_ACR	PLB4A0 Arbiter Control Register	0x0081	R/W	73
PLB4A0_ESRL	PLB4A0 Error Status Register Low	0x0082	R/Clear	77
	Reserved	0x0083		
PLB4A0_EARL	PLB4A0 Error Address Register Low	0x0084	R	78
PLB4A0_EARH	PLB4A0 Error Address Register High	0x0085	R	78
PLB4A0_ESRL*	PLB4A0 Error Status Register Low (*reserved for diagnostic use only)	0x0086	Set(W)	77
PLB4A0_ESRH*	PLB4A0 Error Status Register High (*reserved for diagnostic use only)	0x0087	Set(W)	
PLB4A0_CCR	PLB4 Crossbar Control Register	0x0088	R/W	
PLB4A1_ACR	PLB4A1 Arbiter Control Register	0x0089	R/W	73
PLB4A1_ESRL	PLB4A1 Error Status Register Low	0x008A	R/Clear	77
	Reserved	0x008B		
PLB4A1_EARL	PLB4A1 Error Address Register Low	0x008C	R	78
PLB4A1_EARH	PLB4A1 Error Address Register High	0x008D	R	78
PLB4A1_ESRL*	PLB4A1 Error Status Register Low (*reserved for diagnostic use only)	0x008E	Set(W)	77
PLB4A1_ESRH*	PLB4A1 Error Status Register High (*reserved for diagnostic use only)	0x008F	Set(W)	

2.1.4.2 PLB4 Arbiter Revision ID Register (PLB4A0_REVID)

PLB4A0_REVID is a 32-bit read-only register that contains the revision ID of the PLB4 arbiter. The contents of the register can be accessed by using the move from device control register (mfdcr) instruction. The PLB4A0_REVID register can be accessed with CPU_dcrAddr(6:9) = 0x2. The register is not affected by reset.

Figure 2-7. PLB4A0 Arbiter 0 Revision ID Register (PLB4A0_REVID)

0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

2.1.4.3 PLB4 Arbiter Control Register (PLB4An_ACR)

PLB4An_ACR is a 32-bit register which controls the modes of operation for the arbiter. The priority mode, high bus utilization, read pipeline enable, and write pipeline enable are contained in this register. At reset, Fair priority mode for all priority levels is selected, high bus utilization is enabled, and two deep read and write pipelining are enabled. PLB4An_ACR[0:31] = 0xDB000000.

Figure 2-8. PLB4 Arbiter Control Register (PLB4An_ACR)

0:3	PPM	PLB Priority Mode PPM0 0 Priority level 00 Fixed priority. 1 Priority level 00 Fair priority. PPM1 0 Priority level 01 Fixed priority 1 Priority level 01 Fair priority PPM2 - Reserved (always zero) PPM3 0 Priority level 11 Fixed priority 1 Priority level 11 Fair priority	
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write pipeline disabled 1 2 Deep write pipe	
8:31		Reserved	

The following sections provide details on the effect of the bit settings in PLB4An_ACR.

PPM Field

During the bus arbitration cycle, the bus arbitration control unit uses the Mn_priority0:1 signals to determine which master will be granted the bus. The priority inputs of masters with their respective Mn_request signal asserted are used in determine the highest request priority. In addition, the core supports the fixed priority and the fair priority scheme to handle "tie" situations (that is, situations when two or more masters request the bus simultaneously while presenting the same level of request priority). The selection of the priority mode during tie situations is controlled by these bits.

Under the fixed priority scheme, each bus master is assigned a unique priority level as shown below. Note that only three distinct levels of priority are supported by the Crossbar Arbiter. A master requesting on Priority Level 2 is treated by the Crossbar Arbiter as if it had requested on Priority Level 1. *Table 2-6* shows all five bus masters in the order of priority:

Table 2-6. Master Priority Orders

Highest Priority	Decreasing Priority ----->			Lowest Priority
For Priority Level 3 (highest):				
M0	M1	M2	M3	M4
For Priority Level 2:				
Merged with Priority Level 1				
For Priority Level 1:				
M0	M1	M2	M3	M4
For Priority Level 0 (lowest):				
M0	M1	M2	M3	M4

Similarly, under the fair priority scheme, each bus master is assigned a unique priority level as shown above. However, once one master at a particular Mn_priority0:1 level is granted the bus, all other masters requesting the bus at that priority level at that time that continue to present their request at that same priority level will eventually be granted the bus before any other master (including the one originally granted the bus) is subsequently granted the bus at that priority level.

Once the last master that had not been granted the bus which is still requesting the bus on that priority level is granted the bus (or if no masters that had not been granted the bus continue to request the bus at that priority level), arbitration for the next master will be open to all requesting masters at that priority level again using the original priority above. If two masters request the bus simultaneously and continuously at the same priority level each master will be granted the bus 50% of the time at that priority level. If three masters request the bus simultaneously and continuously at the same priority level, each master will be granted the bus 33% of the time at that priority level. If four masters request the bus simultaneously and continuously at the same priority level, each master will be granted the bus 25% of the time at that priority level, etc.

This assumes all masters assert and continue to assert the same request priority. Note that a "qualified" request is a Mn_Request addressed to this PLB segment which is not otherwise blocked (for example, by the same master having a "like transaction" in progress on the other PLB segment).

The detailed manner in which this fairness algorithm works is as follows:

- For the PLB4An arbiter, there is one "fairness request" latch for each of the 12 masters on each of the three distinct priority levels (for a total of 36 latches for the PLB4An arbiter).
- If fairness on a particular level is *not* enabled, then the latches for that level are ignored (treated as if they were set to 0).
- If fairness on a particular level is enabled and *all* latches on that level are set to 0 (the initial condition), then on the arbitration cycle, arbitration on that level occurs among *all* masters with valid "qualified" requests active on that arbitration level. In the same clock cycle, the "fairness request" latches corresponding to each master with a valid "qualified" request on that arbitration level are set to one by the next clock rising edge.
- If fairness on a particular level is enabled and at least one latch on that level is set to one, then arbitration on that level occurs *only* among those masters with a valid "qualified" request on that arbitration level whose "fairness request" latch is set to one by the next clock rising edge.
- On *any* clock cycle, if a master stops presenting a valid "qualified" request on a particular arbitration level, then that master's "fairness request" latch is cleared to zero by the next clock rising edge.

- On *any* clock cycle, if arbAddrSelRegn becomes active (indicating that Master n has "won" the current arbitration), then each "fairness request" latch associated with Master n is cleared to 0 by the next rising clock edge.

In this manner, all sustained "qualified" requests which have been latched in the "fairness request" latches on the same arbitration level must be satisfied once for PLB4An before any master's request on the same level may be granted a second time on PLB4An. Note that a "qualified" request on a particular arbitration level may be removed without being granted as a result of the master dynamically changing the requested arbitration level or by the master aborting the request.

In this example, when arbAddrSelRegn is a 1 this indicates that master n has been latched as winner of the current arbitration on PLB0 (until terminated after a slave responds with addrack or rearbitrate, or the arbiter's watchdog timer issues a timeout).

HBU Bit

This bit enables a feature that will ensure both the read and write data buses are always busy if there are pending master requests for both buses. An additional requirement is that there are no bus lock requests pending or bus locked transactions in progress. These conditions will temporarily disable this feature. When this bit is 1 the arbiter will promote a lower priority request to the current active request on the address bus under the following conditions, (there are two cases, one for each data bus):

- If the read data bus is busy with a transfer and a secondary read request has been address acknowledged (addrAcked) and there are additional read request(s) pending, and the next highest request in the arbitration queue is a read request. Then a lower priority master write request will be promoted to the current active write bus request on the address bus if the write data bus is idle.
- A similar case also exists for the write bus being busy with both a primary request active and a secondary request address acknowledged (addrAcked) and the next highest request in the arbitration queue is a write request also. If a lower priority read request is pending and the read data bus is idle then the lower priority read request will be promoted to the active request on the address bus.

If read and write pipelining are disabled this feature will have no affect on arbiter operation. The default mode after reset is enabled. Setting this bit to 0 will disable this feature. It is recommended this this bit be cleared immediately after reset.

RDP Field

These bits control the depth of read pipelining. That is the number of outstanding read requests broadcast to the slaves. When PLB4An_ACR[RDP] = 01 the arbiter will default to a two deep read pipeline after reset . This allows the arbiter to broadcast a primary and secondary read to the slaves. When PLB4An_ACR[RDP] = 10 the arbiter is capable of generating a primary, secondary, and third read to the slaves. When PLB4A0_ACR[RDP] = 11 the arbiter is capable of generating a primary, secondary, third, and fourth read to the slaves. Clearing these bits disables read pipelining and only primary read transfers are broadcast. In this case PLB0_SAValid will never be asserted for a read request. The arbiter has eight separate slave address acknowledge signals, Sl_PLB0_addrAck0:7. This allows the arbiter to determine which slave is responding to the PLB0_SAValid signal assertion. The arbiter also has eight separate read primary signals, PLB0_rdPrim0:7, which should be connected to the corresponding slave. Consequently upon completion of the primary transfer the arbiter will notify the appropriate slave that its pipelined transfer is now the primary and may begin driving the data bus and controls two cycles following the assertion of PLB0_rdPrimn.

WRP Field

This bit controls the depth of write pipelining. That is the number of outstanding write requests broadcast to the slaves. When PLB0_ACR[WRP] = 1 the arbiter defaults to a two deep write pipeline after reset. This allows the arbiterto broadcast a primary and secondary write to the slaves. When PLB0_ACR[WRP] = 0 write pipelining will be disabled and only primary write transfers will be broadcast. In this case PLB0_SAValid will never be asserted for a write request.

2.1.4.4 PLB4 Error Status Register Low (PLB4An_ESRL)

The PLB4An_ESRL register described in the following figure identifies time-out errors on PLB4 bus transfers, the master initiating the transfer, and the type of transfer. Each PLB4An_ESRL[PTEn] field (n in the field is the master ID) can be locked by the master. Once locked, PLB4An_ESRL[PTEn] fields cannot be updated if a subsequent error occurs until the corresponding PLB4An_ESRL[FLKn] field is cleared. To clear a PLB4An_ESRL field, write 1 to the field. Writing 0 to a PLB4A0_ESRL field does not affect the field.

The PLB4A0_ESRL register can be accessed with DCR address = 0x82 (read/clear) and DCR address = 0x86 (set). The PLB4A1_ESRL can be accessed with DCR address = 0x8A (read/clear) and DCR address= 0x8E (set). At reset, all bits in the PLB4An_ESRL are loaded with zeroes. The registers at DCR = 0x86 and 0x8E are for diagnostic purposes only.

Figure 2-9. PLB4 Error Status Register Low (PLB4An_ESRL)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 - Processor core instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB4A0_ESR Field Lock 0 Master 0 PLB4A0_ESR field is unlocked 1 Master 0 PLB4A0_ESR field is locked	
3	ALK0	Master 0 PLB4A0_EAR Address Lock 0 Master 0 PLB4A0_EAR is unlocked 1 Master 0 PLB4A0_EAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 - Processor core data cache read unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1PLB4A0_ESR Field Lock 0 Master 1 PLB4A0_ESR field is unlocked 1 Master 1 PLB4A0_ESR field is locked	
7	ALK1	Master 1 PLB4A0_EAR Address Lock 0 Master 1 PLB4A0_EAR is unlocked 1 Master 1 PLB4A0_EAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 - Processor core data cache write unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB4A0_ESR Field Lock 0 Master 2 PLB4A0_ESR field is unlocked 1 Master 2 PLB4A0_ESR field is locked	
11	ALK2	Master 2 PLB4A0_EAR Address Lock 0 Master 2 PLB4A0_EAR is unlocked 1 Master 2 PLB4A0_EAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 - DMA2P40

13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB4A0_ESR Field Lock 0 Master 3 PLB4A0_ESR field is unlocked 1 Master 3 PLB4A0_ESR field is locked	
15	ALK3	Master 3 PLB4A0_EAR Address Lock 0 Master 3 PLB4A0_EAR is unlocked 1 Master 3 PLB4A0_EAR is locked	
16	PTE4	Master 4 PLB Timeout Error Status 0 No Master 4 timeout error 1 Master 4 timeout error	Master 4 - PLB3 to PLB 4 bridge
17	R/W4	Master 4 Read/Write Status 0 Master 4 error operation was a write 1 Master 4 error operation was a read	
18	FLK4	Master 4 PLB4A0_ESR Field Lock 0 Master 4 PLB4A0_ESR field is unlocked 1 Master 4 PLB4A0_ESR field is locked	
19	ALK4	Master 4 PLB4A0_EAR Address Lock 0 Master 4 PLB4A0_EAR is unlocked 1 Master 4 PLB4A0_EAR is locked	
20:31		Reserved	

2.1.4.5 PLB4 Error Address Register Low (PLB4An_EARL)

The read-only PLB4An_EARL register contains the lower 32 bits of the address of the access on which a bus time-out error occurred. The PLB4An_EARL can be locked by the master. Once locked, the PLB4An_EARL cannot be updated, if a subsequent error occurs, until all PLB4An_ESRL[FLCKn] and PLB4An_ESRH[FLCKn] fields are cleared (n is the master ID).

The PLB4An_EARL is not affected by Reset.

Figure 2-10. PLB4 Error Address Register Low (PLB4An_EARL)

0:31		PLB4An lower address of bus timeout error	

2.1.4.6 PLB4 Error Address Register High (PLB4An_EARH)

The read-only PLB4An_EARH register contains the upper 32 bits of the address of the access on which a bus time-out error occurred. The PLB4An_EARH can be locked by the master. Once locked, the PLB4An_EARH cannot be updated, if a subsequent error occurs, until all PLB4An_ESRL[FLCKn] and PLB4An_ESRH[FLCKn] fields are cleared (n is the master ID).

The PLB4An_EARH is not affected by Reset.

Figure 2-11. PLB4 Error Address Register High (PLB4An_EARH)

0:31		PLB4An upper address of bus timeout error

2.1.4.7 PLB4 Crossbar Control Register (PLB4A0_CCR)

The PLB Crossbar Control Register (PLB4A0_CCR) controls the modes of operation for the crossbar switch. The PLB Segmentation Address Decode selection is contained in this register. At reset PLB4A0_CCR[PSA] is loaded with the value of PGM_PLBSegAddr0:3. At reset PLB4A0_CCR[PLB3] is loaded with the value of PGM_PLB3TO0:11.

Figure 2-12. PLB4 Crossbar Control Register (PLB4A0_CCR)

0:3	PSA	PLB Segmentation Upper Address Control See *Table 2-7*	These bits, in combination with Mn_SegABus0:3, determine which slave PLB segment a Master's Mn_request is routed to by the crossbar switch. These bits must be set to 0x8.
4:15	PLB3	PLB3 TimeOut Mode Enabled For each PLB3 bit: 0 Crossbar arbiter's PLB4 timeout mode enabled for this master port 1 Crossbar arbiter's PLB3 timeout mode enabled for this master port	The bits in PLB4A0_CCR[PLB3] correspond to Master Ports 0:5 respectively. These bits must be set to 0.
16:31		Reserved	

Each bit of the PSA field determines whether the corresponding bit of Mn_SegABus0:3 will be used in determining whether to access the PLB4A0 or the PLB4A1 slave segment. If *all* of the Mn_SegABus0:3 bits for which the corresponding bit of PSA is a 1 are also 1s, then PLB4A1 is accessed. Otherwise PLB4A0 is accessed. *Table 2-7* shows how PSA bits are used with Mn_SegAbus0:3 to map master accesses to either PLB4A0 or PLB4A1.

Table 2-7. PLB Segment Access

PSA[0:3]	Mn_segAbus0:3		Access PLB Segment
	From	To	
1000	0000	0111	PLB4A0
1000	1000	1111	PLB4A1

2.1.5 PLB4 to PLB3 Bridge Registers (Bridge Out 0)

The PLB4 to PLB3 bridge registers are DCRs that are accessed using the mfdcr and mtdcr instructions.

Table 2-8. PLB4 to PLB3 Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
P4P3BO0_BESR0	PLB4 to PLB3 Bridge Error Status Register 0 Master Devices 0,1,2,3	0x0020	R/Clear	81
P4P3BO0_BEARL	PLB4 to PLB3 Bridge Error Address Register Low	0x0022	R/Clear	80
P4P3BO0_BEARH	PLB4 to PLB3 Bridge Error Address Register High	0x0023	R/Clear	80
P4P3BO0_BESR1	PLB4 to PLB3 Bridge Error Status Register 1 Master Devices 4,5,6,7	0x0024	R/Clear	83
P4P3BO0_CFG	PLB4 to PLB3 Bridge Configuration Register	0x0026	R/Clear/Set	84
P4P3BO0_PICR	PLB4 to PLB3 Bridge Priority Incrementation Counter Register	0x0027	R/Clear/Set	85
P4P3BO0_PEIR	PLB4 to PLB3 Bridge Parity Error Interrupt Register	0x0028	R/Clear/Set	86
P4P3BO0_REVID	PLB4 to PLB3 Bridge Revision ID Register	0x002A	R/Clear	86

2.1.5.1 PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL)

P4P3BO0_BEARL, described in the following figure, is a read-only register. As part of its error reporting, the PLB4 to PLB3 bridge loads the P4P3BO0_BEARL with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB4 to PLB3 bridge accepted the PLB transfer, the P4P3BO0_BEARL will lock, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P4P3BO0_BESRn). Once locked, the P4P3BO0_BEARL cannot be overwritten until all the P4P3BO0_BESRn address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-13. PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL)

0:31		Lower address of bus timeout error

2.1.5.2 PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH)

P4P3BO0_BEARH described in the following figure is a read-only register. As part of its error reporting, the PLB4-to-PLB3 bridge loads the P4P3BO0_BEARH register with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB4-to-PLB3 Bridge accepted the PLB transfer, the P4P3BO0_BEARH locks, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P4P3BO0_BESR). Once locked, the P4P3BO0_BEARH cannot be overwritten until all the P4P3BO0_BESRn address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-14. PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH)

0:31		Upper address of bus timeout error	

2.1.5.3 PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0)

In addition to driving the current PLB master's error input signal the PLB4 to PLB3 bridge also records the error information into the appropriate master's P4P3BO0_BESR0 field (provided that the register is not already locked by a previous error that was locked), where it could be optionally locked by the master having its Mn_lockError signal asserted at the time the PLB4 to PLB3 bridge accepted the PLB operation. Parity errors and the WIRQ bit operate independently of the field lock function. Thus, if a master locks its field with the lock error function and receives either a parity error or another write error then these bits will be updated.

The WIRQ bit position, in each of the master's field, is used to drive an interrupt like signal back to the system if a write error condition is encountered. This signal stays asserted until cleared by a DCR software access that resets the WIRQ bit. Note that this bit position does not behave like the PTE and R/W bit positions in that if the register is already locked by a previous error then these bit positions will not get modified from any subsequent errors. However, the WIRQ bit will always get set upon detection of a write error and only get reset by a DCR access to clear this bit. It is not affected by the field lock bit. Its function is independent of the field lock bit. For write parity errors, resetting the PAR bits will reset the interrupt, PPBx_MIRQ for the associated PLB master that encountered the write parity error on its write data bus. The PPBx__ABusParErr and PPBX__rdDBusParErr interrupt bits are both reset in the *PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR)* on page 86

Once locked, a master's P4P3BO0_BESR0 field cannot be overwritten if any subsequent data or time out error occurs, until cleared by using the move to device control register (mtdcr) instruction. To clear either P4P3BO0_BESRn, a 1 must be loaded into those register bits that are to be cleared using the proper DCR access address. Writing a 0 to any bit in either P4P3BO0_BESR will not affect the status of that bit.

The PLB4 to PLB3 bridge contains two P4P3BO0_BESR's: P4P3BO0_BESR0 logs error data for PLB masters 0-3, and P4P3BO0_BESR1 logs error data for PLB masters 4-7.

Figure 2-15. PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0)

0:1	SSET0	Secondary PLB Bus Slave Error Type for Master 0 00 No Master 0 error occurred 01 Master 0 timeout error occurred (4x mode only) 10 Master 0 data error occurred 11 Reserved	Master 0 - Processor core instruction cache unit (ICU)
2	R/W0	Read write status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3	FLK0	P4P3BO0_BESR field lock Master 0 0 Master 0 P4P3BO0_BESR field is unlocked 1 Master 0 P4P3BO0_BESR field is locked	
4	ALK0	P4P3BO0_BEAR address lock Master 0 0 Master 0 P4P3BO0_BEAR address is unlocked 1 Master 0 P4P3BO0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0) No write error detected - Master 0 interrupt request is inactive 1) Write error detected - Master 0 interrupt request is active	

6:7	PAR0	Parity Error Type Master 0 00) No parity errors detected for Master 0 01) Read bus parity error detected from secondary PLB slave device read data for Master 0 10) Address bus parity error or BE parity error detected from Master 0 11) Write bus parity error detected from data supplied by Master 0	
8:9	SSET1	Secondary PLB Bus Slave Error Type for Master 1 00 No Master 1 error occurred 01 Master 1 timeout error occurred (4x mode only) 10 Master 1 slave error occurred 11 Reserved	Master 1 - Processor core data cache read unit (DCU)
10	R/W1	Read/write status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
11	FLK1	P4P3BO0_BESR field lock Master 1 0 Master 1 P4P3BO0_BESR field is unlocked 1 Master 1 P4P3BO0_BESR field is locked	
12	ALK1	P4P3BO0_BEAR address lock Master 1 0 Master 1 P4P3BO0_BEAR address is unlocked 1 Master 1 P4P3BO0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0) No write error detected - Master 1interrupt request is inactive 1) Write error detected - Master 1interrupt request is active	
14:15	PAR1	Parity Error Type Master 1 00) No parity errors detected for Master 1 01) Read bus parity error detected from secondary PLB slave device read data for Master 1 10) Address bus parity error or BE parity error detected from Master 1 11) Write bus parity error detected from data supplied by Master 1	
16:17	SSET2	Secondary PLB Bus Slave Error Type for Master 2 00 No Master 2 error occurred 01 Master 2 timeout error occurred (4x mode only) 10 Master 2 slave error occurred 11 Reserved	Master 2 - Processor core data cache writeunit (DCU)
18	R/W2	Read/write status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	P4P3BO0_BESR field lock Master 2 0 Master 2 P4P3BO0_BESR field is unlocked 1 Master 2P4P3BO0_BESR field is locked	
20	ALK2	P4P3BO0_BEAR address lock Master 2 0 Master 2 P4P3BO0_BEAR address is unlocked 1 Master 2 P4P3BO0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0) No write error detected - Master 2interrupt request is inactive 1) Write error detected - Master 2interrupt request is active	

22:23	PAR2	Parity Error Type Master 2 00) No parity errors detected for Master 2 01) Read bus parity error detected from secondary PLB slave device read data for Master 2 10) Address bus parity error or BE parity error detected from Master 2 11) Write bus parity error detected from data supplied by Master 2	
24:25	SSET3	Secondary PLB Bus Slave Error Type for Master 3 00 No Master 3 error occurred 01 Master 3 timeout error occurred (4x mode only) 10 Master 3 slave error occurred 11 Reserved	Master 3 - DMA2P40
26	R/W3	Read/write status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
27	FLK3	P4P3BO0_BESR field lock Master 3 0 Master 3 P4P3BO0_BESR field is unlocked 1 Master 3 P4P3BO0_BESR field is locked	
28	ALK3	P4P3BO0_BEAR address lock Master 3 0 Master 3 P4P3BO0_BEAR address is unlocked 1 Master 3 P4P3BO0_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0) No write error detected - Master 3 interrupt request is inactive 1) Write error detected - Master 3 interrupt request is active	
30:31	PAR3	Parity Error Type Master 3 00) No parity errors detected for Master 3 01) Read bus parity error detected from secondary PLB slave device read data for Master 3 10) Address bus parity error or BE parity error detected from Master 3 11) Write bus parity error detected from data supplied by Master 3	

2.1.5.4 PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1)

Figure 2-16. PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1)

0:1	SSET4	Secondary PLB Bus Slave Error Type for master 4 00 No master 4 error occurred 01 Master 4 timeout error occurred (4x mode only) 10 Master 4 slave error occurred 11 Reserved	Master 4 - DMA2P40
2	R/W4	Read write status master 4 0 Master 4 error operation is a write 1 Master 4 error operation is a read	
3	FLK4	P4P3BO0_BESR field lock master 4 0 Master 4 P4P3BO0_BESR field is unlocked 1 Master 4 P4P3BO0_BESR field is locked	
4	ALK4	P4P3BO0_BEAR address lock master 4 0 Master 4 P4P3BO0_BEAR address is unlocked 1 Master 4 P4P3BO0_BEAR address is locked	

5	WIRQ4	Write Error Interrupt master 4 0) No write error detected - master 4 interrupt request is inactive 1) Write error detected - master 4 interrupt request is active	
6:7	PAR4	Parity Error Type master 4 00) No parity errors detected for master 4 01) Read bus parity error detected from secondary PLB slave device read data for master 4 10) Address bus parity error or BE parity error detected from master 4 11) Write bus parity error detected from data supplied by master 4	
8:31		Reserved	

2.1.5.5 PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG)

Figure 2-17. PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG)

0	SSPS	Slave side primary/secondary priority 0 This PLB4-to-PLB3 Bridge Slave side is the secondary Slave side. 1 This PLB4-to-PLB3 Bridge Slave side is the primary Slave side.	
1:2	FIFOE	FIFO Enablement 00 PLB4-to-PLB3 Bridge Address FIFO is disabled. 01 PLB4-to-PLB3 Bridge Acknowledged Address FIFO is set to one level deep. 10 PLB4-to-PLB3 Bridge Acknowledged Address FIFO is set to two levels deep. 11 Reserved	These are read only bits
3	SRE	Secondary Read Enablement 0 0) PLB4-to-PLB3 Bridge secondary read feature is disabled. 1 1) PLB4-to-PLB3 Bridge secondary read feature is enabled.	Read only bit
4	SWE	Secondary Write Enablement 0 PLB4-to-PLB3 Bridge secondary write feature is disabled. 1 PLB4-to-PLB3 Bridge secondary write feature is enabled.	Read only bit
5	PBASS	PLB Bus Architecture for Slave Side 0 PLB4-to-PLB3 Bridge Slave side is connected to a 3.x PLB Bus. 1 PLB4-to-PLB3 Bridge Slave side is connected to a 4.x PLB Bus.	
6	PBAMS	PLB Bus Architecture for Master Side 0 PLB4-to-PLB3 Bridge Master side is connected to a 3.x PLB Bus. 1 PLB4-to-PLB3 Bridge Master side is connected to a 4.x PLB Bus.	

7	ER	Enable Rearbitration 0 PLB4-to-PLB3 Bridge Slave side rearbitration is disabled. 1 PLB4-to-PLB3 Bridge Slave side rearbitration is enabled.	Read only
8:9	PRI	PLB Priority Bits 00 Lowest 01 Next Highest 10 Next Highest 11 Highest	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. **During reset**, these bits are set to the value present on the PGM_PPB4_priority input.
10	PE	Parity Enable 0 Parity checking is disabled 1 Parity checking is enabled on a per side basis depending on whether connected to a 4.x PLB Bus.	**During reset** this bit is set to the value present on PGM_PPB4_parEnable input .
11	FI	Flush Interrupt Enable 0 Interrupt flushing of writes is disabled and the interrupt will not be forwarded to B-PLB slave via FlushIntComp 1 Interrupt flushing of writes from A-PLB master 'x' is enabled and will be forwarded to B-PLB slave via FlushIntComp	
12:14	FIID	Flush Interrupt ID	Value programmed must match the master ID that the A-PLB bus master 'X' is connected to. (For example, if master 'X' is connected to port 4 of the A-PLB arbiter then the FIID should be programmed to 0b100)
15:31		Reserved	

2.1.5.6 PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR)

When PPBx_PICR[PICE] = 0, PLB4 to PLB3 master side request priority will not increment and is fixed at the programmed value, set in P4P3BO0_CONFG[8:9].

When P4P3BO0_PICR[PICE] = 1, (this is the default after system reset) if the PLB4-to-PLB3 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. Then, if the PLB4-to-PLB3 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. This continues until 0b11 is reached. For next request, priority value starts out at the original value.

Initial Priority Incrementation Counter Value (1:5 bits): The PLB4-to-PLB3 bridge counts blocks of 8 PLB clocks for each value of the counter. Thus, for a count value of 0b00000 the counter would count 8 PLB clocks and increment its request priority if it has not received an acknowledgement from the target PLB slave device. A count value of 0b00001 would expire after 16 PLB clocks, a value of 0b00010 would expire after 24 PLB clocks, etc.

Figure 2-18. PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR)

0	PICE	Priority Incrementation Counter Enable 0 Counter is disabled. 1 Counter is enabled.	
1:5	IPICV	Initial Priority Incrementation Counter Value The maximum count value, 0b11111, is 256 PLB clocks, or 1.92 us at 133MHz.	Value resets to zero. The maximum count value, 0b11111, is 256 PLB clocks, or 1.92 us at 133MHz.
6:31		Reserved	

2.1.5.7 PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR)

Figure 2-19. PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR)			
0	ADRPE	A Bus Parity Error Interrupt 0 No Upper address, lower address or byte enable parity errors detected. 1 Parity error detected on either the upper address, lower address of byte enables of requested transfer to PLB4-to-PLB3 Bridge. PLB4-to-PLB3 address parity error interrupt is active.	
1	RDPE	Read Data Bus Parity Error Interrupt 0 No read data bus parity error detected 1 Read data bus parity error detected from secondary PLB Bus slave device data transfer to PLB4-to-PLB3 Bridge. PLB4-to-PLB3 read data parity error interrupt is active.	
2:31		Reserved	

2.1.5.8 PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID)

Figure 2-20. PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL (hard wired to 0x2)
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL (hard wired to 0x1)

2.1.6 PLB3 to PLB4 Bridge Registers (Bridge In 0)

The PLB3 to PLB4 bridge registers are DCRs that are accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-9. PLB3 to PLB4 Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
P3P4BI0_BESR0	PLB3 to PLB4 Bridge Error Status Register 0 Master Devices 0, 1, 2, 3	0x0030	R/Clear	87
P3P4BI0_BEARL	PLB3 to PLB4 Bridge Error Address Register Low	0x0032	R/Clear	87
P3P4BI0_BEARH	PLB3 to PLB4 Bridge Error Address Register High	0x0033	R/Clear	87
P3P4BI0_BESR1	PLB3 to PLB4 Bridge Error Status Register 1 Master Devices 4, 5, 6, 7	0x0034	R/Clear	90
P3P4BI0_CFG	PLB3 to PLB4 Bridge Configuration Register	0x0036	R/Clear/Set	91
P3P4BI0_PICR	PLB3 to PLB4 Bridge Priority Incrementation Counter Register	0x0037	R/Clear/Set	92
P3P4BI0_PEIR	PLB3 to PLB4 Bridge Parity Error Interrupt Register	0x0038	R/Clear/Set	93
P3P4BI0_REVID	PLB3 to PLB4 Bridge Revision ID Register	0x003A	R/Clear	93

2.1.6.1 PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL)

P3P4BI0_BEARL is a read-only register. As part of its error reporting, the PLB3 to PLB4 bridge loads the P3P4BI0_BEARL with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB3 to PLB4 bridge accepted the PLB transfer, the P3P4BI0_BEARL will lock, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P3P4BI0_BESR). Once locked, the P3P4BI0_BEARL cannot be over-written until all the P3P4BI0_BESR address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-21. PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL)

0:31		Lower address of bus timeout error	

2.1.6.2 PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH)

P3P4BI0_BEARH is a read-only register. As part of its error reporting, the PLB3-to-PLB4 bridge loads the P3P4BI0_BEARH register with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB3-to-PLB4 Bridge accepted the PLB transfer, the P3P4BI0_BEARH locks, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P3P4BI0_BESR). Once locked, the P3P4BI0_BEARH cannot be overwritten until all the P3P4BI0_BESR address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-22. PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH)

0:31		Upper address of bus timeout error	

2.1.6.3 PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0)

In addition to driving the current PLB master's error input signal the PLB3 to PLB4 bridge also records the error information into the appropriate master's P3P4BI0_BESR0 field (provided that the register is not already locked by a previous error that was locked), where it could be optionally locked by the master having its Mn_lockError signal asserted at the time the PLB3 to PLB4 bridge accepted the PLB operation. Parity errors and the WIRQ bit operate independently of the field lock function. Thus, if a master locks its field with the lock error function and receives either a parity error or another write error then these bits will be updated.

The WIRQ bit position, in each of the master's field, is used to drive an interrupt like signal back to the system if a write error condition is encountered. This signal stays asserted until cleared by a DCR software access that resets the WIRQ bit. Note that this bit position does not behave like the PTE and R/W bit positions in that if the register is already locked by a previous error then these bit positions will not get modified from any subsequent errors. However, the WIRQ bit will always get set upon detection of a write error and only get reset by a DCR access to clear this bit. It is not affected by the field lock bit. Its function is independent of the field lock bit. For write parity errors, resetting the PAR bits will reset the interrupt, PPBx_MIRQ for the associated PLB master that encountered the write parity error on its write data bus. The PPBx__ABusParErr and PPBx__rdDBusParErr interrupt bits are both reset in the *PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR)* on page 93

Once locked, a master's P3P4BI0_BESR0 field cannot be overwritten if any subsequent data or time out error occurs, until cleared b using the move to device control register (mtdcr) instruction. To clear either P3P4BI0_BESRn, a 1 must be loaded into those register bits that are to be cleared using the proper DCR access address. Writing a 0 to any bit in either P3P4BI0_BESR will not affect the status of that bit.

The PLB3 to PLB4 bridge contains two P3P4BI0_BESR's: P3P4BI0_BESR0 logs error data for PLB masters 0-3, and P3P4BI0_BESR1 logs error data for PLB masters 4-7.

Figure 2-23. PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0)

0:1	SSET0	Secondary PLB Bus Slave Error Type for master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred (4x mode only) 10 Master 0 data error occurred 11 Reserved	
2	R/W0	Read write status master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3	FLK0	P3P4BI0_BESR field lock master 0 0 Master 0 P3P4BI0_BESR field is unlocked 1 Master 0 P3P4BI0_BESR field is locked	
4	ALK0	P3P4BI0_BEAR address lock master 0 0 Master 0 P3P4BI0_BEAR address is unlocked 1 Master 0 P3P4BI0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt master 0 0) No write error detected - master 0 interrupt request is inactive 1) Write error detected - master 0 interrupt request is active	
6:7	PAR0	Parity Error Type master 0 00) No parity errors detected for master 0 01) Read bus parity error detected from secondary PLB slave device read data for master 0 10) Address bus parity error or BE parity error detected from master 0 11) Write bus parity error detected from data supplied by master 0	
8:9	SSET1	Secondary PLB Bus Slave Error Type for master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred (4x mode only) 10 Master 1 slave error occurred 11 Reserved	
10	R/W1	Read/write status master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
11	FLK1	P3P4BI0_BESR field lock master 1 0 Master 1 P3P4BI0_BESR field is unlocked 1 Master 1 P3P4BI0_BESR field is locked	
12	ALK1	P3P4BI0_BEAR address lock master 1 0 Master 1 P3P4BI0_BEAR address is unlocked 1 Master 1 P3P4BI0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt master 1 0) No write error detected - master 1interrupt request is inactive 1) Write error detected - master 1interrupt request is active	

14:15	PAR1	Parity Error Type master 1 00) No parity errors detected for master 1 01) Read bus parity error detected from secondary PLB slave device read data for master 1 10) Address bus parity error or BE parity error detected from master 1 11) Write bus parity error detected from data supplied by master 1	
16:17	SSET2	Secondary PLB Bus Slave Error Type for master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred (4x mode only) 10 Master 2 slave error occurred 11 Reserved	
18	R/W2	Read/write status master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	P3P4BI0_BESR field lock master 2 0 Master 2 P3P4BI0_BESR field is unlocked 1 Master 2P3P4BI0_BESR field is locked	
20	ALK2	P3P3BO0_BEAR address lock master 2 0 Master 2 P3P4BI0_BEAR address is unlocked 1 Master 2 P3P4BI0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt master 2 0) No write error detected - master 2interrupt request is inactive 1) Write error detected - master 2interrupt request is active	
22:23	PAR2	Parity Error Type master 2 00) No parity errors detected for master 2 01) Read bus parity error detected from secondary PLB slave device read data for master 2 10) Address bus parity error or BE parity error detected from master 2 11) Write bus parity error detected from data supplied by master 2	
24	SSET3	Secondary PLB Bus Slave Error Type for master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred (4x mode only) 10 Master 3 slave error occurred 11 Reserved	
25	R/W3	Read/write status master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
26	FLK3	P3P4BI0_BESR field lock master 3 0 Master 3 P3P4BI0_BESR field is unlocked 1 Master 3 P3P4BI0_BESR field is locked	

27	ALK3	P3P4BI0_BEAR address lock master 3 0 Master 3 P3P4BI0_BEAR address is unlocked 1 Master 3 P3P4BI0_BEAR address is locked	
28	WIRQ3	Write Error Interrupt master 3 0) No write error detected - master 3interrupt request is inactive 1) Write error detected - master 3interrupt request is active	
30:31	PAR3	Parity Error Type master 3 00) No parity errors detected for master 3 01) Read bus parity error detected from secondary PLB slave device read data for master 3 10) Address bus parity error or BE parity error detected from master 3 11) Write bus parity error detected from data supplied by master 3	

2.1.6.4 PLB3 to PLB4 Bridge Error Status Register 1 (P3P43BI0_BESR1)

Figure 2-24. PLB3 to PLB4 Bridge Error Status Register 1 (P3P4BI0_BESR1)

0:1	SSET4	Secondary PLB Bus Slave Error Type for master 4 00 No master 4 error occurred 01 Master 4 timeout error occurred (4x mode only) 10 Master 4 slave error occurred 11 Reserved	
2	R/W4	Read write status master 4 0 Master 4 error operation is a write 1 Master 4 error operation is a read	
3	FLK4	P3P4BI0_BESR field lock master 4 0 Master 4 P3P4BI0_BESR field is unlocked 1 Master 4 P3P4BI0_BESR field is locked	
4	ALK4	P3P4BI0_BEAR address lock master 4 0 Master 4 P3P4BI0_BEAR address is unlocked 1 Master 4 P3P4BI0_BEAR address is locked	
5	WIRQ4	Write Error Interrupt master 4 0) No write error detected - master 4 interrupt request is inactive 1) Write error detected - master 4 interrupt request is active	
6:7	PAR4	Parity Error Type master 4 00) No parity errors detected for master 4 01) Read bus parity error detected from secondary PLB slave device read data for master 4 10) Address bus parity error or BE parity error detected from master 4 11) Write bus parity error detected from data supplied by master 4	
8:9	SSET5	Secondary PLB Bus Slave Error Type for master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred (4x mode only) 10 Master 5 slave error occurred 11 Reserved	
10	R/W5	Read/write status master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read	

11	FLK5	P3P4BI0_BESR field lock master 5 0 Master 5 P3P4BI0_BESR field is unlocked 1 Master 5 P3P4BI0_BESR field is locked	
12	ALK5	P3P4BI0_BEAR address lock master 5 0 Master 5 P3P4BI0_BEAR address is unlocked 1 Master 5 P3P4BI0_BEAR address is locked	
13	WIRQ5	Write Error Interrupt master 5 0) No write error detected - master 5interrupt request is inactive 1) Write error detected - master 5interrupt request is active	
14:15	PAR5	Parity Error Type master 5 00) No parity errors detected for master 5 01) Read bus parity error detected from secondary PLB slave device read data for master 5 10) Address bus parity error or BE parity error detected from master 5 11) Write bus parity error detected from data supplied by master 5	
16:31		Reserved	

2.1.6.5 PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG)

Figure 2-25. PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG)			
0	SSPS	Slave side primary/secondary priority 0 This PLB3-to-PLB4 Bridge Slave side is the secondary Slave side. 1 This PLB3-to-PLB4 Bridge Slave side is the primary Slave side.	
1:2	FIFOE	FIFO Enablement 00 PLB3-to-PLB4 Bridge Address FIFO is disabled. 01 PLB3-to-PLB4 Bridge Acknowledged Address FIFO is set to one level deep. 10 PLB3-to-PLB4 Bridge Acknowledged Address FIFO is set to two levels deep. 11 Reserved	These are read only bits
3	SRE	Secondary Read Enablement 0 0) PLB3-to-PLB4 Bridge secondary read feature is disabled. 1 1) PLB3-to-PLB4 Bridge secondary read feature is enabled.	Read only bit
4	SWE	Secondary Write Enablement 0 PLB3-to-PLB4 Bridge secondary write feature is disabled. 1 PLB3-to-PLB4 Bridge secondary write feature is enabled.	Read only bit
5	PBASS	PLB Bus Architecture for Slave Side 0 PLB3-to-PLB4 Bridge Slave side is connected to a 3.x PLB Bus. 1 PLB3-to-PLB4 Bridge Slave side is connected to a 4.x PLB Bus.	

6	PBAMS	PLB Bus Architecture for Master Side 0 PLB3-to-PLB4 Bridge Master side is connected to a 3.x PLB Bus. 1 PLB3-to-PLB4 Bridge Master side is connected to a 4.x PLB Bus.	
7	ER	Enable Rearbitration 0 PLB3-to-PLB4 Bridge Slave side rearbitration is disabled. 1 PLB3-to-PLB4 Bridge Slave side rearbitration is enabled.	Read only
8:9	PRI	PLB Priority Bits 00 Lowest 01 Next Highest 10 Next Highest 11 Highest	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_PPB3_priority input.
10	PE	Parity Enable 0 Parity checking is disabled 1 Parity checking is enabled on a per side basis depending on whether connected to a 4.x PLB Bus.	During reset this bit is set to the value present on PGM_PPB3_parEnable input .
11	FI	Flush Interrupt Enable 0 Interrupt flushing of writes is disabled and the interrupt will not be forwarded to B-PLB slave via FlushIntComp 1 Interrupt flushing of writes from A-PLB master X is enabled and will be forwarded to B-PLB slave via FlushIntComp	
12:14	FIID	Flush Interrupt ID	Value programmed must match the master ID that the A-PLB bus master X is connected to. (For example, if master X is connected to port 4 of the A-PLB arbiter then the FIID should be programmed to 0b100)
15:31	Reserved		

2.1.6.6 PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR)

When PPBx_PICR[PICE] = 0, PLB3 to PLB4 master side request priority will not increment and is fixed at the programmed value, set in P3P4BI0_CFG[8:9].

When P3P4BI0_PICR[PICE] = 1, (this is the default after system reset) if the PLB3-to-PLB4 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. Then, if the PLB3-to-PLB4 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. This continues until 2'b11 is reached. For next request, priority value starts out at the original value.

Initial Priority Incrementation Counter Value (1:5 bits): The PLB3-to-PLB4 bridge counts blocks of 8 PLB clocks for each value of the counter. Thus, for a count value of 0b00000 the counter would count 8 PLB clocks and increment its request priority if it has not received an acknowledgement from the target PLB slave device. A count value of 0b00001 would expire after 16 PLB clocks, a value of 0b00010 would expire after 24 PLB clocks, etc.

Figure 2-26. PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR)			
0	PICE	Priority Incrementation Counter Enable 0 Counter is disabled. 1 Counter is enabled.	
1:5	IPICV	Initial Priority Incrementation Counter Value The maximum count value, 0b11111, is 256 PLB clocks, or 1.92 us at 133MHz.	Value resets to zero. The maximum count value, 0b11111, is 256 PLB clocks, or 1.92 us at 133MHz.
6:31		Reserved	

2.1.6.7 PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR)

Figure 2-27. PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR)			
0	ADRPE	A Bus Parity Error Interrupt 0 No Upper address, lower address or byte enable parity errors detected. 1 Parity error detected on either the upper address, lower address of byte enables of requested transfer to PLB3-to-PLB4 Bridge. PLB3-to-PLB4 address parity error interrupt is active.	
1	RDPE	Read Data Bus Parity Error Interrupt 0 No read data bus parity error detected 1 Read data bus parity error detected from secondary PLB Bus slave device data transfer to PLB3-to-PLB4 Bridge. PLB3-to-PLB4 read data parity error interrupt is active.	
2:31		Reserved	

2.1.6.8 PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID)

Figure 2-28. PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

2.1.7 PLB4 to OPB Bridge Registers

The PLB4 to OPB bridge registers listed in the following figure are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-10. PLB4 to OPB Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB42OPB0_BESR0	PLB4 to OPB Bridge Error Status Register 0 (Master IDs 0,1, 2, 3)	0x200	R/Clear	94
PLB42OPB0_BEARL	PLB4 to OPB Bridge Error Address Register Low	0x202	R/O	96
PLB42OPB0_BEARH	PLB4 to OPB Bridge Error Upper Address Register High	0x203	R/O	96
PLB42OPB0_BESR1	PLB4 to OPB Bridge Error Status Register (Master IDs 4, 5)	0x204	R/Clear	96
PLB42OPB0_CFG	PLB4 to OPB Bridge Error Configuration Register	0x206	R/Clear	97
PLB42OPB0_LATENCY	PLB4 to OPB Bridge Burst Latency Timer	0x208	R/Clear	98
PLB42OPB0_REVID	PLB4 to OPB Bridge Revision ID Register	0x20A	R/O	98

2.1.7.1 PLB4 to OPB Bridge Error Status Register 0 (PLB42OPB0_BESR0)

The PLB4 to OPB bridge writes error information into the appropriate PLB42OPB0_BESR*m* register. (For master IDs 0, 1, 2, and 3, *m* = 0; for master IDs 4 and 5, *m* = 1.)

PLB42OPB0_BESR*m* fields can be locked using the PLB42OPB0_BESR*m*[FLK*n*] and PLB42OPB0_BESR*m*[ALK*n*] fields (*n* is the master ID). Once locked, the PLB42OPB0_BESR*m* fields associated with a master cannot be overwritten if a subsequent error occurs until the locking fields are cleared. To clear a lock, write 1 to the PLB42OPB0_BESR*m*[FLK*n*] and PLB42OPB0_BESR*m*[ALK*n*] fields that are set. Writing 0 to a lock field does not affect the field

Figure 2-29. PLB4 to OPB Bridge Error Status Register 0 (PLB42OPB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	PLB4 master 0 is the read-only instruction cache unit (ICU)
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3	FLK0	PLB42OPB0_BESR0 Field Lock Master 0 0 Master 0 PLB42OPB0_BESR0 field is unlocked 1 Master 0 PLB42OPB0_BESR0 field is locked	
4	ALK0	PLB42OPB0_BEAR Address Lock Master 0 0 Master 0 PLB42OPB0_BEAR address is unlocked 1 Master 0 PLB42OPB0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active	
6:7		Reserved	

8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	PLB4 master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
11	FLK1	PLB42OPB0_BESR0 Field Lock Master 1 0 Master 1 PLB42OPB0_BESR0 field is unlocked 1 Master 1 PLB42OPB0_BESR0 field is locked	
12	ALK1	PLB42OPB0_BEAR Address Lock Master 1 0 Master 1 PLB42OPB0_BEAR address is unlocked 1 Master 1 PLB42OPB0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active	
14:15		Reserved	
16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	PLB4 master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	PLB42OPB0_BESR0 Field Lock Master 2 0 Master 2 PLB42OPB0_BESR0 field is unlocked 1 Master 2 PLB42OPB0_BESR0 field is locked	
20	ALK2	PLB42OPB0_BEAR Address Lock Master 2 0 Master 2 PLB42OPB0_BEAR address is unlocked 1 Master 2 PLB42OPB0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	
24:25	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	PLB4 master 3 is the DMA2P40 unit
26	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
27	FLK3	PLB42OPB0_BESR0 Field Lock Master 3 0 Master 3 PLB42OPB0_BESR0 field is unlocked 1 Master 3 PLB42OPB0_BESR0 field is locked	
28	ALK3	PLB42OPB0_BEAR Address Lock Master 3 0 Master 3 PLB42OPB0_BEAR address is unlocked 1 Master 3 PLB42OPB0_BEAR address is locked	

29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active	
30:31		Reserved	

2.1.7.2 PLB4 to OPB Bridge Error Address Register Low (PLB42OPB0_BEARL)

The read-only PLB42OPB0_BEARL reports the lower 32 bits of the address of a PLB4 to OPB transfer that results in an error. The PLB4 to OPB bridge writes the error address in the PLB42OPB0_BEARL, unless the associated PLB42OPB0_BESR*m*[ALCK*n*] field is set (*m* is either 0 or 1, depending on the master ID specified by *n*). Once locked, the PLB4 to OPB bridge cannot write PLB42OPB0_BEARL until all PLB42OPB0_BESR*m*[ALCK*n*] fields that are set are cleared.

Figure 2-30. PLB4 to OPB Bridge Error Address Register Low (PLB42OPB0_BEARL)

0:31	BEARL	Lower address of bus error	

2.1.7.3 PLB4 to OPB Bridge Error Address Register High (PLB42OPB0_BEARH)

The read-only PLB42OPB0_BEARH reports the upper four address bits of a PLB4 to OPB transfer that results in an error. The PLB4 to OPB bridge writes the error address in the PLB42OPB0_BEARH, unless the associated PLB42OPB0_BESR*m*[ALCK*n*] field is set (*m* is either 0 or 1, depending on the master ID specified by *n*). Once locked, the PLB4 to OPB bridge cannot write PLB42OPB0_BEARH until all PLB42OPB0_BESR*m*[ALCK*n*] fields that are set are cleared.

Figure 2-31. PLB4 to OPB Bridge Error Address Register High (PLB42OPB0_BEARH)

0:27		Reserved	
28:31	UA	Upper address of bus error	

2.1.7.4 PLB4 to OPB Bridge Error Status Register 1(PLB42OPB0_BESR1)

Figure 2-32. PLB4 to OPB Bridge Error Status Register 1 (PLB42OPB0_BESR1)

0:1	PTE4	PLB Timeout Error Status Master 4 00 No master 4 error occurred 01 Master 4 timeout error occurred 10 Master 4 slave error occurred 11 Reserved	Master 4 is the PLB3 to PLB4 bridge.
2	R/W4	Read/Write Status Master 4 0 Master 4 error operation is a write 1 Master 4 error operation is a read	
3	FLK4	PLB42OPB0_BESR1 Field Lock Master 4 0 Master 4 PLB42OPB0_BESR1 field is unlocked 1 Master 4 PLB42OPB0_BESR1 field is locked	

4	ALK4	PLB42OPB0_BEAR Address Lock Master 4 0 Master 4 PLB42OPB0_BEAR address is unlocked 1 Master 4 PLB42OPB0_BEAR address is locked	
5	WIRQ4	Write Error Interrupt Master 4 0 No write error detected - master 4 interrupt request is inactive 1 Write error detected - master 4 interrupt request is active	
6:7	PTE5	PLB Timeout Error Status Master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	
8	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read	
9	FLK5	PLB42OPB0_BESR1 Field Lock Master 5 0 Master 5 PLB42OPB0_BESR1 field is unlocked 1 Master 5 PLB42OPB0_BESR1 field is locked	
10	ALK5	PLB42OPB0_BEAR Address Lock Master 5 0 Master 5 PLB42OPB0_BEAR address is unlocked 1 Master 5 PLB42OPB0_BEAR address is locked	
11	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - master 5 interrupt request is inactive 1 Write error detected - master 5 interrupt request is active	
12:31		Reserved	

2.1.7.5 PLB4 to OPB Bridge Configuration Register (PLB42OPB0_CFG)

Figure 2-33. PLB4 to OPB Bridge Configuration Register (PLB42OPB0_CFG)

0	ER	Enable rearbitration 0 PLB4 to OPB bridge rearbitration is enabled 1 PLB4 to OPB bridge rearbitration is disabled
1	L32R	Line 32 bit read 0 PLB4 to OPB bridge reads line operations at requested size of master 1 PLB4 to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

2.1.7.6 PLB4 to OPB Bridge Burst Latency Timer (PLB42OPB0_LATENCY)

Figure 2-34. PLB4 to OPB Bridge Burst Latency Timer Register (PLB42OPB0_LATENCY)

0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled	
1:4	LC	Latency Count 0000 Minimum count value. 1111 Maximum count value. PLB4 to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.	When PLB42OPB0_LATENCY[LC=0000] PLB4 to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. When PLB42OPB0_LATENCY[LC=1111] PLB4 to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved	

2.1.7.7 PLB4 to OPB Bridge Revision ID (PLB42OPB0_REVID)

Figure 2-35. PLB4 to OPB Bridge Revision ID Register (PLB42OPB0_REVID)

0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL (hard wired to 0x1)
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL (hard wired to 0x22)

2.1.8 PLB3 Arbiter Registers

PLB3 arbiter registers are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-11. PLB3 Arbiter 0 Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB3A0_REVID	PLB3 Arbiter 0 Revision ID Regisgter	0x072	R	98
PLB3A0_BESR	PLB3 Error Status Register	0x074	R/W	99
PLB3A0_BEAR	PLB3 Error Address Register	0x076	R	100
PLB3A0_ACR	PLB3 Arbiter Control Register	0x077	R/W	101

2.1.8.1 PLB3 Arbiter Revision ID Register (PLB3A0_REVID)

PLB3A0_REVID is a 32-bit read-only register that contains the revision ID of the PLB3 arbiter. The contents of the register can be accessed by using the move from device control register (mfdcr) instruction. The PLB3A0_REVID register can be accessed with CPU_dcrAddr6:9 = 0x2. The register is not affected by reset.

Figure 2-36. PLB3 Arbiter 0 Revision ID Register (PLB3A0_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

2.1.8.2 PLB3 Error Status Register (PLB3A0_BESR)

The read/clear PLB3A0_BESR identifies timeout errors on PLB3 bus transfers, the master initiating the transfer, and the type of transfer.

The PCI Bridge and Memory Access Layer (MAL) masters may qualify their PLB transactions such that the information describing any timeout errors that occur during these transfers becomes locked. Once locked, a master's PLB3A0_BESR[PTEn] and PLB3A0_BESR[R/Wn] fields cannot be updated until the corresponding PLB3A0_BESR[FLKn] field is cleared. To clear a PLB3A0_BESR field, write 1 to the field. Writing 0 to a PLB3A0_BESR field does not affect the field.

For PLB3A0_BESR[FLK3, AL3] to become locked, an error must occur while the PCI bridge has error locking enabled (PCIC0_BRDGOPT1[PLESE] = 1). For PLB3A0_BESR[FLK4, AL4] to become locked, an error must occur while the MAL has error locking enabled (MAL0_CFG[PLBLE]=1).

Figure 2-37. PLB3 Error Status Register (PLB3A0_BESR)			
0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	PLB3 master 0 is PLB4 to PLB3 bridge.
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB3A0_BESR Field Lock 0 Master 0 PLB3A0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB3A0_BEAR Address Lock 0 Master 0 PLB3A0_BEAR is unlocked 1 Master 0 PLB3A0_BEAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	PLB3 master 1 is MAL.
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1 PLB3A0_BESR Field Lock 0 Master 1 PLB3A0_BESR field is unlocked 1 Master 1 PLB3A0_BESR field is locked	
7	ALK1	Master 1 PLB3A0_BEAR Address Lock 0 Master 1 PLB3A0_BEAR is unlocked 1 Master 1 PLB3A0_BEAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is PCI bridge.

9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB3A0_BESR Field Lock 0 Master 2 PLB3A0_BESR field is unlocked 1 Master 2 PLB3A0_BESR field is locked	
11	ALK2	Master 2 PLB3A0_BEAR Address Lock 0 Master 2 PLB3A0_BEAR is unlocked 1 Master 2 PLB3A0_BEAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	PLB3 master 3 is DMA.
13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB3A0_BESR Field Lock 0 Master 3 PLB3A0_BESR field is unlocked 1 Master 3 PLB3A0_BESR field is locked	For these fields to become locked, an error must occur while the PCI has error locking enabled: PCIC0_BRDGOPT1[PLESE] = 1.
15	ALK3	Master 3 PLB3A0_BEAR Address Lock 0 Master 3 PLB3A0_BEAR is unlocked 1 Master 3 PLB3A0_BEAR is locked	For these fields to become locked, an error must occur while the PCI has error locking enabled: PCIC0_BRDGOPT1[PLESE] = 1.
16	PTE4	Master 4 PLB Timeout Error Status 0 No master 4 timeout error 1 Master 4 timeout error	PLB3 master 4 is OPB to PLB3 bridge.
17	R/W4	Master 4 Read/Write Status 0 Master 4 error operation was a write 1 Master 4 error operation was a read	
18	FLK4	Master 4 PLB3A0_BESR Field Lock 0 Master 4 PLB3A0_BESR field is unlocked 1 Master 4 PLB3A0_BESR field is locked	For these fields to become locked, an error must occur while the MAL has error locking enabled: MAL0_CFG[PLBLE] = 1.
19	ALK4	Master 4 PLB3A0_BEAR Address Lock 0 Master 4 PLB3A0_BEAR is unlocked 1 Master 4 PLB3A0_BEAR is locked	For these fields to become locked, an error must occur while the MAL has error locking enabled: MAL0_CFG[PLBLE] = 1.
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	PLB3 master 5 is EBC.
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB3A0_BESR Field Lock 0 Master 5 PLB3A0_BESR field is unlocked 1 Master 5 PLB3A0_BESR field is locked	
23	ALK5	Master 5 PLB3A0_BEAR Address Lock 0 Master 5 PLB3A0_BEAR is unlocked 1 Master 5 PLB3A0_BEAR is locked	
24:31		Reserved	

2.1.8.3 PLB3 Error Address Register (PLB3A0_BEAR)

The read-only PLB3A0_BEAR contains the address of the access on which a bus timeout error occurred. The PCI bridge and memory access layer (MAL) masters can qualify their PLB transactions so that a bus timeout error causes PLB3A0_BEAR to become locked. Once locked, the PLB3A0_BEAR cannot be updated, if a subsequent error occurs, until all PLB3A0_BESR[FLCK*n*] fields are cleared (*n* is the master ID).

Figure 2-38. PLB3 Error Address Register (PLB3A0_BEAR)			
0:31	BEAR	Address of bus timeout error	

2.1.8.4 PLB3 Arbiter 0 Control Register (PLB3A0_ACR)

PLB3A0_ACR controls PLB arbitration priority, which is determined by PLB priority mode and PLB priority order.

Figure 2-39. PLB3 Arbiter Control Register (PLB3A0_ACR)			
0	PPM	PLB Priority Mode 0 Fixed 1 Fair	
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 4, 5 001 Masters 1, 2, 3, 4, 5, 0 010 Masters 2, 3, 4, 5, 0, 1 011 Masters 3, 4, 5, 0, 1, 2 100 Masters 4, 5, 0, 1, 2, 3 101 Masters 5, 0, 1, 2, 3, 4 110 Reserved 111 Reserved	
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	
5:31		Reserved	

The PLB3 arbiter supports fixed and fair priority schemes to handle situations when two or more PLB3 masters simultaneously present requests of the same priority. In both fixed and fair modes, the arbiter services pending requests in the order specified by PLB3A0_ACR[PPO]. Fair mode differs in that the arbiter automatically updates PLB3A0_ACR[PPO] after arbitration cycles where multiple masters requested at the same priority. The value written into PLB3A0_ACR[PPO] gives the next to highest priority-requesting master the highest priority for the next arbitration cycle. For example, if the arbiter is operating in fair mode, PLB3A0_ACR[PPO]=0b011, and masters 0, 3, and 4 request service at the same priority level, master 3 is granted the PLB. Subsequently, PLB3A0_ACR[PPO] is set to 0b100, giving master 4 the highest priority for the next arbitration cycle.

Enabling PLB3A0_ACR[HBU] maximizes the total PLB data bus throughput when there are active master requests for the PLB3 read and write data buses pending with one data bus busy and the other idle. If a primary transfer is busy using one of the data buses, a secondary request for that bus has been acknowledged, and a third request destined for the same bus is the next highest priority request a lower priority master requesting use of the other bus would be "blocked" from gaining use of the address and transfer qualifiers bus. When PLB3A0_ACR[HBU]=1 the PLB3 arbiter internally promotes the lower priority master to the level of the currently active master thus allowing the lower priority master to utilize the idle bus.

2.1.9 PLB3 to OPB Bridge Registers

The PLB3 to OPB bridge registers listed in the following table are DCRs accessed using the mfdcr and mtdcr instructions. PLB3 to OPB bridge and OPB to PLB3 bridges are identical.

Table 2-12. PLB3 to OPB Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB32OPB0_BESR0	PLB3 to OPB Bridge Error Status Register 0 (Master IDs 0, 1, 2, 3)	0x0090	R/Clear	102
PLB32OPB0_BEAR	PLB3 to OPB Bridge Error Address Register	0x0092	R	104
PLB32OPB0_REVID	PLB3 to OPB Revision ID Register	0x0093	R	104
PLB32OPB0_BESR1	PLB3 to OPB Bridge Error Status Register 1 (Master IDs 4, 5)	0x0094	R/Clear	104

PLB32OPB0_BEAR and thePLB32OPB0_BESRn registers record information on errors that occur during PLB3 master transactions that target OPB address space. When an error is detected, the address is saved in the PLB32OPB0_BEAR, and details on the type of error are logged in one of the PLB32OPB0_BESRn registers.

Unlike the other PLB3 masters, the PCI bridge and the MAL can qualify PLB3 transactions with a lock request, so that slave error status information becomes locked. The PCI bridge requests error locking for its PLB3 transfers when PCIC0_BRDGOPT1[PLESE] = 1, and the MAL issues PLB3 transfers with error locking requested when MAL0_CFG[PLBLE] = 1.

When an error occurs during a PLB3 transaction with error locking enabled and PLB32OPB0_BEAR is not already locked, the address of the error is stored in PLB32OPB0_BEAR and the address lock bit (PLB32OPB0_BESR*m*[ALK*n*]) is set for the master whose transaction caused the error. (*m* is either 0 or 1, depending on the master ID specified by *n*) At the same time, if the field lock bit is not set (PLB32OPB0_BESR*m*[FLK*n*]=0), the type of error is saved in PLB32OPB0_BESR*m*[PTE*n*, R/W*n*], and these fields can then be locked by setting PLB32OPB0_BESR*m*[FLK*n*] = 1. When software processes the error, it should clear the error status and both lock bits at the same time.

2.1.9.1 PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0)

Errors detected by the PLB3 to OPB bridge are recorded in the appropriate PLB32OPB0_BESRx register. PLB32OPB0_BESR0 records errors for master ID 0, 1, 2, and 3.

The PCI bridge and MAL can qualify PLB3 transactions so that information describing errors that occur during the transactions becomes locked. When PLB32OPB0_BESR0[FLK*n*] is set, subsequent errors cannot update PLB32OPB0_BESR0[PTE*n*, R/W*n*]. Similarly, if any PLB32OPB0_BESR0[ALK*n*] = 1, PLB32OPB0_BEAR is locked. Once locked, the PLB32OPB0_BESR0 fields associated with a master cannot be overwritten if a subsequent error occurs until the locking fields are cleared. To clear a lock, write 1 to the PLB32OPB0_BESR0[FLK*n*] and PLB32OPB0_BESR0[ALK*n*] fields that are set. Writing 0 to a lock field does not affect the field.

For PLB32OPB0_BESR0[FLK*n*, AL*n*] to become locked, an error must occur while the PCI bridge has error locking enabled (PCIC0_BRDGOPT1[PLESE] = 1).

Figure 2-40. PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	Master 0 - PLB4 to PLB3
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3:4		Reserved	
5:6	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	Master 1 - Media Access Layer (MAL)
7	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
8	FLK1	PLB32OPB0_BESR0 Field Lock Master 1 0 Master 1 PLB32OPB0_BESR0 field is unlocked 1 Master 1 PLB32OPB0_BESR0 field is locked	
9	ALK1	PLB32OPB0_BEAR Address Lock Master 1 0 Master 1 PLB32OPB0_BEAR address is unlocked 1 Master 1 PLB32OPB0_BEAR address is locked	
10:11	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 - PCI Bridge
12	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
13	FLK2	PLB32OPB0_BESR0 Field Lock Master 2 0 Master 2 PLB32OPB0_BESR0 field is unlocked 1 Master 2 PLB32OPB0_BESR0 field is locked	
14	ALK2	PLB32OPB0_BEAR Address Lock Master 2 0 Master 2 PLB32OPB0_BEAR address is unlocked 1 Master 2 PLB32OPB0_BEAR address is locked	
15:16	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	Master 3 - DMA2P30
17	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
18:31		Reserved	

2.1.9.2 PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR)

The read-only PLB32OPB0_BEAR reports the address of a PLB3 to OPB transfer that results in an error. Errors detected by the PLB to OPB bridge are recorded in the PLB32OPB0_BEAR, unless the associated PLB32OPB0_BESR*m*[ALK*n*] field was previously set. (*m* is either 0 or 1, depending on the master ID specified by *n*. For master IDs 0, 1, 2, and 3, *m* = 0; for master IDs 4 and 5 *m* = 1.) An address lock field is set when a PLB master requests error locking, and the PLB3 to OPB Bridge detects an error during transaction. The only masters that can request error locking are the PCI Bridge and the MAL. Once locked, the PLB3 to OPB bridge cannot write PLB32OPB0_BEAR until all PLB32OPB0_BESR*m*[ALK*n*] fields that are set are cleared.

Figure 2-41. PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR)

0:31	BEAR	Address of bus error	

2.1.9.3 PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID)

The PLB32OPB0_REVID register described in the following figure is a 32-bit read-only register that contains the revision ID of the PLB3 to OPB Bridge. The contents of the register can be accessed by using the move from device control register (mfdcr) instruction. The register is not affected by reset.

Figure 2-42. PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID)

0:27		Reserved	
28:31	RID	This value is the revision level for PLB32OPB0.	

2.1.9.4 PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1)

Errors detected by the PLB3 to OPB bridge are recorded in the appropriate PLB32OPB0_BESRx register. PLB32OPB0_BESR1 records errors for master IDs 4 and 3.

Figure 2-43. PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1)

0:1	PTE4	PLB3 Timeout Error Status Master 4 00 No Master 4 error occurred 01 Master 4 timeout error occurred 10 Master 4 slave error occurred 11 Reserved	Master 4 - OPB to PLB bridge
2	R/W4	Read/Write Status Master 4 0 Master 4 error operation is a write 1 Master 4 error operation is a read	

3:4		Reserved	
5:6	PTE5	PLB3 Timeout Error Status Master 5 00 No Master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	Master 5 - External Bus Master Interface (EBMI)
7	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read	
8:31		Reserved	

2.2 On-Chip Peripheral Bus

The OPB attaches peripherals that do not have the high bandwidth or low latency requirements that would justify their direct attachment to the PLB. OPB segments interface to the PLB via the PLB to OPB bridges. These bridges allow PLB masters to access OPB slaves. Reverse bridges (OPB to PLB) allow OPB masters to access PLB slaves. PPC440EP implements two OPB segments, one attached to the PLB4 and the other attached to the PLB3.

2.2.1 OPB Features

The on-chip peripheral bus features:

- A 32-bit address bus and a 32-bit data bus
- Dynamic bus sizing; byte, halfword, and fullword transfers
- Byte and halfword duplication for byte and halfword transfers
- Single-cycle transfer of data between OPB bus master and OPB slaves
- Sequential address (burst) protocol support
- A 16-cycle fixed bus time-out provided by the OPB arbiter
- Bus parking for reduced latency
- Bus arbitration overlapped with last cycle of bus transfers

2.2.2 OPB Master Assignments

Table 2-13 lists the three masters supported by OPBA0.

Table 2-13. OPBA0 Master Assignments

OPB Agents	Description
PLB to OPB Bridge Controller	BGO (master 0)
Direct Memory Access Controller	DMA (master 1)
USB 1.1 Host Controller	USB (master 2)

2.2.3 OPB Arbiter Registers

PPC440EP implements two OPB arbiters: OPBA0 is attached to PLB3 and OPBA1 is attached to PLB4 and serves only the USB 2.0 Device. The addresses for these arbiters' priority and control registers are provided in the following table..

Table 2-14. OPB Arbiter Registers

Mnemonic	Register Name	Address	Access	Page
OPBA0_PR	OPB Arbiter Priority Register	0x0 EF60 0A00	R/W	106
OPBA0_CR	OPB Arbiter Control Register	0x0 EF60 0A01	R/W	107
OPBA1_PR	OPB Arbiter Priority Register	0x0 5000 0000	R/W	106
OPBA1_CR	OPB Arbiter Control Register	0x0 5000 0001	R/W	107

Other than for their address space, the arbiter registers for both OPBs are identical. Hence, in the following topics, they share common definitions as registers OPBAx_PR and OPBAx_CR.

2.2.3.1 OPB Arbiter Priority Register (OPBAx_PR)

The OPBAx_PR assigns priorities to the OPB master IDs.

Figure 2-44. OPB Arbiter Priority Register (OPBAx_PR)

0:1	HPM	High Priority Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	For the OPA0 arbiter, this priority is assigned to the PLB3 to OPB bridge at reset. For the OPA1 arbiter, this priority is assigned to the USB 2.0 device at reset.
2:3	MHP	Medium High Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	For the OPBA0 arbiter, this priority is assigned to the DMA2PLB3 at reset. For the OPBA1 arbiter, this field is not applicable.
4:5	MLP	Medium Low Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0GrantReserved 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	For the OPBA0 arbiter, this priority is assigned to the USB1.1 host at reset. For the OPBA1 arbiter, this field is not applicable.
6:7		Reserved	

2.2.3.2 OPB Arbiter Control Register (OPBAx_CR)

The OPBAx_CR fields controls updating of the OPBAx_PR (described in *OPB Arbiter Priority Register (OPBAx_PR)* on page 106).

Figure 2-45. OPB Arbiter Control Register (OPBAx_CR)

0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled	
1	PEN	Park Enable 0 Park disabled 1 Park enabled	
2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last	
3:4	PID	Parked Master ID 00 Master ID 0 of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID 1 of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2 of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	
5:7		Reserved	

2.2.4 OPB to PLB3 Bridge Registers

Table 2-15 lists the OPB to PLB3 bridge registers.

Table 2-15. OPB to PLB3 Bridge Registers

Mnemonic	Register Name	Address	Access	Page
OPB2PLB30_BCTRL	OPB to PLB3 Bridge Control Register	0x0A8	R/W	108
OPB2PLB30_BSTAT	OPB to PLB3 Bridge Status Register	0x0A9	R	108
OPB2PLB30_REVID	OPB to PLB3 Bridge Revision ID Register	0x0AA	R	108

2.2.4.1 OPB to PLB3 Bridge Control Register (OPB2PLB30_BCTRL)

OPB to PLB3 bridge control register (OPB2PLB30_BCTRL) contains the control bits for the OPB to PLB3 bridge. It is read from and written to via the DCR bus.

Figure 2-46. OPB to PLB3 Bridge Control Register (OPB2PLB30_BCTRL)

0:1	PRI	PLB Priority Bits. 00 Lowest 01 10 11 Highest	Used to determine the priority of OPB to PLB bridge requests on the PLB
2:31		Reserved	

2.2.4.2 OPB to PLB3 Bridge Status Register (OPB2PLB30_BSTAT)

The OPB to PLB3 bridge status register (*OPB2PLB30_BSTAT*) contains the error bit for the OPB to PLB3 bridge. To clear the status register, a "1" must be loaded into those register bits that are to be cleared. Writing a "0" to any bit in the status register does not affect the state of the bit. *Figure 2-47* shows *OPB2PLB30_BSTAT* bit definitions

Figure 2-47. OPB to PLB3 Bridge Status Register (OPB2PLB30_BSTAT)

0	ERR	Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports an error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB to PLB3 bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1:31		Reserved	

2.2.4.3 OPB to PLB3 Bridge Revision ID Register (OPB2PLB30_REVID)

The OPB to PLB3 bridge revision ID register *(OPB2PLB30_REVID)* is a read-only register. The following figure describes *OPB2PLB30_REVID* register bits.

Figure 2-48. OPB to PLB3 Bridge Revision ID Register (OPB2PLB30_REVID)

0:27		Reserved	
28:31	RID	This value indicates the revision level of the OPB-to-PLB3 bridge.	

2.3 Device Control Register (DCR) Bus

The DCR bus provides a mechanism for the PPC440EP to set up other on-chip facilities. For example, programmable resources in the DDR-SDRAM controller are configured for use with various memory devices according to their transfer characteristics and address assignments. DCR's are accessed through the use of the PowerPC mfdcr and mtdcr instructions.

The DCR bus also allows the PPC440 CPU to communicate with peripheral devices without using the PLB interface, thereby avoiding the impact to the primary system bus bandwidth, and without additional segmentation of the useable address map.

3. PLB Performance Monitor

The PLB performance monitor (PPM) in PPC440EP provides hardware for counting certain events associated with the processor local bus transactions. The performance monitor consists of a set of counters whose contents may be read by software and used to analyze and enhance PLB performance, or used as a software debug mechanism.

The PPM can perform both event-occurrence counting and event-duration counting. Occurrence counting is accomplished via a set of counters that increment their value once for each occurrence of a selected event, until a predefined timer has expired. The timer value is programmable via a cycle register and is capable of generating an external interrupt upon expiration.

Duration counting is accomplished via separate registers that increment on every clock cycle that a pre-selected event is active. Each counter can be individually enabled, and is capable of generating an external interrupt once that counter has reached its maximum value. Event selections and counter controls are performed via the control, status, and individual counter selection registers.

Note: To enhance further understanding of the PLB performance monitor function and capabilities see the Processor Local Bus architecture specifications.

Only one of either PLB3 or PLB4 can be monitored at a time. See SDR0_PFC1[P3P4E]. See also *Table 2-1* and *Table 2-2* on page 64

3.1 PPM Features

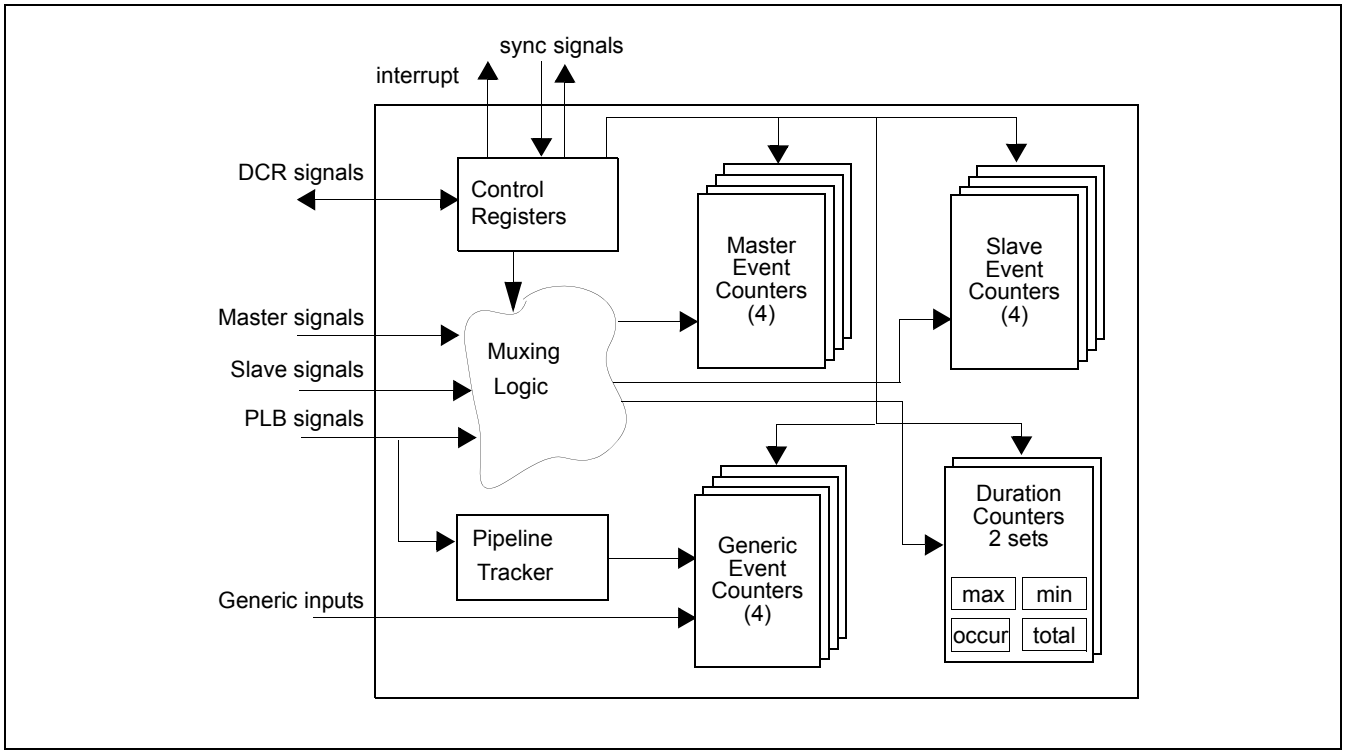
The PLB performance monitor (PPM) features:

- Selection registers to simultaneously track events from up to four selectable PLB masters, up to four PLB slaves and pipeline-stage events
- Four 24-bit master-event counters for counting occurrences of selectable PLB master events. Each master-event counter increments independently once for each PLB clock cycle that the selected event is asserted.
- Four 24-bit slave-event counters for counting occurrences of selectable PLB slave events. Each slave-event counter increments independently once for each PLB clock cycle that the selected event is asserted.
- Four 24-bit generic-event counters for counting pipeline-depth-level occurrences. A selection register allows software to choose which event(s) to monitor.
- A 32-bit cycle counter that increments on the PLB clock allows for a maximum sample period of approximately 26secs @166Mhz.
- An interrupt status register and an external interrupt signal that is triggered from any/all counter overruns.
- Master input ports connect up to eight PLB masters. In a chip implementation the PLB masters are hardwired, refer to the PPM0_MCSR0-PPM0_MCSR3 register section for more information.
- Slave input ports connect up to eight PLB slaves. In a chip implementation the PLB slaves are hardwired, refer to the PPM0_SCSR0-PPM0_SCSR3 register section for more information.
- Configuration bits for selecting priority matching, address matching, transfer-size matching, and/or read/write matching allow finger granularity when tracking certain PLB master events. Configuration bits for selecting MasterID matching also allow finer granularity when tracking certain PLB slave events.
- Programmable Address Matching Registers that allow finer granularity when tracking PLB events. If enabled, PLB events containing a PLB address matching the 64-bit contents of the Address Matching Register(s) are tracked.

- Control and Status registers allowing software full control of PPM functionality, as well as a means of monitoring the status of each counter.
- Two sets of duration counters allow concurrent tracking of up to two event-duration measurements from any of the four slaves. Event-duration measurements include counting of read, write, and/or wait tenures associated with transactions from any of four possible PLB slaves. Each event duration counter set contains an individual counter for tracking the minimum duration value, maximum duration value, total duration value, and the total event occurrences.
- Pipeline-depth tracking feature allows for independent monitoring of hits-per-pipeline depth level for both read and write data buses. PLB implementations having a read-pipeline depth no greater than four, and a write-pipeline depth no greater than two are supported.
- Power management support for low power consumption mode. When sleep-mode is desired, the PPM can be placed into pseudo-active mode where all internal logic is halted with the exception of the pipeline-depth-tracking logic, thus eliminating unnecessary power consumption.
- Additional power management feature for ultra low power consumption mode. When not used, the PPM can be placed into inactive mode, where all internal clocking is halted with the exception of the DCR register access which is used for awakening.

.

Figure 3-1. Logical Organization of PLB Performance Monitor



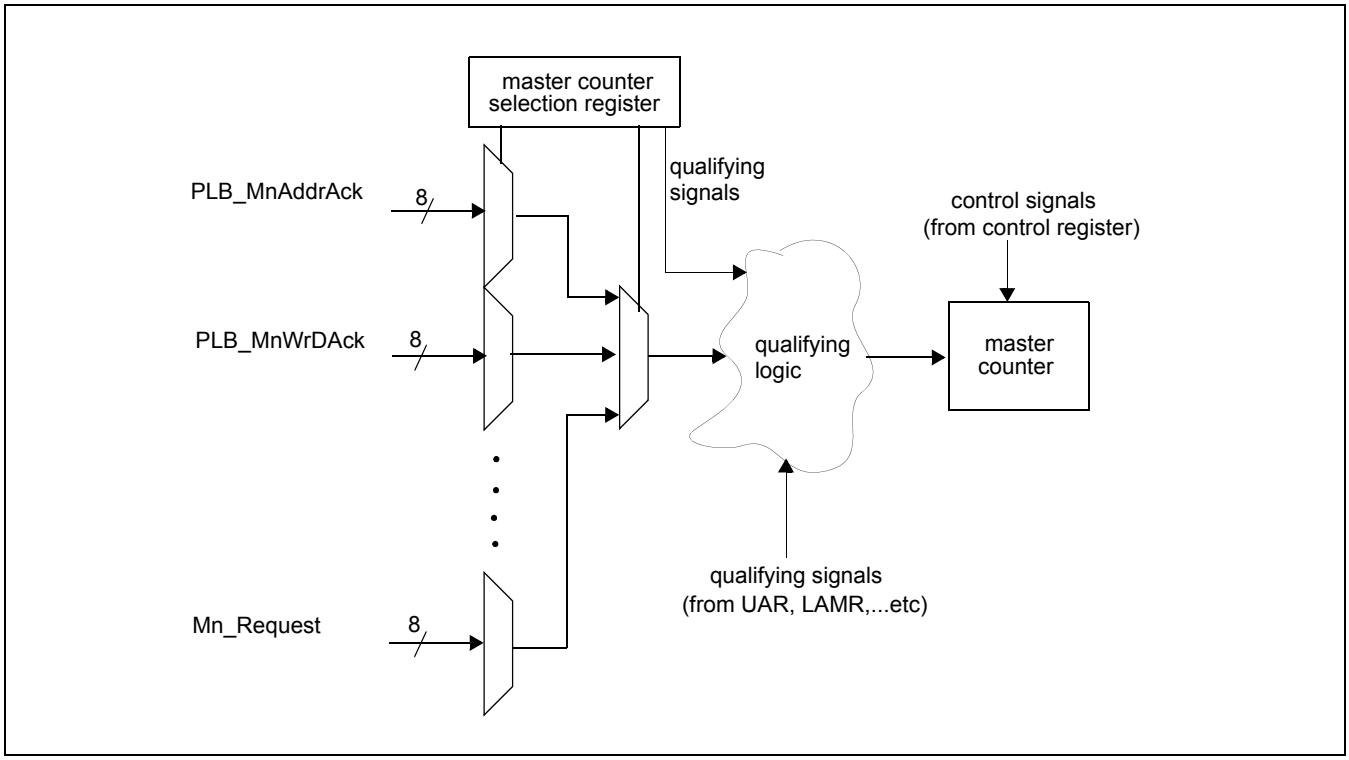
3.2 PPM Functional Description

This section provides various examples with functional block diagrams of PPM master counter logic, slave counter logic, generic counter logic, and duration counter logic.

3.2.1 Master Event Counter Logic

Various signals associated with PLB bus masters are muxed together to form the event selection capabilities for master-related event occurrence monitoring. Each master event counter is fed the set of inputs, one of which ultimately creates the trigger for the counter. The selection of the event (signal) is accomplished via an associated master event counter selection register. All master event counters are enabled, reset, and interrupt-enabled via the main control register of the PPM. *Figure 3-2* illustrates the organization of the master counter(s) logic within the PLB performance monitor.

Figure 3-2. PPM Master Event Counter Logic

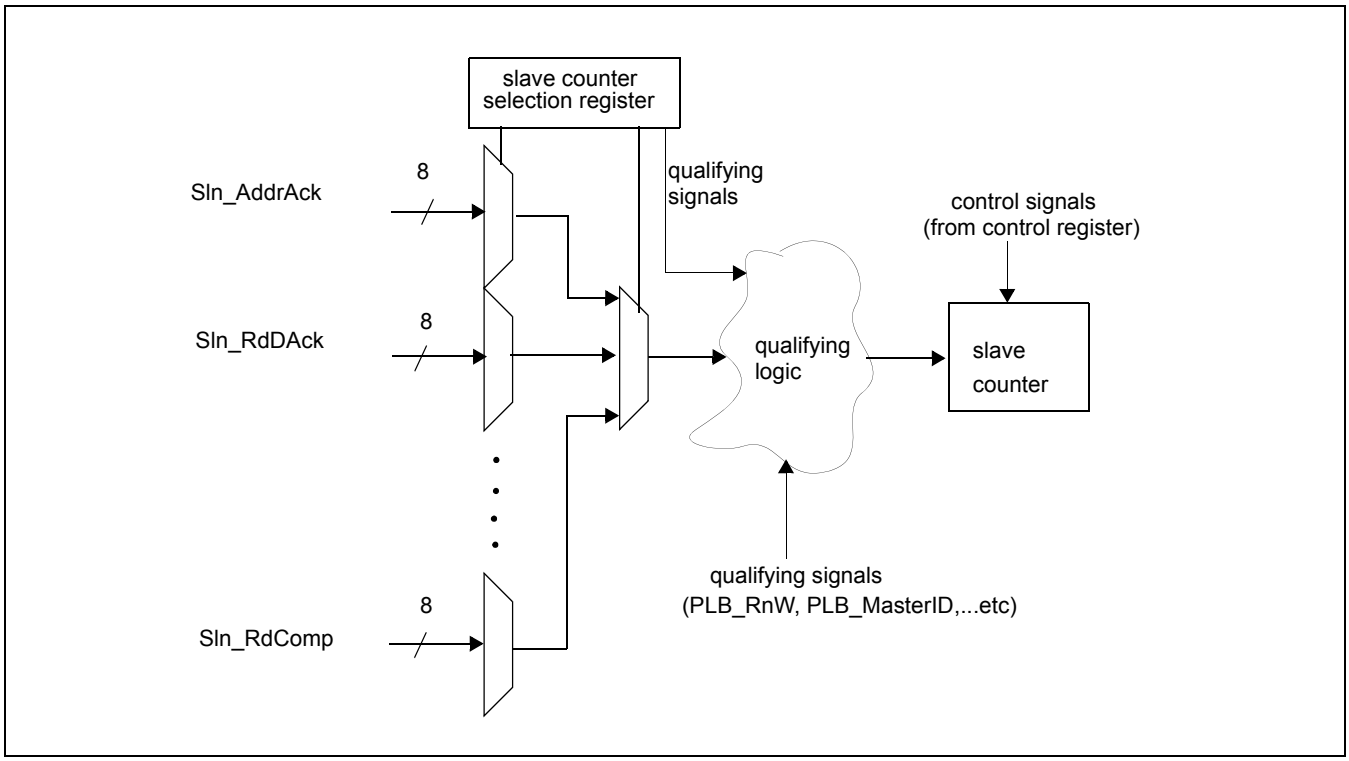


3.2.2 Slave Event Counter Logic

Various signals associated with PLB bus slaves are muxed together to form the event selection capabilities for slave-related event occurrence monitoring. Each slave event counter is fed the set of inputs, one of which ultimately creates the trigger for the counter. The selection of the event (signal) is accomplished via an associated

slave event counter selection register. All slave event counters are enabled, reset, and interrupt-enabled via the main control register of the PPM. *Figure 3-3* illustrates the organization of the slave event counter logic within the PLB performance monitor.

Figure 3-3. PPM Slave Event Counter Logic

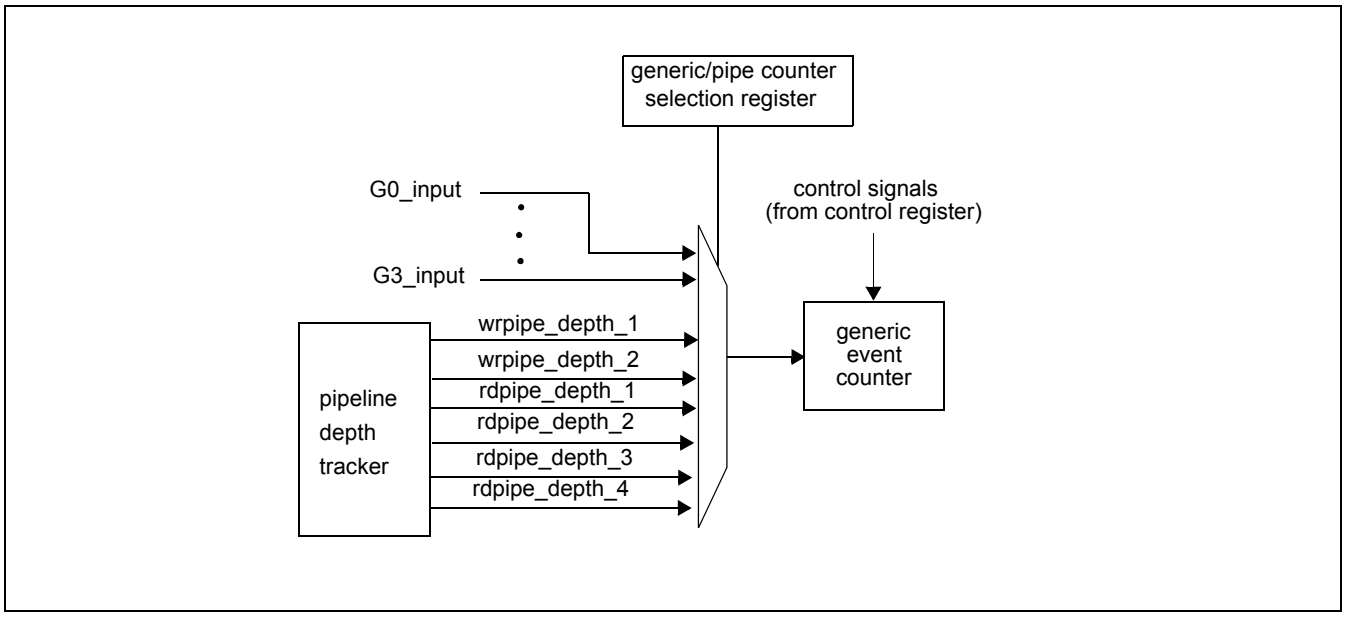


3.2.3 Generic Pipeline Event Counter Logic

Up to four external signals, possibly not associated with PLB activity, can be monitored via the generic counter logic. These signals, along with outputs of an internal PLB-pipeline-tracking logic, are muxed together to form the event selection capabilities for generic/pipeline-related event occurrence monitoring. Each generic event counter is fed the same set of inputs, one of which ultimately creates the trigger for the counter. The selection of the event

(signal) is accomplished via an associated generic counter selection register. All generic event counters are enabled, reset, and interrupt-enabled via the main control register of the PPM. *Figure 3-4* illustrates the organization of the generic event counter logic within the PLB performance monitor.

Figure 3-4. PPM Generic Counter Logic



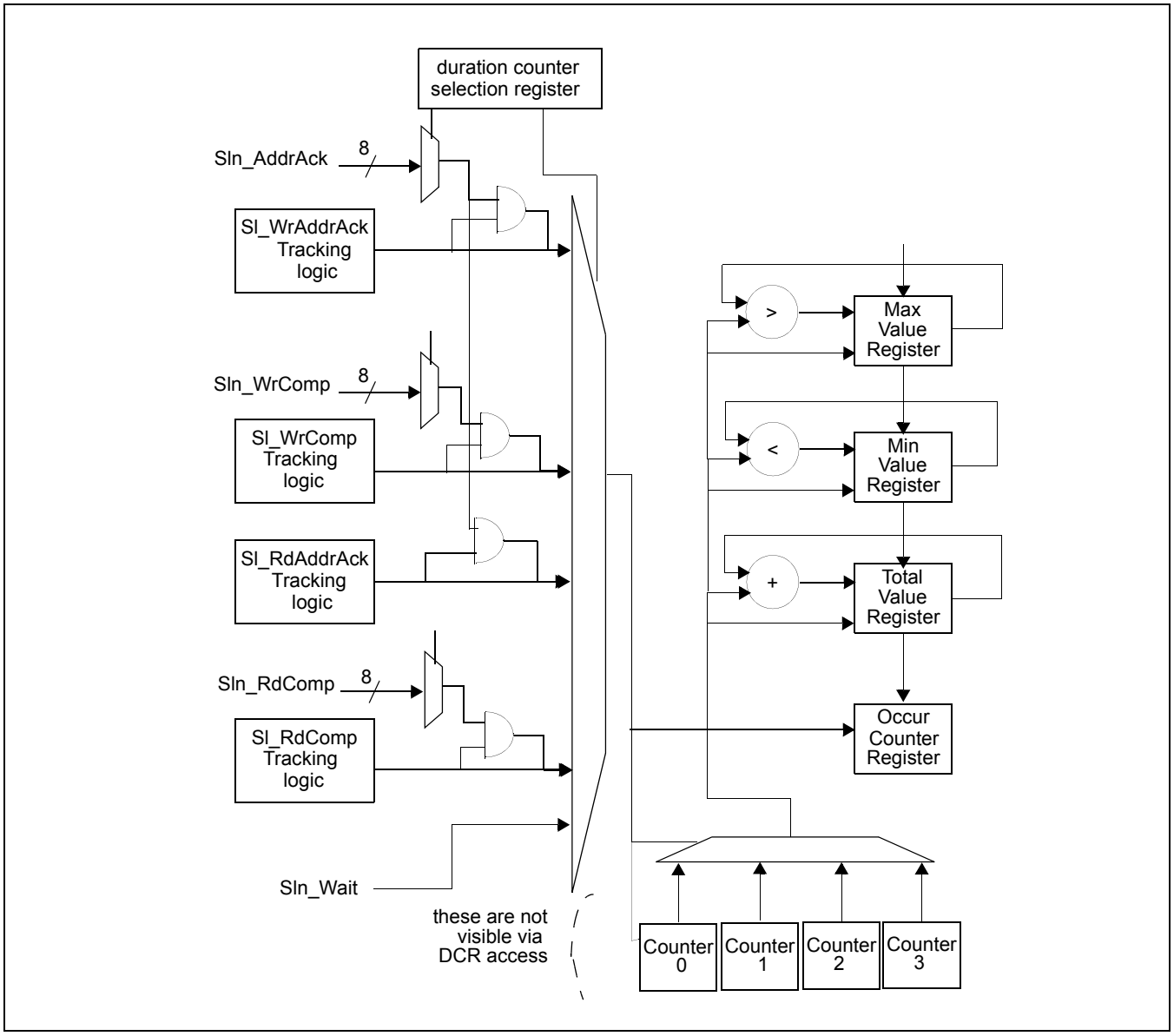
3.2.4 Duration Counter Logic

Duration monitoring is accomplished via several internal state machines, logic muxing, and counters that track slave-related tenure responses. These modules are limited to monitoring slave read data tenure, write data tenure, and wait tenure only. Duration events are selected via a duration selection register, and unlike the event occurrence counters, are tracked independent of the value of the cycle-counter register. Via separate control register(s), duration events can be selected and also whether the sampling should be a 'single-shot' or a 'running-total' of multiple occurrences.

Each duration module contains four registers that track the Min length, Max length, Total of all lengths, and total # of occurrences. This allows a user to perform some "averaging" analysis via software code. Duration measurements are enabled, reset, and interrupt-enabled via the main control register of the PPM.

Figure 3-5 illustrates the organization of the duration event counter logic within the PLB performance monitor.

Figure 3-5. PPM Duration Counter Logic



3.3 Monitoring of PLB Events

The PLB performance monitor can monitor selected events simultaneously via a set of event counters. The event definitions are described in *Table 3-1* below. Bits residing in the event selection registers allow software to select which event to count.

3.3.1 PLB Event-Occurrence Qualification

When validating PLB transactions as acceptable PPM events for monitoring, certain conditions apply. These events along with their qualifying signals and conditions are shown in *Table 3-1*.

Table 3-1. PLB Event Occurrence

PPM Event	Qualifications and Conditions
PLB_MnAddrAck (non-specific)	Signal active & SYS_plbClk (rising edge)
PLB_MnAddrAck (w/ address matching)	Signal active & SYS_plbClk (rising edge) & Masked (PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register)
PLB_MnAddrAck (w/ rd./war matching)	Signal active & SYS_plbClk (rising edge) & (RnW bit of MCounterN Selection register == Mn_RnW value)
PLB_MnAddrAck (w/ address and rd/wr matching)	Signal active & SYS_plbClk (rising edge) & Masked (PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register) & (RnW bit of MCounterN Selection register == Mn_RnW value)
PLB_MnAddrAck (w/ msize matching)	Signal active & SYS_plbClk (rising edge) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnAddrAck (w/ address and size matching)	Signal active & SYS_plbClk (rising edge) & Masked (PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnAddrAck (w/ rd/wr and size matching)	Signal active & SYS_plbClk (rising edge) &(RnW bit of MCounterN Selection register == Mn_RnW value) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnAddrAck (w/ address, rd/wr, and size matching)	Signal active & SYS_plbClk (rising edge) & Masked(PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register) & (RnW bit of MCounterN Selection register == Mn_RnW value) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnRdAck	Signal active & SYS_plbClk (rising edge)
PLB_MnWrDAck	Signal active & $\overline{\text{Mn\_Abort}}$ & SYS_plbClk(rising edge)
PLB_MnRdErr	Signal active & SYS_plbClk (rising edge)
PLB_MnWrErr	Signal active & SYS_plbClk (rising edge)
Mn_Request (non-specific)	Signal (rising_edge) & $\overline{\text{Mn\_Abort}}$ & $\overline{\text{PLB\_MnRearbitrate}}$ & $\overline{\text{PLB\_MnTimeout}}$ & SYS_plbClk (rising edge)
Mn_Request (w/ priority matching)	Signal (rising_edge) & $\overline{\text{Mn\_Abort}}$ & $\overline{\text{PLB\_MnRearbitrate}}$ & $\overline{\text{PLB\_MnTimeout}}$ & SYS_plbClk (rising edge) &PLB_MnPriority(0:1) = value of PM bits in Specified MCounterN Selection register)
Mn_Abort (non-specific)	Signal active & PLB_MnRequest & SYS_plbClk (rising edge)
Mn_Abort (w rd/wr matching)	Signal active & PLB_MnRequest & SYS_plbClk (rising edge) & (RnW bit of MCounterN Selection register == Mn_RnW value)
Mn_WrBurst	Signal active & SYS_plbClk (rising edge)
Mn_RdBurst	Signal active & SYS_plbClk (rising edge)
Sln_AddrAck (non-specific)	Signal active & SYS_plbClk (rising edge)
Sln_AddrAck (w/ MasterID matching)	Signal active & SYS_plbClk (rising edge) & (PLB_MasterID(0:2) = value of MIM bits in specified SCounterN Selection register)
Sln_RdDAck	Signal active & SYS_plbClk (rising edge)

Table 3-1. PLB Event Occurrence (continued)

PPM Event	Qualifications and Conditions
Sln_WrDAck	Signal active & SYS_plbClk (rising edge)
Sln_rearbitrate (non-specific)	Signal active & SYS_plbClk (rising edge)
Sln_rearbitrate (w/ rd/wr matching)	Signal active & SYS_plbClk (rising edge) & (RnW bit of MCounterN Selection register == Mn_RnW value)
Sln_rearbitrate (w/ PAValid)	Signal active & SYS_plbClk (rising edge) & PAValid
Sln_rearbitrate (w/ SAValid)	Signal active & SYS_plbClk (rising edge) & SAValid
Sln_rearbitrate (w/ PAValid and rd/wr matching)	Signal active & SYS_plbClk (rising edge) & PAValid & (RnW bit of MCounterN Selection register == Sln_RnW value)
Sln_rearbitrate (w/ SAValid and rd/wr matching)	Signal active & SYS_plbClk (rising edge) & SAValid & (RnW bit of MCounterN Selection register == Sln_RnW value)
Sln_wrComp	Signal active & SYS_plbClk (rising edge) & $\overline{\text{Mn\_Abort}}$
Sln_rdComp	Signal active & SYS_plbClk (rising edge) & $\overline{\text{Mn\_Abort}}$
Sln_Wait	Signal active (rising edge only) & $\overline{\text{SLn\_Addrack}}$
G0_input	Signal active & SYS_plbClk (rising edge)
G1_input	Signal active & SYS_plbClk (rising edge)
G2_input	Signal active & SYS_plbClk (rising edge)
G3_input	Signal active & SYS_plbClk (rising edge)
Pipeline_depth 1 event (write bus)	PAValid & Sln_Addrack (w/o Abort) & $\overline{\text{PLB\_RnW}}$ & SYS_plbClk (rising edge)
Pipeline_depth 2 event (write bus)	SAValid & Sln_Addrack (w/o Abort) & $\overline{\text{PLB\_ RnW}}$ & SYS_plbClk (rising edge)
Pipeline_depth_1 event (read bus)	PAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)
Pipeline_depth_2 event (read bus)	First (SAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)) without Sln_rdComp occurrence AND current pipeline-depth is level 1.
Pipeline_depth 3 event (read bus)	First (SAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)) without Sln_rdComp occurrence AND current pipeline-depth is level 2.
Pipeline_depth 4 event (read bus)	(SAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)) AND current pipeline-depth is level 3.
Note: In the above context, pipeline-depth level differs from pipeline-depth event, in that the depth level refers to the highest depth containing an outstanding plb transaction, while the depth event refers to the depth in which a newly accepted plb transaction is placed.	

3.3.2 PLB Event-Duration Measurements

If event-duration monitoring is desired, the Duration counters can be individually enabled to perform this function. *Table 3-2* shows the different types of event-duration measurements that the PPM can be programmed to monitor.

Note: The duration counters are programmable to operate in "single-shot" mode, where the counter will stop incrementing its value once the desired event has been captured, or "running total" mode, where the counters will continually start and stop counting on each event occurrence. All duration counters that have been programmed to

operate in the "running total" mode will continue to function, until the global start/stop duration (SSD) bit in the Control Register has been cleared, or the Cycle Counter has timed out.

Table 3-2. PLB Event Duration Measurement

Duration Measurement Type	Measurement Qualifiers	Measurements Taken
Sln Write Tenure	# of clock cycles between assertion of Sln_AddrAck(qualified) and assertion of Sln_WrComp (qualified) signal	(1) Total clock cycle count(s), (2) Total # of occurrences (3) # of clock cycles in shortest measurement (Min), (4) # of clocks cycles in longest measurement (Max), (5) # of Sln_WrDAcks**
Sln Read Tenure	# of clock cycles between assertion of Sln_AddrAck and assertion of Sln_RdComp signal	(1) Total clock cycle count, (2) Total # of occurrences, (3) # of clock cycles in shortest measurement (Min), (4) # of clocks cycles in longest measurement (Max), (5) # of Sln_RdDAcks**
Sln Wait Tenure	(# of clock cycles between assertion and deassertion of Sln_Wait signal) & Sln_AddrAck	(1) Total clock cycle count(s), (2) Total # of occurrences, (3) # of clock cycles in shortest measurement (Min), (4) # of clocks cycles in longest measurement (Max),
Note: **This measurement is to be performed in separate event counter registers as described in *PLB Event-Occurrence Qualification* on page 117.		

3.4 PLB Performance Monitor Registers

All PPM registers are accessed using the **mtdcr** and **mfdcr** instructions. The PPM registers are accessed through the PPM0_ADDR and PPM0_DATA registers using an indirect addressing method. PPM0_CFGADDR and PPM0_CFGDATA are described in *Table 3-3*.

Table 3-3. PPM DCR Addresses

Mnemonic	DCR Address	Access	Register	Page
PPM0_CFGADDR	0x0016	R/W	PPM Address Register	121
PPM0_CFGDATA	0x0017	R/W	PPM Data Register	121

To read or write one of the indirectly accessed PPM registers, software must first write the target register address offset into the PPM0_ADDR register. The target register can then be read using **mfdcr** or written using **mtdcr** at the DCR address for PPM0_DATA. Indirectly accessed PPM registers and their address offsets are listed in *Table 3-4*.

Table 3-4. PPM Indirectly Accessed Registers

Mnemonic	DCR Address	Access	Register	Page
PPM0_ISR	0x0000	R	Interrupt Status Register	121
PPM0_CR	0x0002	R/W	Control Register	123
PPM0_CCR	0x0003	R/W	Cycle Control Register	124
PPM0_UAR	0x0004	R/W	Upper Address Register	125
PPM0_LAR	0x0005	R/W	Lower Address Register	125

Table 3-4. PPM Indirectly Accessed Registers (continued)

Mnemonic	DCR Address	Access	Register	Page
PPM0_UAMR	0x0006	R/W	Upper Address Mask Register	125
PPM0_LAMR	0x0007	R/W	Lower Address Mask Register	126
PPM0_RIDR	0x0008	R	Revision ID Register	126
PPM0_MCSR0	0x0009	R/W	Master Event Counter Selection Register 0	126
PPM0_MCSR1	0x000A	R/W	Master Event Counter Selection Register 1	126
PPM0_MCSR2	0x000B	R/W	Master Event Counter Selection Register 2	126
PPM0_MCSR3	0x000C	R/W	Master Event Counter Selection Register 3	126
PPM0_SCSR0	0x0011	R/W	Slave Event Counter Selection Register 0	128
PPM0_SCSR1	0x0012	R/W	Slave Event Counter Selection Register 1	128
PPM0_SCSR2	0x0013	R/W	Slave Event Counter Selection Register 2	128
PPM0_SCSR3	0x0014	R/W	Slave Event Counter Selection Register 3	128
PPM0_GCSR0	0x0019	R/W	Generic Event Counter Selection Register 0	130
PPM0_GCSR1	0x001A	R/W	Generic Event Counter Selection Register 1	130
PPM0_GCSR2	0x001B	R/W	Generic Event Counter Selection Register 2	130
PPM0_GCSR3	0x001C	R/W	Generic Event Counter Selection Register 3	130
PPM0_MCR0	0x001D	R/W	Master Event Counter Register 0	130
PPM0_MCR1	0x001E	R/W	Master Event Counter Register 1	130
PPM0_MCR2	0x001F	R/W	Master Event Counter Register 2	130
PPM0_MCR3	0x0020	R/W	Master Event Counter Register 3	130
PPM0_SCR0	0x0025	R/W	Slave Event Counter Register 0	130
PPM0_SCR1	0x0026	R/W	Slave Event Counter Register 1	130
PPM0_SCR2	0x0027	R/W	Slave Event Counter Register 2	130
PPM0_SCR3	0x0028	R/W	Slave Event Counter Register 3	130
PPM0_GCR0	0x002D	R/W	Generic Pipeline Event Counter Register 0	131
PPM0_GCR1	0x002E	R/W	Generic Pipeline Event Counter Register 1	131
PPM0_GCR2	0x002F	R/W	Generic Pipeline Event Counter Register 2	131
PPM0_GCR3	0x0030	R/W	Generic Pipeline Event Counter Register 3	131
PPM0_DCSR0	0x0031	R/W	Duration Event Counter Selection Register 0	131
PPM0_DCSR1	0x0032	R/W	Duration Event Counter Selection Register 1	131
PPM0_DCMXR0	0x0033	R/W	Duration Event Counter Maximum Register 0	132
PPM0_DCMXR1	0x0034	R/W	Duration Event Counter Maximum Register 1	132
PPM0_DCMNR0	0x0035	R/W	Duration Event Counter Minimum Register 0	132
PPM0_DCMNR1	0x0036	R/W	Duration Event Counter Minimum Register 1	132
PPM0_DCTVR0	0x0037	R/W	Duration Event Counter Total Value Register 0	133

Table 3-4. PPM Indirectly Accessed Registers (continued)

Mnemonic	DCR Address	Access	Register	Page
PPM0_DCTVR1	0x0038	R/W	Duration Event Counter Total Value Register 1	133
PPM0_DCOTR0	0x0039	R/W	Duration Event Counter Occurrence Total Register 0	133
PPM0_DCOTR1	0x003A	R/W	Duration Event Counter Occurrence Total Register 1	133

3.4.1 PPM Configuration Address Register (PPM0_CFGADDR)

The PPM0_CFGADDR Register is a 32-bit read/write register which holds an 8-bit offset associated with one of the PPM's configuration registers.

Figure 3-6. PPM Configuration Address Register (PPM0_CFGADDR)

0:23			
24:31	DCRA	8-bit PPM Register Offset Value	This value can range from 0x000000 to 0x7F. Its contents are used as the indirect DCR address for accessing a PPM register.

3.4.2 PPM Configuration Data Register (PPM0_CFGDATA)

The PPM0_CFGDATA Register is a 32-bit read/write register which holds a 32-bit read/write data to/from one of the PPM's configuration registers.

Figure 3-7. PPM Configuration Data Register (PPM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contain the value of the PPM register as indicated by the PPM_CFGADDR register contents.

3.4.3 Interrupt Status Register (PPM0_ISR)

The PPM_ISR is a read-only 32-bit register where PPM interrupts are captured and held until bits are intentionally reset. The only way to reset a bit in this register is to clear the corresponding counter register. The contents of the ISR can be accessed by using the **mfdcr** and **mtdcr** instructions.

The default value of this register is 0.

Figure 3-8. Interrupt Status Register (PPM0_ISR)			
0	MC0	Master Counter 0 Interrupt Status 0 Master counter 0 interrupt did not occur 1 Master counter 0 interrupt occurred	
1	MC1	Master Counter 1 Interrupt Status 0 Master counter 1 interrupt did not occur 1 Master counter 1 interrupt occurred	
2	MC2	Master Counter 2 Interrupt Status 0 Master counter 2 interrupt did not occur 1 Master counter 2 interrupt occurred	
3	MC3	Master Counter 3 Interrupt Status 0 Master counter 3 interrupt did not occur 1 Master counter 3 interrupt occurred	
4	SC0	Slave Counter 0 Interrupt Status 0 Slave counter 0 interrupt did not occur 1 Slave counter 0 interrupt occurred	
5	SC1	Slave Counter 1 Interrupt Status 0 Slave counter 1 interrupt did not occur 1 Slave counter 1 interrupt occurred	
6	SC2	Slave Counter 2 Interrupt Status 0 Slave counter 2 interrupt did not occur 1 Slave counter 2 interrupt occurred	
7	SC3	Slave Counter 3 Interrupt Status 0 Slave counter 3 interrupt did not occur 1 Slave counter 3 interrupt occurred	
8	GC0	Generic Counter 0 Interrupt Status 0 Generic counter 0 interrupt did not occur 1 Generic counter 0 interrupt occurred	
9	GC1	Generic Counter 1 Interrupt Status 0 Generic counter 1 interrupt did not occur 1 Generic counter 1 interrupt occurred	
10	GC2	Generic Counter 2 Interrupt Status 0 Generic counter 2 interrupt did not occur 1 Generic counter 2 interrupt occurred	
11	GC3	Generic Counter 3 Interrupt Status 0 Generic counter 3 interrupt did not occur 1 Generic counter 3 interrupt occurred	
12	DC0	Duration Counter 0 Interrupt Status 0 Duration counter 0 interrupt did not occur 1 Duration counter 0 interrupt occurred	
13	DC1	Duration Counter 1 Interrupt Status 0 Duration counter 1 interrupt did not occur 1 Duration counter 1 interrupt occurred	
14	CC	Cycle Counter Interrupt Status 0 Cycle counter interrupt did not occur 1 Cycle counter interrupt occurred	
15:31		Reserved	

3.4.4 Control Register (PPM0_CR)

The PPM0_CR is a 32-bit read/write register which allows full control of the PLB performance monitor. The contents of the CR can be accessed by using the **mfdcr** and **mtdcr** instructions.

This register must be the last register programmed. The default value of this register is 0.

Figure 3-9. Control Register (PPM0_CR)

0	EMC0	Enable Master Counter 0 0 Disable master counter 0. 1 Enable master counter 0.	
1	EMC1	Enable Master Counter 1 0 Disable master counter 1. 1 Enable master counter 1.	
2	EMC2	Enable Master Counter 2 0 Disable master counter 2 1 Enable master counter 2	
3	EMC3	Enable Master Counter 3 0 Disable master counter 3 1 Enable master counter 3	
4:7		Reserved	
8	ESC0	Enable Slave Counter 0 0 Disable slave counter 0 1 Enable slave counter 0	
9	ESC1	Enable Slave Counter 1 0 Disable slave counter 1 1 Enable slave counter 1	
10	ESC2	Enable Slave Counter 2 0 Disable slave counter 2 1 Enable slave counter 2	
11	ESC3	Enable Slave Counter 3 0 Disable slave counter 3 1 Enable slave counter 3	
12:14		Reserved	
15	SOS	Synchout Selector 0 Drives PPM_synchOutSSC signal with value of SSC bit and PPM_synchOutSSD signal with value of SSD bit. 1 Asserts PPM_synchOutSSC signal when master counter 0, slave counter 0, generic counter 0, or cycle counter has overflow/underrun its value. Asserts PPM_synchOurSSD signal when duration counter 0 occurrence total counter, duration counter 1 occurrence total counter, duration counter o total value counter, and duration counter 1 total value counter has overrun their contents. **Note:** The associated interrupts must be enabled for this feature to work.	
16	EGC0	Enable Generic Counter 0 0 Disable generic counter 0 1 Enable generic counter 0	

17	EGC1	Enable Generic Counter 1 0 Disable generic counter 1 1 Enable generic counter 1	
18	EGC2	Enable Generic Counter 2 0 Disable generic counter 2. 1 Enable generic counter 2.	
19	EGC3	Enable Generic Counter 3 0 Disable generic counter 3. 1 Enable generic counter 3.	
20	EDC0	Enable Duration Counter Set 0 0 Disable duration counter 0. 1 Enable duration counter 0.	
21	EDC1	Enable Duration Counter Set 1 0 Disable duration counter 1. 1 Enable duration counter 1.	
22:23	PME	Power Mode Enable 00 Normal mode 01 Low power mode 10 Ultra low power mode 11 Reserved	
24	RDC0	Reset Event Counters 0 No action. 1 Reset all DC0 counters.	
25	RDC1	Reset Event Counters 0 No action. 1 Reset all DC1 counters.	
26	SYNC	SyncIn Enable 0 Disable syncIn functionality. 1 Enable syncIn functionality.	
27	EI	Enable Interrupts 0 Disable interrupts. 1 Enable interrupts	
28	SSD	Start/Stop Duration Events 0 Stop all duration events 1 Start all duration events	
29	SSC	Start/Stop Cycle Counter 0 Stop decrementing cycle counter. 1 Start decrementing cycle counter.	
30	RCC	Reset Cycle Counters 0 No action. 1 Reset cycle counter.	
31	REC	Reset Event Counters 0 No action. 1 Reset all PPM0_MCRn, PPM0_SCRn and PPM_GCRn.	

3.4.5 Cycle Counter Register (PPM0_CCR)

The PPM0_CCR is a 32-bit read/write register which holds a time value for PPM sampling. When enabled, the content of the CCR register value is decremented once on each PLB clock cycle. The contents of the CCR register can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value for this register is 0x FFFFFF (max value).

Figure 3-10. Cycle Counter Register (PPM0_CCR)			
0:31	CCV	32-bit Clock Count Value	This value can range from 0x0h to 0xFFFFFFFF. Its contents are decremented by 1 (for every PLB clock) if the enable bit in the CR register is active. This counter will stop decrementing when a value of 0x0h reached.

3.4.6 Upper Address Register (PPM0_UAR)

The PPM0_UAR is a 32-bit register that contains a value that is compared against the upper 32 bits of the 64-bit PLB address bus. When an MCounterN is programmed to use the address-matching feature, that counter will only track events where the PLB address matches the contents of this register and the lower address register(LAR), collectively. This register is used along with the upper address mask register (UAMR), lower address register (LAR), and the lower address mask register (LAMR). The contents of the UAR can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-11. Upper Address Register (PPM0_UAR)			
0:31	UAMV	Upper Address Match Value	If enabled, this value is matched against the upper PLB address bits UAbus(0-31) of the PLB transaction event.

3.4.7 Lower Address Register (PPM0_LAR)

The LAR is a 32-bit register that contains a value that is compared against the lower 32 bits (0-31) of the 64-bit PLB address bus. When an MCounterN is programmed to use this feature (enabled by EAM bit in MCounterN Selection register(s)), that counter will only track events where the PLB address matches the contents of this register and the upper address register, collectively. This register is used along with the lower address mask register (LAMR), upper address register (UAR), and the upper address mask register (UAMR). The contents of the LAR can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-12. Lower Address Register (PPM0_LAR)			
0:31	LAMV	Lower Address Match Value	If enabled, this value is match against the upper address bits of the PLB transaction event. Its contents can range from 0x00000000 to 0xFFFFFFF.

3.4.8 Upper Address Mask Register (PPM0_UAMR)

The UAMR is a 32-bit register used to individually isolate bit-level address comparison when using the address matching feature. Each bit in this register corresponds directly with an associated bit in the UAR register. This register is used along with the UAR match register. The contents of the UAMR can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0xFFFFFFFF.

Figure 3-13. Upper Address Mask Register (PPM0_UAMR)			
0:31	UAMn	Upper Address Bit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in theUAR register is match against the same address bit-n of the PLB upper address bus.

3.4.9 Lower Address Mask Register (PPM0_LAMR)

The LAMR is a 32-bit register used to individually isolate bit-level address comparison when using the address matching feature. Each bit in this register corresponds directly with an associated bit in the LAR register. This register is used along with the LAR match register described in a previous section. The contents of the LAMR can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0x FFFFFFFF.

Figure 3-14. Lower Address Mask Register (PPM0_LAMR)			
0:31	LAMn	Lower AddressBit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the LAR register is match against the same address bit-n of the PLB lower address bus.

3.4.10 Revision ID Register (PPM0_RIDR)

The PPM0_RIDR is a 32-bit read-only register used to identify the current design level implemented for PPM.

Figure 3-15. Revision ID Register (PPM0_RIDR)			
0:31	REVID	RevID value	Read Only field, where x is the Revision #. The default vaule for x in PPC440EP is 1.

3.4.11 Master Event Counter Selection Register 0:3 (PPM0_MCSR0:PPM0_MCSR3)

The MCSR0:MCSR3 are 32-bit read/write registers which allow event selections for corresponding master counters. The contents of the MCSR0:MCSR3 can be accessed by using the **mfdcr** and **mtdcr** instructions.

The default value of this register is 0.

Figure 3-16. Master Event Counter Selection Register 0:7 (PPM0_MCSR0:PPM0_MCSR7)			
0:2	MSEL	Master Selection 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Selects which master's signal transitions to count with this counter set.
3		Reserved	
4:7	ESEL	Master Event Selection 0000 Selects PLB_MnAddrAck 0001 Selects PLB_MnRdDAck 0010 Selects PLB_MnWrDAck 0011 Selects PLB_MnRdErr 0100 Selects PLB_MnWrErr 0101 Selects PLB_MnRequest 0110 Selects PLB_MnAbort 1000 Selects Mn_WrBurst 1001 Selects Mn_RdBurst 1010-1111 Reserved	Selects the master's PLB event to be tracked (refer to *Table 3-1 PLB Event Occurrence* on page 117).
8		Reserved	
9:10	PM	Priority Matching 00 Lowest priority 01 Low priority 10 High priority 11 Highest priority	Sets the priorty decode value, which will be used if EPM bit is set.
11	EPM	Enable Priority Matching 0 Disable priority matching 1 Enable priority matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0101 only. This bit is ignored for all other MES values.
12:15	SM	PLB transaction Size Matching 0000 One to 4 bytes 0001 4-word line 0010 8-word line 0011 16-word line 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Burst transfer - bytes 1001 Burst transfer - half words 1010 Burst transfer - words 1011 Burst transfer - double words 1100 Burst transfer - quad words 1101 Burst transfer - octal words 1110 Reserved 1111 Reserved	Sets the PLB transaction size decode value, which will be used during event tracking if the ESM bit is set.
16	ESM	Enable Size Matching 0 Disable transaction Size matching 1 Enable transaction Size matching	Selects the SM bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.

17	EAM	Enable Address Matching 0 Disable address matching feature 1 Enable address matching feature	Selects the contents of the UAR/LAR registers to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
18	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the transaction matching criteria for Mn_Request transactions only.
19	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 or 0110 only. This bit is ignored for all other values.
20:31		Reserved	

3.4.12 Slave Event Counter Selection Register 0:7 (PPM0_SCSR0:PPM0_SCSR3)

SCSR0:SCSR7 are 32-bit read/write registers which allow event selections for corresponding slave counters. The contents of the SCSR0:SCSR3 can be accessed by using the **mfdcr** and **mtdcr** instructions.

The default value of this register is 0.

Figure 3-17. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0:PPM0_SCSR3)

0:2	SSEL	Slave Selection 000 Selects Slave 0 001 Selects Slave 1 010 Selects Slave 2 011 Selects Slave 3 100 Selects Slave 4 101 Selects Slave 5 110 Selects Slave 6 111 Selects Slave 7	Selects which Slave's signal transitions to count with this counter set.
3		Reserved	
4:7	ESEL	Slave Event Selection 0000 Selects PLB_SlnAddrAck 0001 Selects PLB_SlnRdDAck 0010 Selects PLB_SlnWrDAck 0011 Selects PLB_SlnRearbitrate 0100 Selects Sln_Wait 0101 Selects Sln_WrComp 0110 Selects Sln_RdComp 0111-1111 Reserved	Selects the PLB slave event to be tracked.

8		Reserved	
9:11	MIM	Master ID Matching 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Sets the MasterID decode value, which will be used if EMIM bit is set.
12	EMIM	Enable Master ID Matching 0 Disable master ID matching 1 Enable master ID matching	Selects the MIM bits to be used when tracking an event **Note:** This feature is functional when SES bits=0000 only. This bit is ignored for all other SES values.
13	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the Read or Write value that will be used when ERM bit is set.
14	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the RWM bit to be used when tracking an event **Note:** This feature is functional when SES bits=0000 or 0011 only. This bit is ignored for all other SES values.
15	AVM	Address Valid Matching 0 PAValid matching 1 SAValid matching	Selects either the PAValid or SAValid signals to used when the EVM bit is set.
16	EVM	Enable Address Valid Matching 0 Disable matching 1 Enable matching	Enables the Primary of Secondary Address Valid matching feature to be used when tracking an event **Note:** This feature is functional when SES bits= 0011 only. This bit is ignored for all other SES values.
17:31		Reserved	

3.4.13 Generic Event Counter Selection Register 0:3 (PPM0_GCSR0:PPM0_GCSR3)

The GCSR0:GCSR3 are 32-bit read/write registers which allow selections for generic events corresponding to an external device or a pipeline-depth signal corresponding to one of four levels. The contents of the GCSR0:GCSR3 can be accessed by using the **mfdcr** and **mtdcr** instructions.

The default value of this register is 0.

Figure 3-18. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0:PPM0_GCSR3)

0-3	GES	Generic Event Selection 0000 Selects G0_input 0001 Selects G1_input 0010 Selects G2_input 0011 Selects G3_input 0100 Selects write-pipeline-depth 1 0101 Selects write-pipeline-depth 2 0110 Selects read-pipeline-depth 1 0111 Selects read-pipeline-depth 2 1000 Selects read-pipeline-depth 3 1001 Selects read-pipeline-depth 4	Selects the external signal n, or the pipeline-depth n signal as input to this counter. **Note:** Using the pipeline depth selection(s) one can determine the depth of PLB pipelining that has been reached under specific applications.
4:31		Reserved	

3.4.14 Master Event Counter Register 0:3 (PPM0_MCR0:PPM0_MCR3)

MCR0:MCR3 are 32-bit read/write register which that will increment according to the operation mode as indicated in the MCounterN Selection Register. In occurrence mode, these counters will increment by 1 for each occurrence of its preselected event. The value will continue to increment until the contents of the Cycle Counter register = 0x0. In duration mode, these registers increment by 1 for each PLB clock cycle until the first pre-selected event occurs as defined in *PLB Event-Duration Measurements* on page 118. Through program control, an interrupt can be generated once the contents of this register reaches a value of 0xFFFFFF.The contents of the MCR0:MCR3 can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-19. Master Event Counter Registers 0:3 (PPM0_MCR0:PPM0_MCR3)

0:7		Reserved	
8:31	MECV	Master Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding Mcounter Selection register. This register will only be updated if its corresponding MCn enable bit in the CR register is active.

3.4.15 Slave Event Counter Register 0:3 (PPM0_SCR0:PPM0_SCR3)

SCR0:SCR3 are 32-bit read/write register that will increment by 1 for each occurrence of its event as dictated by the corresponding "SCounterN selection register". When enabled, this register will continue to increment until the contents of the Cycle Counter register = 0x0. Through program control, an interrupt can be generated once the contents of this register reaches a value of 0xFFFFFF. The contents of the SCR0:SCR3 can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-20. Slave Event Counter Registers 0:3 (PPM0_SCR0:PPM0_SCR3)			
0:7		Reserved	
8:31	SECV	Slave Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding SCounter Selection register. This register can only be updated if its corresponding SCn enable bit in the CR register is active

3.4.16 Generic Pipeline Event Counter Register 0:3 (PPM0_GCR0:PPM0_GCR3)

GCR0:GCR3 are 32-bit read/write register that will increment by 1 for each occurrence of its event as dictated by the corresponding GCounterN selection register. When enabled, this register will continue to increment until the contents of the Cycle Counter register = 0x0. The contents of the GCR0:GCR3 can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-21. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0:PPM0_GCR3)			
0:7		Reserved	
8:31	GECV	Generic Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding GCounter Selection register. This register can only be updated if its corresponding GCn enable bit in the CR register is active

3.4.17 Duration Counter Selection Register 0:1 (PPM0_DCSR0:PPM0_DCSR1)

DCSR0:DCSR1 are 32-bit read/write registers which allow event selections for corresponding duration counters. The contents of the DCSR0:DCSR1 can be accessed by using the **mfdcr** and **mtdcr** instructions.

The default value of this register is 0.

Figure 3-22. Duration Counter Selection Registers 0:1 (PPM0_DCSR0:PPM0_DCSR1)			
0	SSS	Single Shot Selection 0 Disable single shot measurement 1 Enable single shot measurement	Selects which Slave's signal transitions to count with this counter set.
1:2		Reserved	
3:4	MT	Measurement Type 00 Write tenure 01 Read tenure 10 Wait tenure 11 Reserved	Selects the type of slave duration measurement to be performed (see *Table 3-2 PLB Event Duration Measurement* on page 119).

5:7	SLS	Slave Selection 000 Selects Sl0 001 Selects Sl1 010 Selects Sl2 011 Selects Sl3 100 Selects Sl4 101 Selects Sl5 110 Selects Sl6 111 Selects Sl7	Selects the desired PLB Slave device in which the measurement is to be performed
8:31		Reserved	

3.4.18 Duration Counter Maximum Register 0:1 (PPM0_DCMXR0:PPM0_DCMXR1)

DCMXR0:DCMXR1 are 32-bit read/write registers that will contain the last calculated maximum duration for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of 0xFFFFFF. The contents of this register can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-23. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0:PPM0_DCMXR1)			
0:7		Reserved	
8:31	MXV	Highest Max value	This value can range from 0x0 to 0xFFFFFF. Its contents are the highest maximum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

3.4.19 Duration Counter Minimum Register 0:1 (PPM0_DCMNR0:PPM0_DCMNR1)

DCMNR0:DCMNR1 are 32-bit read/write registers that will contain the last calculated minimum duration for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of 0xFFFFFFF. The contents of this register can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-24. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0:PPM0_DCMNR1)			
0:7		Reserved	
8:31	MNV	Least Min value	This value can range from 0x000000 to 0xFFFFFF. Its contents are the least minimum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

3.4.20 Duration Counter Total Value Register 0:1 (PPM0_DCTVR0:PPM0_DCTVR1)

The Duration Counter Total Value Registers 0:1 (DCTVR0:DCTVR1) are 32-bit read/write registers that will contain the running total value of all durations for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of 0xFFFFFFF. The contents of this register can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-25. Duration Total Value Register 0:1 (PPMx_DCTVR0:PPM0_DCTVR1)

0:31	DTV	Duration Total Value	This value can range from 0x0 to 0xFFFFFFFF. The value is incremented in accordance with the duration values calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active.

3.4.21 Duration Counter Occurrence Total Register 0:1 (PPM0_DCOTR0:PPM0_DCOTR1)

The Duration Counter Occurrence Total Registers 0:1 (DCOTR0:DCOTR1) are 32-bit read/write registers that will contain the total number of event occurrences for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFh. The contents of this register can be accessed by using the **mfdcr** and the **mtdcr** instructions.

The default value of this register is 0.

Figure 3-26. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0:PPM0_DCOTR1)

0:7		Reserved	
8:31	DOT	Total number of event occurrences	This value can range from 0x0 to 0xFFFFFF. Its contents are incremented once for each occurrence of an event as selected in the corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

Part II. PPC440EP RISC Processor

4. Programming Model

The programming model of the PPC440EP chip corresponds to the programming model of the PPC440 CPU core. The *PPC440 Processor User's Manual* describes how the following features and operations of the processor core appear to programmers:

- Storage addressing (including data types and byte ordering)
- Registers
- Instruction classes
- Instruction set
- Branch processing
- Integer processing
- Processor control
- User and supervisor state
- Speculative access
- Synchronization

4.1 Storage Addressing

As a 32-bit implementation of the Book-E Enhanced PowerPC Architecture, the PPC440EP implements a uniform 32-bit effective address (EA) space. Effective addresses are expanded into virtual addresses and then translated to 36-bit (64GB) real addresses by the memory management unit.

The PPC440EP generates an effective address whenever it executes a storage access, branch, cache management, or translation look aside buffer (TLB) management instruction, or when it fetches the next sequential instruction.

Table 4-1. PPC440EP System Memory Address Map

Function	Sub Function	Start Address	End Address	Size
Local Memory[1]	DDR SDRAM Controller (SDRAM0)	0x0 0000 0000	0x0 3FFF FFFF	1GB
	OPBUSB Arbiter to 128-bit PLB (OPBA1)	0x0 5000 0000	0x0 5000 003F	64B
	USB2.0 Device Controller (USB2D0)	0x0 5000 0100	0x0 5000 017F	128B
EBC	External Bus Controller (EBC0)	0x0 8000 0000	0x0 9FFF FFFF	512MB
PCI	PCI Memory	0x0 A000 0000	0x0 DFFF FFFF	1GB
	PCI I/O	0x0 E800 0000	0x0 E800 FFFF	64KB
	PCI I/O	0x0 E880 0000	0x0 EBFF FFFF	56MB
	Configuration Registers (PCIC0)	0x0 EEC0 0000	0x0 EEC0 0007	8B
	PCI Interrupt Ack / Special Cycle	0x0 EED0 0000	0x0 EED0 0003	4B
	Local Configuration Registers (PCIL0)	0x0 EF40 0000	0x0 EF40 003F	64B
Internal Peripherals	General Purpose Timer (GPT0)	0x0 EF60 0000	0x0 EF60 01FF	512B
	Universal Asynchronous Receiver/Transmitter 0 (UART0)	0x0 EF60 0300	0x0 EF60 0307	8B
	Universal Asynchronous Receiver/Transmitter 1 (UART1)	0x0 EF60 0400	0x0 EF60 0407	8B
	Universal Asynchronous Receiver/Transmitter 2 (UART2)	0x0 EF60 0500	0x0 EF60 0507	8B
	Universal Asynchronous Receiver/Transmitter 3 (UART3)	0x0 EF60 0600	0x0 EF60 0607	8B
	Inter Integrated Circuit Bus 0 (IIC0)	0x0 EF60 0700	0x0 EF60 071F	32B
	Inter Integrated Circuit Bus 1 (IIC1)	0x0 EF60 0800	0x0 EF60 081F	32B
	Serial Peripheral Interface (SPI0)	0x0 EF60 0900	0x0 EF60 0906	6B
	OPB Arbiter to 64-bit PLB (OPBA0)	0x0 EF60 0A00	0x0 EF60 0A3F	64B
	General Purpose I/O Controller 0 (GPIO0)	0x0 EF60 0B00	0x0 EF60 0B7F	128B
	General Purpose I/O Controller 1 (GPIO1)	0x0 EF60 0C00	0x0 EF60 0C7F	128B
	Ethernet to PHY Bridge (ZMII0)	0x0 EF60 0D00	0x0 EF60 0D0F	16B
	Ethernet Media Access Controller 0 (EMAC0)	0x0 EF60 0E00	0x0 EF60 0EFF	256B
	Ethernet Media Access Controller 1 (EMAC1)	0x0 EF60 0F00	0x0 EF60 0FFF	256B
	USB 1.1 Host Controller (USBH0)	0x0 EF60 1000	0x0 EF60 107F	128B
EBC		0x0 F000 0000	0x0 FFDF FFFF	254MB
Boot space (EBC Bank 0 or PCI)		0x0 FFE0 0000	0x0 FFFF FFFF	2MB

Notes:
1. DDR SDRAM can be located anywhere in the Local Memory area of the memory map.
2. EBC and PCI are relocatable, but this map reflects the suggested configuration.

Table 4-2. PPC440EP Device Configuration Register Address Map

Function	Start Address	End Address	Size
Total DCR Address Space[1]	0x0000	0x0 3FF	1KW (4KB)[1]
By function:			
Clocking Power On Reset (CPR0)	0x000C	0x000D	2W
System Device Control Registers (SDR0)	0x000E	0x000F	2W
DDR Memory Controller (SDRAM0)	0x0010	0x0011	2W
External Bus Controller (EBC0)	0x0012	0x0013	2W
128-bit PLB Performance Monitor (PPM0)	0x0016	0x0017	2W
128-bit PLB to 64-bit PLB Bridge Out (P4P3BO0)	0x0020	0x002F	16W
64-bit PLB to 128-bit PLB Bridge In (P3P4BI0)	0x0030	0x003F	16W
64-bit PLB Arbiter (PLB3A0)	0x0070	0x007F	16W
128-bit PLB Arbiter (PLB4A0)	0x0080	0x008F	16W
64-bit PLB to OPB Bridge Out (P3OPB0)	0x0090	0x009F	16W
OPB to 64-bit PLB Bridge In (OPB2PLB30)	0x00A8	0x00AF	8W
Clock and Power Management (CPM0)	0x00B0	0x00B7	8W
Universal Interrupt Controller 0 (UIC0)	0x00C0	0x00CF	16W
Universal Interrupt Controller 1 (UIC1)	0x00D0	0x00DF	16W
DMA to 64-bit PLB Controller (DMA2P30)	0x0100	0x013F	64W
Ethernet MAL (MAL0)	0x0180	0x01FF	128W
128-bit PLB to OPB Bridge (PLB42OPB0))	0x0200	0x020F	16W
DMA to 128-bit PLB Controller (DMA2P40)	0x0300	0x033F	64W

Notes:

1. DCR address space is addressable with up to 10 bits (1024 or 1K unique addresses). Each unique address represents a single 32-bit (word) register. One kiloword (1024W) equals 4KB (4096 bytes).

4.1.1 Device Control Registers

DCRs may be used to control various on-chip system functions, such as the operation of on-chip buses, peripherals, and certain processor core behaviors. The DCR access instructions are mtdcr (move to device control register) and mfdcr (move from device control register), which move data between GPRs and the DCRs. Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR.

Table 4-3. PPC440EP Device Control Registers

Register	**DCR Number or Offset** (See Note)	Access	Description
Clocking and Power-on Reset (See Note)			
CPR0_CFGADDR	0x000C	R/W	Clock, Power-On, Reset Configuration Address Register
CPR0_CFGDATA	0x000D	R/W	Clock, Power-On, Reset Configuration Data Register
CPR0_CLKUPD	offset 0x0020	R/W	Clocking Update Register

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
CPR0_PLLC0	offset 0x0040	R/W	PLL Control Register
CPR0_PLLD0	offset 0x0060	R/W	PLL Divisor Register
CPR0_PRIMAD0	offset 0x0080	R/W	Primary Divisor Register A
CPR0_PRIMBD0	offset 0x00A0	R/W	Primary Divisor Register B
CPR0_OPBD0	offset 0x00C0	R/W	OPB Clock Divisor Register
CPR0_PERD0	offset 0x00E0	R/W	Peripheral Clock Divisor Register
CPR0_MALD	offset 0x0100	R/W	MAL Clock Divisor Register
CPR0_SPCID	offset 0x0120	R/W	Synchronous PCI Clock Divisor Register
CPR0_ICFG	offset 0x0140	R/W	Initial Configuration Register
System Device Control (See Note)			
SDR0_CFGADDR	0x000E	R/W	System DCR Configuration Address Register
SDR0_CFGDATA	0x000F	R/W	System DCR Configuration Data Register
SDR0_SDSTP0	offset 0x0020	R	Serial Device Strap Register 0
SDR0_SDSTP1	offset 0x0021	R	Serial Device Strap Register 1
SDR0_PINSTP	offset 0x0040	R	Pin Strapping Register
SDR0_SDCS0	offset 0x0060	R/W	Serial Device Controller Settings Register
SDR0_ECID0	offset 0x0080	R/W	Electronic Chip ID Register 0
SDR0_ECID1	offset 0x0081	R/W	Electronic Chip ID Register 1
SDR0_ECID2	offset 0x0082	R/W	Electronic Chip ID Register 2
SDR0_ECID3	offset 0x0083	R/W	Electronic Chip ID Register 3
SDR0_JTAG	offset 0x00C0	R/W	JTAG ID Register
SDR0_DDRDL0	offset 0x00E0	R/W	DDR Delay Line Register
SDR0_EBC0	offset 0x0100	R/W	EBC Configuration Register
SDR0_UART0	offset 0x0120	R/W	UART Configuration Register 0
SDR0_UART1	offset 0x0121	R/W	UART Configuration Register 1
SDR0_UART2	offset 0x0122	R/W	UART Configuration Register 2
SDR0_UART3	offset 0x0123	R/W	UART Configuration Register 3
SDR0_CP440	offset 0x0180	R/W	440CPU Control Register
SDR0_SRST0	offset 0x0200	R/W	Individual Core Reset Control Register 0
SDR0_SRST1	offset 0x0201	R/W	Individual Core Reset Control Register 1
SDR0_SLPIPE0	offset 0x0220	R/W	PLB Slave Address Pipeline Disabling Register
SDR0_AMP0	offset 0x0240	R/W	Alternate PLB4 Master Priority Register
SDR0_AMP1	offset 0x0241	R/W	Alternate PLB3 Master Priority Register
SDR0_MIRQ0	offset 0x0260	R/W	Master Interrupt Request Register 0 (PLB3)
SDR0_MALTBL	offset 0x0280	R/W	MAL Transmit Burst Length Register
SDR0_MALRBL	offset 0x02A0	R/W	MAL Receive Burst Length Register
SDR0_MALTBS	offset 0x02C0	R/W	Reserved
SDR0_MALRBS	offset 0x02E0	R/W	Reserved

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
SDR0_PCI0	offset 0x0300	R/W	PCI Control Register
SDR0_USB0	offset 0x0320	R	Universal Serial Bus Register 0
SDR0_CUST0	offset 0x4000	R/W	Register0 Reserved for Customer Use
SDR0_SDSTP2	offset 0x4001	R	Read Only Version of SDR0_CUST0
SDR0_CUST1	offset 0x4002	R/W	Register1 Reserved for Customer Use
SDR0_SDSTP3	offset 0x4003	R	Read Only Version of SDR0_CUST1
SDR0_PFC0	offset 0x4100	R/W	Pin Function Control Register 0
SDR0_PFC1	offset 0x4101	R/W	Pin Function Control Register 1
SDR0_MFR	offset 0x4300	R/W	Miscellaneous Function Register
SDR0_EMAC0RXST	offset 0x4301	R/W	EMAC0 RX Status Register
SDR0_EMAC0TXST	offset 0x4302	R/W	EMAC0 TX Status Register
SDR0_EMAC0REJCNT	offset 0x4303	R	EMAC0 RX Packet Reject Counter
SDR0_EMAC1RXST	offset 0x4304	R/W	EMAC1 RX Status Register
SDR0_EMAC1TXST	offset 0x4305	R/W	EMAC1 TX Status Register
SDR0_EMAC1REJCNT	offset 0x4306	R	EMAC1 RX Packet Reject Counter
SDR0_HSF	offset 0x4400	R/W	DDR Hardware Self Refresh Register
PLB Performance Monitor			
PPM0_CFGADDR	0x0016	R/W	Performance Monitor Configuration Address Register
PPM0_CFGDATA	0x0017	R/W	Performance Monitor Configuration Data Register
PPM0_ISR	offset 0x0000	R	Interrupt Status Register
PPM0_CR	offset 0x0002	R/W	Control Register
PPM0_CCR	offset 0x0003	R/W	Cycle Control Register
PPM0_UAR	offset 0x0004	R/W	Upper Address Register
PPM0_LAR	offset 0x0005	R/W	Lower Address Register
PPM0_UAMR	offset 0x0006	R/W	Upper Address Mask Register
PPM0_LAMR	offset 0x0007	R/W	Lower Address Mask Register
PPM0_RIDR	offset 0x0008	R	Revision ID Register
PPM0_MCSR0	offset 0x0009	R/W	Master Event Counter Selection Register 0
PPM0_MCSR1	offset 0x000A	R/W	Master Event Counter Selection Register 1
PPM0_MCSR2	offset 0x000B	R/W	Master Event Counter Selection Register 2
PPM0_MCSR3	offset 0x000C	R/W	Master Event Counter Selection Register 3
PPM0_SCSR0	offset 0x0011	R/W	Slave Event Counter Selection Register 0
PPM0_SCSR1	offset 0x0012	R/W	Slave Event Counter Selection Register 1
PPM0_SCSR2	offset 0x0013	R/W	Slave Event Counter Selection Register 2
PPM0_SCSR3	offset 0x0014	R/W	Slave Event Counter Selection Register 3
PPM0_GCSR0	offset 0x0019	R/W	Generic Event Counter Selection Register 0
PPM0_GCSR1	offset 0x001A	R/W	Generic Event Counter Selection Register 1
PPM0_GCSR2	offset 0x001B	R/W	Generic Event Counter Selection Register 2

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
PPM0_GCSR3	offset 0x001C	R/W	Generic Event Counter Selection Register 3
PPM0_MCR0	offset 0x001D	R/W	Master Event Counter Register 0
PPM0_MCR1	offset 0x001E	R/W	Master Event Counter Register 1
PPM0_MCR2	offset 0x001F	R/W	Master Event Counter Register 2
PPM0_MCR3	offset 0x0020	R/W	Master Event Counter Register 3
PPM0_SCR0	offset 0x0025	R/W	Slave Event Counter Register 0
PPM0_SCR1	offset 0x0026	R/W	Slave Event Counter Register 1
PPM0_SCR2	offset 0x0027	R/W	Slave Event Counter Register 2
PPM0_SCR3	offset 0x0028	R/W	Slave Event Counter Register 3
PPM0_GCR0	offset 0x002D	R/W	Generic Pipeline Event Counter Register 0
PPM0_GCR1	offset 0x002E	R/W	Generic Pipeline Event Counter Register 1
PPM0_GCR2	offset 0x002F	R/W	Generic Pipeline Event Counter Register 2
PPM0_GCR3	offset 0x0030	R/W	Generic Pipeline Event Counter Register 3
PPM0_DCSR0	offset 0x0031	R/W	Duration Counter Selection Register 0
PPM0_DCSR1	offset 0x0032	R/W	Duration Counter Selection Register 1
PPM0_DCMXR0	offset 0x0033	R/W	Duration Counter Maximum Register 0
PPM0_DCMXR1	offset 0x0034	R/W	Duration Counter Maximum Register 1
PPM0_DCMNR0	offset 0x0035	R/W	Duration Counter Minimum Register 0
PPM0_DCMNR1	offset 0x0036	R/W	Duration Counter Minimum Register 1
PPM0_DCTVR0	offset 0x0037	R/W	Duration Counter Total Value Register 0
PPM0_DCTVR1	offset 0x0038	R/W	Duration Counter Total Value Register 1
PPM0_DCOTR0	offset 0x0039	R/W	Duration Counter Occurrence Total Register 0
PPM0_DCOTR1	offset 0x003A	R/W	Duration Counter Occurrence Total Register 1
PLB4 to PLB3 Bridge			
P4P3BO0_BESR0	0x0020	R/Clear	PLB4 to PLB3 Bridge Error Status 0
P4P3BO0_BEARL	0x0022	R/Clear	PLB4 to PLB3 Bridge Error Address Low
P4P3BO0_BEARH	0x0023	R/Clear	PLB4 to PLB3 Bridge Error Address High
P4P3BO0_BESR1	0x0024	R/Clear	PLB4 to PLB3 Bridge Error Status 1
P4P3BO0_CFG	0x0026	R/Clear/Set	PLB4 to PLB3 Bridge Configuration
P4P3BO0_PICR	0x0027	R/Clear/Set	PLB4 to PLB3 Bridge Priority
P4P3BO0_PEIR	0x0028	R/Clear/Set	PLB4 to PLB3 Bridge Parity Error Interrupt
P4P3BO0_REVID	0x002A	R/Clear	PLB4 to PLB3 Bridge Revision ID
PLB3 to PLB4 Bridge			
P3P4BI0_BESR0	0x0030	R/Clear	PLB3 to PLB4 Bridge Error Status 0
P3P4BI0_BEARL	0x0032	R/Clear	PLB3 to PLB4 Bridge Error Address Low
P3P4BI0_BEARH	0x0033	R/Clear	PLB3 to PLB4 Bridge Error Address High
P3P4BI0_BESR1	0x0034	R/Clear	PLB3 to PLB4 Bridge Error Status 1
P3P4BI0_CFG	0x0036	R/Clear/Set	PLB3 to PLB4 Bridge Configuration

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
P3P4BI0_PICR	0x0037	R/Clear/Set	PLB3 to PLB4 Bridge Priority
P3P4BI0_PEIR	0x0038	R/Clear/Set	PLB3 to PLB4 Bridge Parity Error Interrupt
P3P4BI0_REVID	0x003A	R/Clear	PLB3 to PLB4 Bridge Revision ID
PLB3 Arbiter Registers			
PLB3A0_REVID	0x0072	R	PLB3 Arbiter Revision ID Register
PLB3A0_BESR	0x0074	R/Clear	PLB3 Bus Error Status Register
PLB3A0_BEAR	0x0076	R/W	PLB3 Bus Error Address Register
PLB3A0_ACR	0x0077	R/W	PLB3 Arbiter Control Register
PLB4 Arbiter Registers			
PLB4A0_REVID	0x0080	R	PLB4 Crossbar ID/Revision Register
PLB4A0_ACR	0x0081	R/W	PLB4A0 Arbiter Control Register
PLB4A0_ESRL	0x0082	R/Clear	PLB4A0 Error Status Register Low
	0x0083		Reserved
PLB4A0_EARL	0x0084	R	PLB4A0 Error Address Register Low
PLB4A0_EARH	0x0085	R	PLB4A0 Error Address Register High
PLB4A0_ESRL*	0x0086	Set(W)	PLB4A0 Error Status Register Low (*reserved for diagnostic use only)
PLB4A0_ESRH*	0x0087	Set(W)	PLB4A0 Error Status Register High (*reserved for diagnostic use only)
PLB4A0_CCR	0x0088	R/W	PLB4 Crossbar Control Register
PLB4A1_ACR	0x0089	R/W	PLB4A1 Arbiter Control Register
PLB4A1_ESRL	0x008A	R/Clear	PLB4A1 Error Status Register Low
	0x008B		Reserved
PLB4A1_EARL	0x008C	R	PLB4A1 Error Address Register Low
PLB4A1_EARH	0x008D	R	PLB4A1 Error Address Register High
PLB4A1_ESRL*	0x008E	Set(W)	PLB4A1 Error Status Register Low (*reserved for diagnostic use only)
PLB4A1_ESRH*	0x008F	Set(W)	PLB4A1 Error Status Register High (*reserved for diagnostic use only)
PLB3 to OPB Bridge			
PLB32OPB0_BESR0	0x0090	R/Clear	PLB 3 to OPB Bridge Error Status Register 0
PLB32OPB0_BEAR	0x0092	R	PLB 3 to OPB Bridge Error Address Register
PLB32OPB0_REVID	0x0093	R	PLB 3 to OPB Bridge Revision ID Register
PLB32OPB0_BESR1	0x0094	R/Clear	PLB 3 to OPB Bridge Error Status Register 1
OPB to PLB3 Bridge Registers			
OPB2PLB30_BCTRL	0x00A8	R/W	OPB to PLB 3 Bridge Control Register
OPB2PLB30_BSTAT	0x00A9	R	OPB to PLB 3 Bridge Status Register
OPB2PLB30_REVID	0x00AA	R	OPB to PLB 3 Bridge Revision ID Register

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
Clocking and Power Management			
CPM0_ER	0x00B0	R/W	CPM Enable Register
CPM0_FR	0x00B1	R/W	CPM Force Register
CPM0_SR	0x00B2	R/W	CPM Status Register
Universal Interrupt Controller 0			
UIC0_SR	0x00C0	R/Clear	UIC 0 Status Register
UIC0_SRS	0x00C1	W/Set	UIC 0 Status Register Set (reserved for debug only)
UIC0_ER	0x00C2	R/W	UIC 0 Enable Register
UIC0_CR	0x00C3	R/W	UIC 0 Critical Register
UIC0_PR	0x00C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x00C5	R/W	UIC 0 Triggering Register
UIC0_MSR	0x00C6	R	UIC 0 Masked Status Register
UIC0_VR	0x00C7	R	UIC 0 Vector Register
UIC0_VCR	0x00C8	W	UIC 0 Vector Configuration Register
Universal Interrupt Controller 1			
UIC1_SR	0x00D0	R/Clear	UIC 1 Status Register
UIC1_SRS	0x00D1	W/Set	UIC 1 Status Register Set (reserved for debug only)
UIC1_ER	0x00D2	R/W	UIC 1 Enable Register
UIC1_CR	0x00D3	R/W	UIC 1 Critical Register
UIC1_PR	0x00D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x00D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x00D6	R	UIC 1 Masked Status Register
UIC1_VR	0x00D7	R	UIC 1 Vector Register
UIC1_VCR	0x00D8	W	UIC 1 Vector Configuration Register
DMA to PLB3 Controller			
DMA2P30_CR0	0x0100	R/W	DMA to PLB 3 Channel Control Register 0
DMA2P30_CT0	0x0101	R/W	DMA to PLB 3 Count Register 0
DMA2P30_DA0	0x0102	R/W	DMA to PLB 3 Destination Address Register 0
DMA2P30_SA0	0x0103	R/W	DMA to PLB 3 Source Address Register 0
DMA2P30_SG0	0x0104	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 0
DMA2P30_SC0	0x0107	R/W	DMA to PLB 3 Subchannel ID Register 0
DMA2P30_CR1	0x0108	R/W	DMA to PLB 3 Channel Control Register 1
DMA2P30_CT1	0x0109	R/W	DMA to PLB 3 Count Register 1
DMA2P30_DA1	0x010A	R/W	DMA to PLB 3 Destination Address Register 1
DMA2P30_SA1	0x010B	R/W	DMA to PLB 3 Source Address Register 1
DMA2P30_SG1	0x010C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 1
DMA2P30_SC1	0x010F	R/W	DMA to PLB 3 Subchannel ID Register 1
DMA2P30_CR2	0x0110	R/W	DMA to PLB 3 Channel Control Register 2

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
DMA2P30_CT2	0x0111	R/W	DMA to PLB 3 Count Register 2
DMA2P30_DA2	0x0112	R/W	DMA to PLB 3 Destination Address Register 2
DMA2P30_SA2	0x0113	R/W	DMA to PLB 3 Source Address Register 2
DMA2P30_SG2	0x0114	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 2
DMA2P30_SC2	0x0117	R/W	DMA to PLB 3 Subchannel ID Register 2
DMA2P30_CR3	0x0118	R/W	DMA to PLB 3 Channel Control Register 3
DMA2P30_CT3	0x0119	R/W	DMA to PLB 3 Count Register 3
DMA2P30_DA3	0x011A	R/W	DMA to PLB 3 Destination Address Register 3
DMA2P30_SA3	0x011B	R/W	DMA to PLB 3 Source Address Register 3
DMA2P30_SG3	0x011C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 3
DMA2P30_SC3	0x011F	R/W	DMA to PLB 3 Subchannel ID Register 3
DMA2P30_SR	0x0120	R/W	DMA to PLB 3 Status Register
DMA2P30_SGC	0x0123	R/W	DMA to PLB 3 Scatter/Gather Command Register
DMA2P30_ADR	0x0124	R/W	DMA to PLB 3 Address Decode Register
DMA2P30_SLP	0x0125	R/W	DMA to PLB 3 Sleep Mode Register
DMA2P30_POL	0x0126	R/W	DMA to PLB 3 Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x0180	R/W	MAL Configuration Register
MAL0_ESR	0x0181	R/Clear	MAL Error Status Register
MAL0_IER	0x0182	R/W	MAL Interrupt Enable Register
MAL0_TXCASR	0x0184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x0185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x0186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x0187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_RXCASR	0x0190	R/W	MAL Rx Channel Active Register (Set)
MAL0_RXCARR	0x0191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x0192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x0193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_TXCTP0R	0x01A0	R/W	MAL Tx Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x01A1	R/W	MAL Tx Channel 1 Table Pointer Register
MAL0_TXCTP2R	0x01A2	R/W	MAL Tx Channel 2 Table Pointer Register
MAL0_TXCTP3R	0x01A3	R/W	MAL Tx Channel 3 Table Pointer Register
MAL0_RXCTP0R	0x01C0	R/W	MAL Rx Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x01C1	R/W	MAL Rx Channel 1 Table Pointer Register
MAL0_RCBS0	0x01E0	R/W	MAL Rx Channel 0 Buffer Size Register
MAL0_RCBS1	0x01E1	R/W	MAL Rx Channel 1 Buffer Size Register
PLB4 to OPB Bridge			
PLB42OPB0_BESR0	0x0200	R/Clear	PLB4 to OPB Bridge Error Status Register 0

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
PLB42OPB0_BEARL	0x0202	R	PLB4 to OPB Bridge Error Address Register
PLB42OPB0_BEARH	0x0203	R	PLB4 to OPB Bridge Error Address Register
PLB42OPB0_BESR1	0x0204	R/Clear	PLB4 to OPB Bridge Error Status Register 1
PLB42OPB0_CFG	0x0206	R/Clear	PLB4 to OPB Bridge Error Configuration Register
PLB42OPB0_LATENCY	0x0208	R/Clear	PLB4 to OPB Bridge Burst Latency Timer
PLB42OPB0_REVID	0x020A	R	PLB4 to OPB Bridge Revision ID Register
DMA to PLB4 Controller			
DMA2P40_CR0	0x0300	R/W	DMA to PLB 4 Channel Control Register 0
DMA2P40_CTC0	0x0301	R/W	DMA to PLB 4 Count and Control Register 0
DMA2P40_SAH0	0x0302	R/W	DMA to PLB 4 Source Address High Register 0
DMA2P40_SAL0	0x0303	R/W	DMA to PLB 4 Source Address Low Register 0
DMA2P40_DAH0	0x0304	R/W	DMA to PLB 4 Destination Address High Register 0
DMA2P40_DAL0	0x0305	R/W	DMA to PLB 4 Destination Address Low Register 0
DMA2P40_SGH0	0x0306	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 0
DMA2P40_SGL0	0x0307	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 0
DMA2P40_CR1	0x0308	R/W	DMA to PLB 4 Channel Control Register 1
DMA2P40_CTC1	0x0309	R/W	DMA to PLB 4 Count and Control Register 1
DMA2P40_SAH1	0x030A	R/W	DMA to PLB 4 Source Address High Register 1
DMA2P40_SAL1	0x030B	R/W	DMA to PLB 4 Source Address Low Register 1
DMA2P40_DAH1	0x030C	R/W	DMA to PLB 4 Destination Address High Register 1
DMA2P40_DAL1	0x030D	R/W	DMA to PLB 4 Destination Address Low Register 1
DMA2P40_SGH1	0x030E	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 1
DMA2P40_SGL1	0x030F	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 1
DMA2P40_CR2	0x0310	R/W	DMA to PLB 4 Channel Control Register 2
DMA2P40_CTC2	0x0311	R/W	DMA to PLB 4 Count and Control Register 2
DMA2P40_SAH2	0x0312	R/W	DMA to PLB 4 Source Address High Register 2
DMA2P40_SAL2	0x0313	R/W	DMA to PLB 4 Source Address Low Register 2
DMA2P40_DAH2	0x0314	R/W	DMA to PLB 4 Destination Address High Register 2
DMA2P40_DAL2	0x0315	R/W	DMA to PLB 4 Destination Address Low Register 2
DMA2P40_SGH2	0x0316	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 2
DMA2P40_SGL2	0x0317	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 2
DMA2P40_CR3	0x0318	R/W	DMA to PLB 4 Channel Control Register 3
DMA2P40_CTC3	0x0319	R/W	DMA to PLB 4 Count and Control Register 3
DMA2P40_SAH3	0x031A	R/W	DMA to PLB 4 Source Address High Register 3
DMA2P40_SAL3	0x031B	R/W	DMA to PLB 4 Source Address Low Register 3
DMA2P40_DAH3	0x031C	R/W	DMA to PLB 4 Destination Address High Register 3

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
DMA2P40_DAL3	0x031D	R/W	DMA to PLB 4 Destination Address Low Register 3
DMA2P40_SGH3	0x031E	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 3
DMA2P40_SGL3	0x031F	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 3
DMA2P40_SR	0x0320	R/W	DMA to PLB 4 Status Register
DMA2P40_SGC	0x0323	R/W	DMA to PLB 4 Scatter/Gather Command Register
DMA2P40_SLP	0x0325	R/W	DMA to PLB 4 Sleep Mode Register
DMA2P40_POL	0x0326	R/W	DMA to PLB 4 Polarity Configuration Register
DDR SDRAM Controller (See Note)			
SDRAM0_CFGADDR	0x0010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x0011	R/W	DDR-SDRAM Data Register
SDRAM0_BESR0	offset 0x0000	R/W	Bus Error Status Register 0
SDRAM0_BESR1	offset 0x0008	R/W	Bus Error Status Register 1
SDRAM0_BEAR	offset 0x0010	R	Bus Error Address Register
SDRAM0_MIRQ	offset 0x0011	R/W	Master Write Interrupt
SDRAM0_SLI0	offset 0x0018	R/W	PLB Slave Interface Options
SDRAM0_CFG0	offset 0x0020	R/W	DDR SDRAM Controller Options 0
SDRAM0_CFG1	offset 0x0021	R/W	DDR SDRAM Controller Options 1
SDRAM0_DEVOPT	offset 0x0022	R/W	DDR SDRAM Device Options
SDRAM0_MCSTS	offset 0x0024	R	DDR SDRAM Memory Controller Status
SDRAM0_RTR	offset 0x0030	R/W	Refresh Timer Register
SDRAM0_PMIT	offset 0x0034	R/W	Power Management Idle Timer
SDRAM0_UABBA	offset 0x0038	R/W	PLB UA Bus Base Address
SDRAM0_B0CR	offset 0x0040	R/W	DDR SDRAM Bank 0 Configuration
SDRAM0_B1CR	offset 0x0044	R/W	DDR SDRAM Bank 1 Configuration
SDRAM0_B2CR	offset 0x0048	R/W	DDR SDRAM Bank 2 Configuration
SDRAM0_B3CR	offset 0x004C	R/W	DDR SDRAM Bank 3 Configuration
SDRAM0_TR0	offset 0x0080	R/W	DDR SDRAM Timing Register 0
SDRAM0_TR1	offset 0x0081	R/W	DDR SDRAM Timing Register 1
SDRAM0_CLKTR	offset 0x0082	R/W	DDR SDRAM Clock Timing Register
SDRAM0_WDDCTR	offset 0x0083	R/W	Write Data, DQS, DM Clock Timing Register
SDRAM0_DLYCAL	offset 0x0084	R/W	Delay Line Calibration Register
SDRAM0_ECCESR	offset 0x0098	R/W	ECC Error Status
SDRAM0_CID	offset 0x00A4	R/W	Controller ID Register
SDRAM0_RID	offset 0x00A8	R	Revision ID Register
External Bus Controller (See Note)			
EBC0_CFGADDR	0x0012	R/W	Peripheral Bank Address Register
EBC0_CFGDATA	0x0013	R/W	Peripheral Bank Data Register
EBC0_B0CR	offset 0x0000	R/W	Peripheral Bank 0 Configuration Register

Table 4-3. PPC440EP Device Control Registers (continued)

Register	DCR Number or Offset (See Note)	Access	Description
EBC0_B1CR	offset 0x0001	R/W	Peripheral Bank 1 Configuration Register
EBC0_B2CR	offset 0x0002	R/W	Peripheral Bank 2 Configuration Register
EBC0_B3CR	offset 0x0003	R/W	Peripheral Bank 3 Configuration Register
EBC0_B4CR	offset 0x0004	R/W	Peripheral Bank 4 Configuration Register
EBC0_B5CR	offset 0x0005	R/W	Peripheral Bank 5 Configuration Register
EBC0_B0AP	offset 0x0010	R/W	Peripheral Bank 0 Access Parameters
EBC0_B1AP	offset 0x0011	R/W	Peripheral Bank 1 Access Parameters
EBC0_B2AP	offset 0x0012	R/W	Peripheral Bank 2 Access Parameters
EBC0_B3AP	offset 0x0013	R/W	Peripheral Bank 3 Access Parameters
EBC0_B4AP	offset 0x0014	R/W	Peripheral Bank 4 Access Parameters
EBC0_B5AP	offset 0x0015	R/W	Peripheral Bank 5 Access Parameters
EBC0_BEAR	offset 0x0020	R/W	Peripheral Bus Error Address Register
EBC0_BESR0	offset 0x0021	R/W	Peripheral Bus Error Status Register 0
EBC0_BESR1	offset 0x0022	R/W	Peripheral Bus Error Status Register 1
EBC0_CFG	offset 0x0023	R/W	External Peripheral Control Register

Note: The CPR0_, SDR0_, PPM0_, SDRAM0_ and EBC0_ register groups include both registers that are accessed through their unique DCR numbers and registers that are accessed through their offset values. Each of these groups has its uniquely-numbered xxxx_CFGADDR and xxxx_CFGDATA registers that are used to indirectly address the other registers in the group through their offset values. *Section 20.6 EBC Registers* on page 483 contains a detailed example of indirect addressing.

4.1.2 Memory Mapped Registers

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses.

Table 4-4. PPC440EP Memory Mapped Registers

Register	Address	Access	Description
OPB Arbiter 1 (attached to 128-bit PLB)			
OPBA1_PR	0x0 5000 0000	R/W	OPB Arbiter Priority Register
OPBA1_CR	0x0 5000 0001	R/W	OPB Arbiter Control Register
USB 2.0 Device			
USB2D0_INTRIN	0x0 5000 0100	R	Interrupt register for Endpoint 0 plus IN Endpoints 1 to 3
USB2D0_POWER	0x0 5000 0102	R/W	Power management register
USB2D0_FADDR	0x0 5000 0103	R/W	Function address register
USB2D0_INTRINE	0x0 5000 0104	R/W	Interrupt enable register for USB2D0_INTRIN
USB2D0_INTROUT	0x0 5000 0106	R	Interrupt register for OUT Endpoints 1 to 3
USB2D0_INTRUSBE	0x0 5000 0108	R/W	Interrupt enable register for USB2D0_INTRUSB
USB2D0_INTRUSB	0x0 5000 0109	R	Interrupt register for common USB interrupts
USB2D0_INTROUTE	0x0 5000 010A	R/W	Interrupt enable register for IntrOut
USB2D0_TSTMODE	0x0 5000 010C	R/W	Enables the USB 2.0 test modes
USB2D0_INDEX	0x0 5000 010D	R/W	Index register for selecting the Endpoint status/control registers
USB2D0_FRAME	0x0 5000 010E	R	Frame number
USB2D0_INCSR0	0x0 5000 0110	R/W	Control Status register for Endpoint 0. (Index register set to select Endpoint 0)
USB2D0_INCSR	0x0 5000 0110	R/W	Control Status register for IN Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_INMAXP	0x0 5000 0112	R/W	Maximum packet size for IN Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_OUTCSR	0x0 5000 0114	R/W	Control Status register for OUT Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_OUTMAXP	0x0 5000 0116	R/W	Maximum packet size for OUT Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_OUTCOUNT0	0x0 5000 011A	R	Number of received bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)
USB2D0_OUTCOUNT	0x0 5000 011A	R	Number of bytes in OUT Endpoint FIFO. (Index register set to select Endpoints 1–3)
PCI to 64-bit PLB Bridge			
PCIC0_CFGADDR	0x0 EEC0 0000	R/W	PCI Space Configuration Address
PCIC0_CFGDATA	0x0 EEC0 0004	R/W	PCI Space Configuration Data
PCIC0_VENDID	offset 0x8000 0000	R/W	PCI Vendor ID
PCIC0_DEVID	offset 0x8000 0002	R/W	PCI Device ID
PCIC0_CMD	offset 0x8000 0004	R/W	PCI Command
PCIC0_STATUS	offset 0x8000 0006	R/W	PCI Status

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
PCIC0_REVID	offset 0x8000 0008	R/W	PCI Revision ID
PCIC0_CLS	offset 0x8000 0009	R/W	PCI Class
PCIC0_CACHELS	offset 0x8000 000C	R	PCI Cache Line Size
PCIC0_LATTIM	offset 0x8000 000D	R/W	PCI Latency Timer
PCIC0_HDTYPE	offset 0x8000 000E	R	PCI Header Type
PCIC0_BIST	offset 0x8000 000F	R	PCI Built In Self Test Control
PCIC0_PTM1BAR	offset 0x8000 0014	R/W	PCI PTM 1 BAR
PCIC0_PTM2BAR	offset 0x8000 0018	R/W	PCI PTM 2 BAR
PCIC0_SBSYSVID	offset 0x8000 002C	R/W	PCI Subsystem Vendor ID
PCIC0_SBSYSID	offset 0x8000 002E	R/W	PCI Subsystem ID
PCIC0_CAP	offset 0x8000 0034	R	PCI Capabilities Pointer
PCIC0_INTLN	offset 0x8000 003C	R/W	PCI Interrupt Line
PCIC0_INTPN	offset 0x8000 003D	R	PCI Interrupt Pin
PCIC0_MINGNT	offset 0x8000 003E	R	PCI Minimum Grant
PCIC0_MAXLTNCY	offset 0x8000 003F	R	PCI Maximum Latency
PCIC0_ICS	offset 0x8000 0044	R/W	PCI Interrupt Control/Status
PCIC0_ERREN	offset 0x8000 0048	R/W	Error Enable
PCIC0_ERRSTS	offset 0x8000 0049	R/W	Error Status
PCIC0_BRDGOPT1	offset 0x8000 004A	R/W	PCI Bridge Options 1
PCIC0_PLBBESR0	offset 0x8000 004C	R/W	PLB Slave Error Syndrome 0
PCIC0_PLBBESR1	offset 0x8000 0050	R/W	PLB Slave Error Syndrome 1
PCIC0_PLBBEAR	offset 0x8000 0054	R/W	PLB Slave Error Address Register
PCIC0_CAPID	offset 0x8000 0058	R	Capability Identifier
PCIC0_NEXTIPTR	offset 0x8000 0059	R	Next Item Pointer
PCIC0_PMC	offset 0x8000 005A	R	Power Management Capabilities
PCIC0_PMCSR	offset 0x8000 005C	R/W	Power Management Control Status
PCIC0_PMCSRBSE	offset 0x8000 005E	R	PMCSR PCI-to-PCI Bridge Support Extensions
PCIC0_DATA	offset 0x8000 005F	R	Data
PCIC0_BRDGOPT2	offset 0x8000 0060	R/W	PCI Bridge Options 2
PCIC0_PMSCRR	offset 0x8000 0064	R/W	Power Management State Change Request Register
PCIL0_PMM0LA	0x0 EF40 0000	R/W	PMM 0 Local Address
PCIL0_PMM0MA	0x0 EF40 0004	R/W	PMM 0 Mask/Attribute
PCIL0_PMM0PCILA	0x0 EF40 0008	R/W	PMM 0 PCI Low Address
PCIL0_PMM0PCIHA	0x0 EF40 000C	R/W	PMM 0 PCI High Address
PCIL0_PMM1LA	0x0 EF40 0010	R/W	PMM 1 Local Address
PCIL0_PMM1MA	0x0 EF40 0014	R/W	PMM 1 Mask/Attribute
PCIL0_PMM1PCILA	0x0 EF40 0018	R/W	PMM 1 PCI Low Address
PCIL0_PMM1PCIHA	0x0 EF40 001C	R/W	PMM 1 PCI High Address

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
PCIL0_PMM2LA	0x0 EF40 0020	R/W	PMM 2 Local Address
PCIL0_PMM2MA	0x0 EF40 0024	R/W	PMM 2 Mask/Attribute
PCIL0_PMM2PCILA	0x0 EF40 0028	R/W	PMM 2 PCI Low Address
PCIL0_PMM2PCIHA	0x0 EF40 002C	R/W	PMM 2 PCI High Address
PCIL0_PTM1MS	0x0 EF40 0030	R/W	PTM 1 Memory Size/Attribute
PCIL0_PTM1LA	0x0 EF40 0034	R/W	PTM 1 Local Address
PCIL0_PTM2MS	0x0 EF40 0038	R/W	PTM 2 Memory Size/Attribute
PCIL0_PTM2LA	0x0 EF40 003C	R/W	PTM 2 Local Address
General Purpose Timer			
GPT0_TBC	0x0 EF60 0000	R/W	GPT Time Base Counter
GPT0_IM	0x0 EF60 0018	R/W	GPT Interrupt Mask
GPT0_ISS	0x0 EF60 001C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x0 EF60 0020	R/W	GPT Interrupt Status (Clear bits if write 1)
GPT0_IE	0x0 EF60 0024	R/W	GPT Interrupt Enable
GPT0_COMP0	0x0 EF60 0080	R/W	GPT Compare Timer 0
GPT0_COMP1	0x0 EF60 0084	R/W	GPT Compare Timer 1
GPT0_COMP2	0x0 EF60 0088	R/W	GPT Compare Timer 2
GPT0_COMP3	0x0 EF60 008C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x0 EF60 0090	R/W	GPT Compare Timer 4
GPT0_COMP5	0x0 EF60 0094	R/W	GPT Compare Timer 5
GPT0_COMP6	0x0 EF60 0098	R/W	GPT Compare Timer 6
GPT0_MASK0	0x0 EF60 00C0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x0 EF60 00C4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x0 EF60 00C8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x0 EF60 00CC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x0 EF60 00D0	R/W	GPT Compare Mask 4
GPT0_MASK5	0x0 EF60 00D4	R/W	GPT Compare Mask 5
GPT0_MASK6	0x0 EF60 00D8	R/W	GPT Compare Mask 6
GPT0_DCT0	0x0 EF60 0110	R/W	Down Count Timer
GPT0_DCIS	0x0 EF60 011C	R/W	Down Count Timer Interrupt Status
UART Port 0			
UART0_RBR	0x0 EF60 0300	R	UART 0 Receiver Buffer Register
UART0_THR		W	UART 0 Transmitter Holding Register
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB
UART0_IER	0x0 EF60 0301	R/W	UART 0 Interrupt Enable Register
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB
UART0_IIR	0x0 EF60 0302	R	UART 0 Interrupt Identification Register
UART0_FCR	0x0 EF60 0302	W	UART 0 FIFO Control Register

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
UART0_LCR	0x0 EF60 0303	R/W	UART 0 Line Control Register
UART0_MCR	0x0 EF60 0304	R/W	UART 0 Modem Control Register
UART0_LSR	0x0 EF60 0305	R/W	UART 0 Line Status Register
UART0_MSR	0x0 EF60 0306	R/W	UART 0 Modem Status Register
UART0_SCR	0x0 EF60 0307	R/W	UART 0 Scratch Register
UART Port 1			
UART1_RBR	0x0 EF60 0400	R	UART 1 Receiver Buffer Register
UART1_THR		W	UART 1 Transmitter Holding Register
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB
UART1_IER	0x0 EF60 0401	R/W	UART 1 Interrupt Enable Register
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB
UART1_IIR	0x0 EF60 0402	R	UART 1 Interrupt Identification Register
UART1_FCR	0x0 EF60 0402	W	UART 1 FIFO Control Register
UART1_LCR	0x0 EF60 0403	R/W	UART 1 Line Control Register
UART1_MCR	0x0 EF60 0404	R/W	UART 1 Modem Control Register
UART1_LSR	0x0 EF60 0405	R/W	UART 1 Line Status Register
UART1_MSR	0x0 EF60 0406	R/W	UART 1 Modem Status Register
UART1_SCR	0x0 EF60 0407	R/W	UART 1 Scratch Register
UART Port 2			
UART2_RBR	0x0 EF60 0500	R	UART 2 Receiver Buffer Register
UART2_THR		W	UART 2 Transmitter Holding Register
UART2_DLL		R/W	UART 2 Baud-rate Divisor Latch LSB
UART2_IER	0x0 EF60 0501	R/W	UART 2 Interrupt Enable Register
UART2_DLM		R/W	UART 2 Baud-rate Divisor Latch MSB
UART2_IIR	0x0 EF60 0502	R	UART 2 Interrupt Identification Register
UART2_FCR	0x0 EF60 0502	W	UART 2 FIFO Control Register
UART2_LCR	0x0 EF60 0503	R/W	UART 2 Line Control Register
UART2_MCR	0x0 EF60 0504	R/W	UART 2 Modem Control Register
UART2_LSR	0x0 EF60 0505	R/W	UART 2 Line Status Register
UART2_MSR	0x0 EF60 0506	R/W	UART 2 Modem Status Register
UART2_SCR	0x0 EF60 0507	R/W	UART 2 Scratch Register
UART Port 3			
UART3_RBR	0x0 EF60 0600	R	UART 3 Receiver Buffer Register
UART3_THR		W	UART 3 Transmitter Holding Register
UART3_DLL		R/W	UART 3 Baud-rate Divisor Latch LSB
UART3_IER	0x0 EF60 0601	R/W	UART 3 Interrupt Enable Register
UART3_DLM		R/W	UART 3 Baud-rate Divisor Latch MSB
UART3_IIR	0x0 EF60 0602	R	UART 3 Interrupt Identification Register

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
UART3_FCR	0x0 EF60 0602	W	UART 3 FIFO Control Register
UART3_LCR	0x0 EF60 0603	R/W	UART 3 Line Control Register
UART3_MCR	0x0 EF60 0604	R/W	UART 3 Modem Control Register
UART3_LSR	0x0 EF60 0605	R/W	UART 3 Line Status Register
UART3_MSR	0x0 EF60 0606	R/W	UART 3 Modem Status Register
UART3_SCR	0x0 EF60 0607	R/W	UART 3 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x0 EF60 0700	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x0 EF60 0702	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x0 EF60 0704	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x0 EF60 0705	R/W	IIC 0 High Master Address
IIC0_CNTL	0x0 EF60 0706	R/W	IIC 0 Control
IIC0_MDCNTL	0x0 EF60 0707	R/W	IIC 0 Mode Control
IIC0_STS	0x0 EF60 0708	R/W	IIC 0 Status
IIC0_EXTSTS	0x0 EF60 0709	R/W	IIC 0 Extended Status
IIC0_LSADR	0x0 EF60 070A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x0 EF60 070B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x0 EF60 070C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x0 EF60 070D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x0 EF60 070E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x0 EF60 070F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x0 EF60 0710	R/W	IIC 0 Direct Control
IIC0_INTR	0x0 EF60 0711	R	IIC 0 Interrupt
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x0 EF60 0800	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x0 EF60 0802	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x0 EF60 0804	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x0 EF60 0805	R/W	IIC 1 High Master Address
IIC1_CNTL	0x0 EF60 0806	R/W	IIC 1 Control
IIC1_MDCNTL	0x0 EF60 0807	R/W	IIC 1 Mode Control
IIC1_STS	0x0 EF60 0808	R/W	IIC 1 Status
IIC1_EXTSTS	0x0 EF60 0809	R/W	IIC 1 Extended Status
IIC1_LSADR	0x0 EF60 080A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x0 EF60 080B	R/W	IIC 1 High Slave Address
IIC1_CLKDIV	0x0 EF60 080C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x0 EF60 080D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x0 EF60 080E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x0 EF60 080F	R/W	IIC 1 Extended Control and Slave Status

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
IIC1_DIRECTCNTL	0x0 EF60 0810	R/W	IIC 1 Direct Control
IIC1_INTR	0x0 EF60 0811	R	IIC 1 Interrupt
Serial Port Interface			
SPI0_MODE	0x0 EF60 0900	R/W	SPI Mode Register
SPI0_RxD	0x0 EF60 0901	R	SPI Receive Data Register
SPI0_TxD	0x0 EF60 0902	R/W	SPI Transmit Data Register
SPI0_CR	0x0 EF60 0903	R/W	SPI Control Register
SPI0_SR	0x0 EF60 0904	R	SPI Status Register
SPI0_CDM	0x0 EF60 0906	R/W	SPI Clock Divisor Modulus Register
OPB Arbiter 0 (attached to 64-bit PLB)			
OPBA0_PR	0x0 EF60 0A00	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x0 EF60 0A01	R/W	OPB Arbiter Control Register
General-Purpose I/O 0			
GPIO0_OR	0x0 EF60 0B00	R/W	GPIO0 Output
GPIO0_TCR	0x0 EF60 0B04	R/W	GPIO0 Three-State Control Register
GPIO0_OSRL	0x0 EF60 0B08	R/W	GPIO0 Output Select Register Low
GPIO0_OSRH	0x0 EF60 0B0C	R/W	GPIO0 Output Select Register High
GPIO0_TSRL	0x0 EF60 0B10	R/W	GPIO0 Three-State Select Register Low
GPIO0_TSRH	0x0 EF60 0B14	R/W	GPIO0 Three-State Select Register High
GPIO0_ODR	0x0 EF60 0B18	R/W	GPIO0 Open Drain Register
GPIO0_IR	0x0 EF60 0B1C	R	GPIO0 Input Register
GPIO0_RR1	0x0 EF60 0B20	R/W	GPIO0 Receive Register 1
GPIO0_RR2	0x0 EF60 0B24	R/W	GPIO0 Receive Register 2
GPIO0_RR3	0x0 EF60 0B28	R/W	GPIO0 Receive Register 3
GPIO0_ISR1L	0x0 EF60 0B30	R/W	GPIO0 Input Select Register 1 Low
GPIO0_ISR1H	0x0 EF60 0B34	R/W	GPIO0 Input Select Register 1 High
GPIO0_ISR2L	0x0 EF60 0B38	R/W	GPIO0 Input Select Register 2 Low
GPIO0_ISR2H	0x0 EF60 0B3C	R/W	GPIO0 Input Select Register 2 High
GPIO0_ISR3L	0x0 EF60 0B40	R/W	GPIO0 Input Select Register 3 Low
GPIO0_ISR3H	0x0 EF60 0B44	R/W	GPIO0 Input Select Register 3 High
General-Purpose I/O 1			
GPIO1_OR	0xEF60 0C00	R/W	GPIO1 Output Register
GPIO1_TCR	0xEF60 0C04	R/W	GPIO1 Three-State Control Register
GPIO1_OSRL	0xEF60 0C08	R/W	GPIO1 Output Select Register Low
GPIO1_OSRH	0xEF60 0C0C	R/W	GPIO1 Output Select Register High
GPIO1_TSRL	0x0 EF60 0C10	R/W	GPIO1 Three-State Select Register Low
GPIO1_TSRH	0x0 EF60 0C14	R/W	GPIO1 Three-State Select Register High
GPIO1_ODR	0x0 EF60 0C18	R/W	GPIO1 Open Drain Register

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
GPIO1_IR	0x0 EF60 0C1C	R	GPIO1 Input Register
GPIO1_RR1	0x0 EF60 0C20	R/W	GPIO1 Receive Register 1
GPIO1_RR2	0x0 EF60 0C24	R/W	GPIO1 Receive Register 2
GPIO1_RR3	0x0 EF60 0C28	R/W	GPIO1 Receive Register 3
GPIO1_ISR1L	0x0 EF60 0C30	R/W	GPIO1 Input Select Register 1 Low
GPIO1_ISR1H	0x0 EF60 0C34	R/W	GPIO1 Input Select Register 1 High
GPIO1_ISR2L	0x0 EF60 0C38	R/W	GPIO1 Input Select Register 2 Low
GPIO1_ISR2H	0x0 EF60 0C3C	R/W	GPIO1 Input Select Register 2 High
GPIO1_ISR3L	0x0 EF60 0C40	R/W	GPIO1 Input Select Register 3 Low
GPIO1_ISR3H	0x0 EF60 0C44	R/W	GPIO1 Input Select Register 3 High
Ethernet to PHY Bridge (ZMII)			
ZMII0_FER	0x0 EF60 0D00	R/W	ZMII Function Enable Register
ZMII0_SSR	0x0 EF60 0D04	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x0 EF60 0D08	R/W	ZMII SMII Status Register
Ethernet MAC 0			
EMAC0_MR0	0x0 EF60 0E00	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x0 EF60 0E04	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x0 EF60 0E08	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x0 EF60 0E0C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x0 EF60 0E10	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x0 EF60 0E14	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x0 EF60 0E18	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x0 EF60 0E1C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x0 EF60 0E20	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x0 EF60 0E24	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x0 EF60 0E28	R/W	EMAC 0 VLAN TCI Register
EMAC0_PTR	0x0 EF60 0E2C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x0 EF60 0E30	R/W	EMAC 0 Individual Address Hash Table 1
EMAC0_IAHT2	0x0 EF60 0E34	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x0 EF60 0E38	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x0 EF60 0E3C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x0 EF60 0E40	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x0 EF60 0E44	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x0 EF60 0E48	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x0 EF60 0E4C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x0 EF60 0E50	R	EMAC 0 Last Source Address Low
EMAC0_LSAL	0x0 EF60 0E54	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x0 EF60 0E58	R/W	EMAC 0 Inter-Packet Gap Value Register

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
EMAC0_STACR	0x0 EF60 0E5C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x0 EF60 0E60	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x0 EF60 0E64	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x0 EF60 0E68	R	EMAC 0 Number of Octets Transmitted
EMAC0_OCRX	0x0 EF60 0E6C	R	EMAC 0 Number of Octets Received
Ethernet MAC 1			
EMAC1_MR0	0x0 EF60 0F00	R/W	EMAC 1 Mode Register 0
EMAC1_MR1	0x0 EF60 0F04	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x0 EF60 0F08	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x0 EF60 0F0C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x0 EF60 0F10	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x0 EF60 0F14	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x0 EF60 0F18	R/W	EMAC 1 Interrupt Status Enable Register
EMAC1_IAHR	0x0 EF60 0F1C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x0 EF60 0F20	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x0 EF60 0F24	R/W	EMAC 1 VLAN TPID Register
EMAC1_VTCI	0x0 EF60 0F28	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x0 EF60 0F2C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x0 EF60 0F30	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x0 EF60 0F34	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x0 EF60 0F38	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x0 EF60 0F3C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x0 EF60 0F40	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x0 EF60 0F44	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x0 EF60 0F48	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x0 EF60 0F4C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x0 EF60 0F50	R	EMAC 1 Last Source Address Low
EMAC1_LSAL	0x0 EF60 0F54	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x0 EF60 0F58	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x0 EF60 0F5C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x0 EF60 0F60	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x0 EF60 0F64	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x0 EF60 0F68	R	EMAC 1 Number of Octets Transmitted
EMAC1_OCRX	0x0 EF60 0F6C	R	EMAC 1 Number of Octets Received
USB 1.1 Host			
USBH0_REVID	0x0 EF60 1000	R	USB Revision ID Register
USBH0_CR	0x0 EF60 1004	R/W	USB Control Register
USBH0_CS	0x0 EF60 1008	R/W - R	USB Command Status Register

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
USBH0_IS	0x0 EF60 100C	R/W	USB Interrupt Status Register
USBH0_IE	0x0 EF60 1010	R/W	USB Interrupt Enable Register
USBH0_ID	0x0 EF60 1014	R/W	USB Interrupt Disable Register
USBH0_HCCA	0x0 EF60 1018	R/W	USB Host Controller Communication Area Register
USBH0_PCED	0x0 EF60 101C	R	USB Period Current Endpoint Descriptor Register
USBH0_CHED	0x0 EF60 1020	R/W	USB Control Head Endpoint Descriptor Register
USBH0_CCED	0x0 EF60 1024	R/W	USB Control Current Endpoint Descriptor Register
USBH0_BHED	0x0 EF60 1028	R/W	USB Bulk Head Endpoint Descriptor Register
USBH0_BCED	0x0 EF60 102C	R/W	USB Bulk Current Endpoint Descriptor Register
USBH0_DH	0x0 EF60 1030	R	USB Done Head Register
USBH0_FI	0x0 EF60 1034	R/W	USB Frame Interval Register
USBH0_FR	0x0 EF60 1038	R	USB Frame Remaining Register
USBH0_FN	0x0 EF60 103C	R	USB Frame Number Register
USBH0_PS	0x0 EF60 1040	R/W	USB Periodic Start Register
USBH0_LST	0x0 EF60 1044	R/W	USB Low Speed Threshold Register
USBH0_RHDA	0x0 EF60 1048	R/W -R	USB Root Hub Descriptor Register A
USBH0_RHDB	0x0 EF60 104C	R/W	USB Root Hub Descriptor Register B
USBH0_RHSR	0x0 EF60 1050	R/W - R	USB Root Hub Status Register
USBH0_RHPS1	0x0 EF60 1054	R/W	USB Root Hub Port Status Register 0
USBH0_RHPS2	0x0 EF60 1058	R/W	USB Root Hub Port Status Register 1
USBH0_OPBMC	0x0 EF60 1070	R/W	USB OPB Master Interface Control Register
Nand Flash Controller Note: Address is an offset of the EBC address			
NDFC0_CMD	0x0 xxxx 0000	R/W	NDFC Command Register
NDFC0_ADDR	0x0 xxxx 0004	R/W	NDFC Address Register
NDFC0_DATA	0x0 xxxx 0008	R/W	NDFC Data Register
NDFC0_ECC0	0x0 xxxx 0010	R	NDFC ECC Register 0
NDFC0_ECC1	0x0 xxxx 0014	R	NDFC ECC Register 1
NDFC0_ECC2	0x0 xxxx 0018	R	NDFC ECC Register 2
NDFC0_ECC3	0x0 xxxx 001C	R	NDFC ECC Register 3
NDFC0_ECC4	0x0 xxxx 0020	R	NDFC ECC Register 4
NDFC0_ECC5	0x0 xxxx 0024	R	NDFC ECC Register 5
NDFC0_ECC6	0x0 xxxx 0028	R	NDFC ECC Register 6
NDFC0_ECC7	0x0 xxxx 002C	R	NDFC ECC Register 7
NDFC0_B0CR	0x0 xxxx 0030	R/W	NDFC Bank Configuration Register 0
NDFC0_B1CR	0x0 xxxx 0034	R/W	NDFC Bank Configuration Register 1
NDFC0_B2CR	0x0 xxxx 0038	R/W	NDFC Bank Configuration Register 2
NDFC0_B3CR	0x0 xxxx 003C	R/W	NDFC Bank Configuration Register 3
NDFC0_CR	0x0 xxxx 0040	R/W	NDFC Configuration Register

Table 4-4. PPC440EP Memory Mapped Registers (continued)

Register	Address	Access	Description
NDFC0_SR	0x0 xxxx 0044	R	NDFC Status Register
NDFC0_HWCTL	0x0 xxxx 0048	R/W	NDFC Direct Hardware Control Register
NDFC0_REVID	0x0 xxxx 0050	R	NDFC Revision ID Register

4.2 Instruction Classes

PowerPC Book-E architecture defines all instructions as falling into one of the following four classes, as determined by the primary opcode (and the extended opcode, if any):

1. Defined
2. Allocated
3. Preserved
4. Reserved (illegal or nop)

Refer to the *PPC440 Processor User's Manual* for details.

5. FPU Programming Model

The programming model of the PPC440EP FPU describes how the following features and operations appear to programmers:

- Storage addressing (including storage operands, effective address calculation, and data storage addressing modes), starting on page 159
- Floating-point exceptions, starting on page 161
- Floating-point registers, starting on page 161
- Floating-point data formats, starting on page 165
- Floating-point execution models, starting on page 171
- Floating-point instructions, starting on page 175

The Book-E Enhanced PowerPC Architecture (referred to as Book-E) specifies that the floating-point unit (FPU) implements a floating-point system as defined in ANSI/IEEE Standard 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic* (referred to as IEEE 754), but the architecture requires software support to conform fully with the standard. IEEE 754 defines certain required "operations" (addition, subtraction, and so on); the term "floating-point operation" is used to refer to one of these required operations, or to the operation performed by one of the *Multiply-Add* or *Reciprocal Estimate* instructions. All floating-point operations conform to the IEEE standard, unless software sets the IEEE Mode (NI) bit to 1 in the Floating-Point Status and Control Register (FPSCR). When FPSCR[NI] = 1, floating-point operations do not necessarily conform to the IEEE standard.

Important: Before using the FPU, it must be enabled and configured in the processor MSR and CCR0 registers. Specifically, set MSR[FP] = 1 and CCR0[DAPUIB] = 0. Also, MSR[FE0, FE1] must be set as desired. Refer to the MSR and CCR0 registers in the *PPC440 Processor User's Manual* for details.

5.1 Storage Addressing

The PPC440EP FPU accesses storage in the same uniform 32-bit (4GB) effective address (EA) space as the PPC440 processor core. Effective addresses are expanded into virtual addresses and then translated to 36-bit (64GB) real addresses by the memory management unit (MMU) of the processor core.

The PPC440EP FPU generates an effective address whenever it executes a *Load/Store* instruction.

5.1.1 Storage Operands

Bytes in storage are numbered consecutively starting with 0. Each number is the address of the corresponding byte.

The data storage operands accessed by the PPC440EP FPU load/store instructions can be words (4 bytes, or 32 bits) or double words (8 bytes, or 64 bits). The address of a storage operand is the address of its first byte (that is, of its lowest-numbered byte). Byte ordering can be either big endian or little endian, as controlled by the endian (E) storage attribute.

Operand length is implicit for each scalar storage access instruction. The operand of such a scalar storage access instruction has a "natural" alignment boundary equal to the operand length. In other words, the "natural" address of an operand is an integral multiple of the operand length. A storage operand is said to be *aligned* if it is aligned at its natural boundary; otherwise, it is said to be *unaligned*.

Data storage operands for storage access instructions have the following characteristics.

Table 5-1. Data Operand Definitions

Storage Access Instruction Type	Operand Length	$A_{28:31}$ if aligned
Word	4 bytes	0bxx00
Doubleword	8 bytes	0bx000
Note: An "x" in an address bit position indicates that the bit can be 0 or 1 regardless of the state of other bits in the address.		

The alignment of the operand effective address of some storage access instructions can affect performance, and in some cases can cause an Alignment exception to occur. For such storage access instructions, the best performance is obtained when the storage operands are naturally aligned. Table 2-2 summarizes the effects of alignment on those storage access instruction types for which such effects exist. If an instruction type is not shown in the table, then there are no alignment effects for that instruction type.

Table 5-2. Alignment Effects for Storage Access Instructions

Storage Access Instruction Type	Alignment Effects
FP Load/Store Word	Alignment exception if the storage crosses a 16-byte boundary ($EA_{28:31}$ = 0b1100); otherwise, no effect
FP Load/Store Doubleword	Alignment exception if the storage crosses 16-byte boundary ($EA_{28:31}$ > 0b1000); otherwise no effect

Instruction storage operands, on the other hand, are a word, and the effective addresses calculated by branch instructions are therefore always word-aligned.

5.1.2 Effective Address Calculation

For a storage access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address of $2^{32} – 1$ (that is, the storage operand itself crosses the maximum address boundary), the result of the operation is undefined, as specified by the architecture. The PPC440EP performs the operation as if the storage operand wrapped around from the maximum effective address to effective address 0. Software, however, should not depend upon this behavior, so that can be ported to other implementations that do not handle such accesses in the same manner. Software should ensure that no data storage operands cross the maximum address boundary.

Note that because instructions are words, and because the effective addresses of instructions are always implicitly on word boundaries, an instruction storage operand cannot cross any word boundary, including the maximum address boundary.

Effective address arithmetic, which calculates the starting address for storage operands, wraps around from the maximum address to address 0, for all effective address computations except next sequential instruction fetching.

5.1.3 Data Storage Addressing Modes

The PPC440 FPU supports the following data storage addressing modes.

- Base + displacement (D-mode) addressing mode:

  The 16-bit D field is sign-extended to 32 bits and added to the contents of the GPR designated by RA, or to zero if RA = 0. The low-order 32 bits of the sum form the effective address of the data storage operand.

- Base + index (X-mode) addressing mode:

The contents of the GPR designated by RB (or the value 0 for **lswi** and **stswi**) are added to the contents of the GPR designated by RA, or to zero if RA = 0; the low-order 32 bits of the sum form the effective address of the data storage operand.

5.2 Floating-Point Exceptions

Each floating-point exception, and each category of Invalid Operation Exception, is associated with an exception bit in the FPSCR. The following floating-point exceptions are detected by the processor; the associated FPSCR fields are listed with each exception and Invalid Operation exception category:

Table 5-3. Invalid Operation Exception Categories

Category	FPSCR Field
SNaN	VXSNAN
Infinity – Infinity	VXISI
Infinity ÷ Infinity	VXIDI
Zero ÷ Zero	VXZDZ
Infinity × Zero	VXIMZ
Invalid Compare	VXVC
Software Request	VXSOFT
Invalid Square Root	VXSQRT
Invalid Integer Convert	VXCVI

- Invalid Operation Exception (VX)
- Zero Divide Exception (ZX)
- Overflow Exception (OX)
- Underflow Exception (UX)
- Inexact Exception (XI)

Each floating-point exception also has a corresponding enable bit in the FPSCR. See *Floating-Point Status and Control Register Instructions* on page 181 for descriptions of these exception and enable bits, and *Floating Point Unit Interrupts and Exceptions* on page 259 for a detailed discussion of floating-point exceptions, including the effects of the FPSCR enable bits.

5.3 Floating-Point Registers

This section provides an overview of the register implemented in the PPC440 FPU. An alphabetical summary of all registers, including bit definitions, is provided in *Register Summary* on page 811.

Certain bits in some registers are *reserved* and thus not necessarily implemented. For all registers with fields marked as reserved, these reserved fields should be written as 0 and read as *undefined*. The recommended coding practice is to perform the initial write to a register with reserved fields set to 0, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register; use logical instructions to alter defined fields, leaving reserved fields unmodified; and write the register.

5.3.1 Register Types

The PPC440EP floating point unit provides 32 Floating Point Registers (FPRs) and a Floating-Point Status and Control Register (FPSCR). The following subsections provide an overview of each register type and the instructions associated with them.

5.3.1.1 Floating-Point Registers (FPR0:FPR31)

The PPC440 FPU provides 32 Floating-Point Registers (FPRs), each 64 bits wide. In any cycle, the FPR file can read the operands for a store instruction and an arithmetic instruction, or write the data from a load instruction and the result of an arithmetic instruction.

Figure 5-1. Floating-Point Registers (FPR0:FPR31)

0:63	FPRD	Floating-Point Register data	

The FPRs are numbered FPR0:FPR31. The floating-point instruction formats provide 5-bit fields to specify the FPRs used as operands in the execution of the associated instructions.

Each FPR contains 64 bits that support the floating-point double format. All instructions that interpret the contents of an FPR as a floating-point value uses the floating-point double format for this interpretation.

The computational instructions, and the *Move* and *Select* instructions, operate on data located in FPRs and, with the exception of the *Compare* instructions, place the result value into a FPR and optionally place status information into the Condition Register (CR).

Load and store double instructions are provided that transfer 64 bits of data between storage and the FPRs with no conversion. *Load Single* instructions transfer and convert floating-point values in floating-point single format from storage to the same value in floating-point double format in the FPRs. Store single instructions are provided to transfer and convert floating-point values in floating-point double format from the FPRs to the same value in floating-point single format in storage.

Some floating-point instructions update the FPSCR and CR explicitly. Some of these instructions move data to and from an FPR to the FPSCR, or from the FPSCR to an FPR.

The computational instructions and the *Select* instruction accept values from the FPRs in double format. For single-precision arithmetic instructions, all input values must be representable in single format;. if not, the result placed into the target FPR, and the setting of status bits in the FPSCR are undefined.

5.3.1.2 Floating-Point Status and Control Register (FPSCR)

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. See *Floating-Point Status and Control Register* on page 269 for a more detailed description of the FPSCR.

Figure 5-2. Floating-Point Status and Control Register (FPSCR)

0	FX	Floating-Point Exception Summary 0 No FPSCR exception bits changed from 0 to 1. 1 At least one FPSCR exception bit changed from 0 to 1.	All floating-point instructions, except **mtfsfi** and **mtfsf**, implicitly set this field to 1 if the instruction causes any floating-point exception bits in the FPSCR to change from 0 to 1. **mcrfs**, **mtfsfi**, **mtfsf**, **mtfsb0**, and **mtfsb1** can alter this field explicitly.

1	FEX	Floating-Point Enabled Exception Summary	The OR of all the floating-point exception fields masked by their respective enable fields. **mcrfs**, **mtfsfi**, **mtfsf**, **mtfsb0**, and **mtfsb1** cannot alter this field explicitly.
2	VX	Floating-Point Invalid Operation Exception Summary	The OR of all the Invalid Operation exception fields. **mcrfs**, **mtfsfi**, **mtfsf**, **mtfsb0**, and **mtfsb1** cannot alter this field explicitly.
3	OX	Floating-Point Overflow Exception 0 A Floating-Point Overflow exception did not occur. 1 A Floating-Point Overflow exception occurred.	See *Overflow Exception* on page 265
4	UX	Floating-Point Underflow Exception 0 A Floating-Point Underflow exception did not occur. 1 A Floating-Point Underflow exception occurred.	See *Underflow Exception* on page 266
5	ZX	Floating-Point Zero Divide Exception 0 A Floating-Point Zero Divide exception did not occur. 1 A Floating-Point Zero Divide exception occurred.	See *Zero Divide Exception* on page 265
6	IX	Floating-Point Inexact Exception 0 A Floating-Point Inexact exception did not occur. 1 A Floating-Point Inexact exception occurred.	This field is a sticky version of FPSCR[FI] The following rules describe how a given instruction sets this field. If the instruction affects FPSCR[FI], the new value of this field is obtained by ORing the old value of this field with the new value of FPSCR[FI]. If the instruction does not affect FPSCR[FI], the value of this field is unchanged.
7	VXSNAN	Floating-Point Invalid Operation Exception (SNaN) 0 A Floating-Point Invalid Operation exception (VXSNAN) did not occur. 1 A Floating-Point Invalid Operation exception (VXSNAN) occurred.	See *Invalid Operation Exception* on page 263
8	VXISI	Floating-Point Invalid Operation Exception ($\infty - \infty$) 0 A Floating-Point Invalid Operation exception (VXISI) did not occur. 1 A Floating-Point Invalid Operation exception (VXISI) occurred.	See *Invalid Operation Exception* on page 263
9	VXIDI	Floating-Point Invalid Operation Exception ($\infty \div \infty$) 0 A Floating-Point Invalid Operation exception (VXIDI) did not occur. 1 A Floating-Point Invalid Operation exception (VXIDI) occurred.	See *Invalid Operation Exception* on page 263
10	VXZDZ	Floating-Point Invalid Operation Exception ($0 \div 0$) 0 A Floating-Point Invalid Operation exception (VXZDZ) did not occur. 1 A Floating-Point Invalid Operation exception (VXZDZ) occurred.	See *Invalid Operation Exception* on page 263
11	VXIMZ	Floating-Point Invalid Operation Exception ($\infty \times 0$) 0 A Floating-Point Invalid Operation exception (VXIMZ) did not occur. 1 A Floating-Point Invalid Operation exception (VXIMZ) occurred.	See *Invalid Operation Exception* on page 263
12	VXVC	Floating-Point Invalid Operation Exception (Invalid Compare) 0 A Floating-Point Invalid Operation exception (VXVC) did not occur. 1 A Floating-Point Invalid Operation exception (VXVC) occurred.	See *Invalid Operation Exception* on page 263

13	FR	Floating-Point Fraction Rounded	The last *Arithmetic* or *Rounding and Conversion* instruction either produced an inexact result during rounding or caused a disabled Overflow Exception. See *Rounding* on page 170. This bit is not sticky.
14	FI	Floating-Point Fraction Inexact	The last *Arithmetic* or *Rounding and Conversion* instruction either produced an inexact result during rounding or caused a disabled Overflow Exception. See *Rounding* on page 170. This bit is not sticky. See the definition of FPSCR[XX] regarding the relationship between FPSCR[FI] and FPSCR[XX].
15	FPRF	Floating-Point Result Flag (FPRF)	
16	FL	Floating-Point Less Than or Negative	
17	FG	Floating-Point Greater Than or Positive	
18	FE	Floating-Point Equal to Zero	
19	FU	Floating-Point Unordered or NaN	
20		Reserved	
21	VXSOFT	Floating-Point Invalid Operation Exception (Software Request) 0 A Floating-Point Invalid Operation exception (Software Request) did not occur. 1 A Floating-Point Invalid Operation exception (Software Request) occurred.	See *Invalid Operation Exception* on page 263
22	VXSQRT	Floating-Point Invalid Operation Exception (Invalid Square Root) 0 A Floating-Point Invalid Operation exception (Invalid Square Root) did not occur. 1 A Floating-Point Invalid Operation exception (Invalid Square Root) occurred.	See *Invalid Operation Exception* on page 263
23	VXCVI	Floating-Point Invalid Operation Exception (Invalid Integer Convert) 0 A Floating-Point Invalid Operation exception (Invalid Integer Convert) did not occur. 1 A Floating-Point Invalid Operation exception (Invalid Integer Convert) occurred.	See *Invalid Operation Exception* on page 263
24	VE	Floating-Point Invalid Operation Exception Enabled 0 Floating-Point Invalid Operation exceptions are disabled. 1 Floating-Point Invalid Operation exceptions are enabled.	
25	OE	Floating-Point Overflow Exception Enable 0 Floating-Point Overflow exceptions are disabled. 1 Floating-Point Overflow exceptions are enabled.	
26	UE	Floating-Point Underflow Exception Enable 0 Floating-Point Underflow exceptions are disabled. 1 Floating-Point Underflow exceptions are enabled.	
27	ZE	Floating-Point Zero Divide Exception Enable 0 Floating-Point Zero Divide exceptions are disabled. 1 Floating-Point Zero Divide exceptions are enabled.	

28	XE	Floating-Point Inexact Exception Enable 0 Floating-Point Inexact exceptions are disabled. 1 Floating-Point Inexact exceptions are enabled.	
29	NI	Floating-Point Non-IEEE Mode 0 Non-IEEE mode is disabled 1 Non-IEEE mode is enabled.	If FPSCR[NI = 1, the remaining FPSCR bits may have meanings other than those given in this document, and the results of floating-point operations need not conform to the IEEE standard. If the IEEE-conforming result of a floating-point operation would be a denormalized number, the result of that operation is 0 (with the same sign as the denormalized number) if FPSCR[NI] = 1. The behavior when FPSCR[NI] = 1 can vary from one implementation to another
30:31	RN	Floating-Point Rounding Control 00 Round to nearest 01 Round toward zero 10 Round toward +Infinity 11 Round toward –Infinity	See *Rounding* on page 170.

Programming Note: Setting FPSCR[NI] = 1 is intended to permit results to be approximate and to cause performance to be more predictable and less data-dependent than when FPSCR[NI] = 0. For example, in non-IEEE mode, 0 is returned instead of a denormalized number, and non-IEEE mode may return a large number instead of an infinity. In non-IEEE mode, an implementation should provide a means for ensuring that all results are produced without software assistance (that is, without causing an Enabled exception type Program interrupt or a Floating-Point Unimplemented Instruction exception type Program interrupt, and without invoking an "emulation assist." See *Floating Point Unit Interrupts and Exceptions* on page 259 The means may be controlled by one or more other FPSCR bits (recall that the other FPSCR bits have implementation-dependent meanings when FPSCR[NI] = 1).

5.4 Floating-Point Data Formats

Floating-point values are represented in two binary fixed-length formats. Single-precision values are represented in the 32-bit single format. Double-precision values are represented in the 64-bit double format. The single format can be used for data in storage, but cannot be stored in the FPRs. The double format can be used for data in storage and for data in the FPRs. When a floating-point value is loaded from storage using a *Load Single* instruction, it is converted to double format and placed in the target FPR. Conversely, a floating-point value stored from an FPR into storage using a Store Single instruction is converted to single format before being placed in storage.

The lengths of the exponent and the fraction fields differ between these two formats. The structure of the single and double formats are shown in *Table 5-4* and *Table 5-5*, respectively.

Table 5-4. Floating-Point Single Format

S	EXP	FRACTION	
0	1	9	31

Table 5-5. Floating-Point Double Format

S	EXP	FRACTION	
0	1	12	63

Values in floating-point format are composed of three fields:

Table 5-6. Format Fields

Field	Description
S	Sign Bit
EXP	Exponent + bias
FRACTION	Fraction

If only a portion of a floating-point data item in storage is accessed, such as with a load or store instruction for a byte or half word (or word in the case of floating-point double format), the value affected depends on whether the PowerPC Embedded system is operating with big endian or little endian byte ordering.

5.4.1 Value Representation

Representation of numeric values in the floating-point formats consists of a sign bit (S), a biased exponent (EXP), and the fraction portion (FRACTION) of the significand. The significand consists of a leading implied bit concatenated on the right with the FRACTION. This leading implied bit is 1 for normalized numbers and 0 for denormalized numbers and is located in the unit bit position (that is, the first bit to the left of the binary point). Values representable within the two floating-point formats can be specified by the parameters listed in *Table 5-7*.

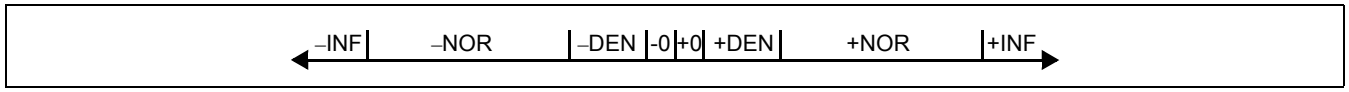
Table 5-7. IEEE 754 Floating-Point Fields

	Single	Double
Exponent Bias	+127	+1023
Maximum Exponent	+127	+1023
Minimum Exponent	–126	–1022
Field Widths (Bits)		
Sign	1	1
Exponent	8	11
Fraction	23	52
Significand	24	53

The FPRs support the floating-point double format only.

The numeric and nonnumeric values representable within each of the two supported formats are approximations to the real numbers and include the normalized numbers, denormalized numbers, and zero values. The nonnumeric values representable are the infinities and the Not a Numbers (NaNs). The infinities are adjoined to the real numbers, but are not numbers themselves, and the standard rules of arithmetic do not hold when they are used in an operation. They are related to the real numbers by order alone. It is possible, however. to define restricted operations among numbers and infinities. The relative location on the real number line for each of the defined entities is shown in *Figure 5-3*.

Figure 5-3. Approximation to Real Numbers



The NaNs are not related to the numeric values or infinities by order or value, but are encodings used to convey diagnostic information such as the representation of uninitialized variables.

The following is a description of the different floating-point values defined in the architecture:

5.4.2 Binary Floating-Point Numbers

Machine-representable values used as approximations to real numbers. Three categories of numbers are supported: normalized numbers, denormalized numbers, and zero values.

5.4.2.1 Normalized Numbers

Normalized numbers (±NOR) have an unbiased exponent value in the range:

- –126 to 127 in single format
- –1022 to 1023 in double format

They are values in which the implied unit bit is 1. Normalized numbers are interpreted as follows:

$$NOR = (-1)^s \times 2^E \times (1.fraction)$$

where s is the sign, E is the unbiased exponent, and 1.fraction is the significand, which is composed of a leading unit bit (implied bit) and a fraction part.

The ranges covered by the magnitude (M) of a normalized floating-point number are approximately equal to:

- Single Format:

  $1.2 \times 10^{-38} \leq M \leq 3.4 \times 10^{38}$

- Double Format:

  $2.2 \times 10^{-308} \leq M \leq 1.8 \times 10^{308}$

5.4.2.2 Denormalized Numbers

Denormalized numbers (±DEN) are values that have a biased exponent value of zero and a nonzero fraction value. They are nonzero numbers smaller in magnitude than the representable normalized numbers. They are values in which the implied unit bit is 0. Denormalized numbers are interpreted as follows:

$$DEN = (-1)^s \times 2^{Emin} \times (0.fraction)$$

where Emin is the minimum representable exponent value (–126 for single-precision, –1022 for double-precision).

5.4.2.3 Zero Values

Zero values (±0) have a biased exponent value of zero and a fraction value of zero. Zeros can have a positive or negative sign. The sign of zero is ignored by comparison operations; comparison treats +0 as equal to –0).

5.4.3 Infinities

Infinities (±∞)are values that have the maximum biased exponent value:

- 255 in single format
- 2047 in double format

and a zero fraction value. They are used to approximate values greater in magnitude than the maximum normalized value.

Infinity arithmetic is defined as the limiting case of real arithmetic, with restricted operations defined among numbers and infinities. Infinities and the real numbers can be related by ordering in the affine sense:

$-\infty$ < every finite number < $+\infty$

Arithmetic on infinities is always exact and does not signal any exception, except when an exception occurs due to the invalid operations as described in *Invalid Operation Exception* on page 263.

5.4.3.1 Not a Numbers

Not a Numbers (NaNs) are values that have the maximum biased exponent value and a nonzero fraction value. The sign bit is ignored, that is, NaNs are neither positive nor negative. If the high-order bit of the fraction field is 0, the NaN is a Signalling NaN (SNaN); otherwise it is a Quiet NaN (QNaN).

Signaling NaNs are used to signal exceptions when they appear as operands of computational instructions.

Quiet NaNs are used to represent the results of certain invalid operations, such as invalid arithmetic operations on infinities or on NaNs, when Invalid Operation Exception is disabled (FPSCR[VE] = 0). Quiet NaNs propagate through all floating-point instructions except **fcmpo**, **frsp**, and **fctiw**. Quiet NaNs do not signal exceptions, except for ordered comparison and conversion to integer operations. Specific encodings in QNaNs can thus be preserved through a sequence of floating-point operations, and used to convey diagnostic information to help identify results from invalid operations.

When a QNaN is the result of a floating-point operation because one of the operands is a NaN or because a QNaN was generated due to a disabled Invalid Operation exception, the following rule is applied to determine the NaN with the high-order fraction bit set to 1 that is to be stored as the result.

```
if FPR(FRA) is a NaN
then FPR(FRT) ← FPR(FRA)
else if FPR(FRB) is a NaN
then if instruction is frsp
        then FPR(FRT) ← FPR(FRB)0:34 || 29 0
        else FPR(FRT) ← FPR(FRB)
else if FPR(FRC) is a NaN
        then FPR(FRT) ← FPR(FRC)
        else if generated QNaN
                then FPR(FRT) ← generated QNaN
```

If the operand specified by FRA is a NaN, that NaN is stored as the result. Otherwise, if the operand specified by FRB is a NaN (if the instruction specifies an FRB operand), that NaN is stored as the result, with the low-order 29 bits of the result set to 0 if the instruction is **frsp**. Otherwise, if the operand specified by FRC is a NaN (if the instruction specifies an FRC operand), that NaN is stored as the result. Otherwise, if a QNaN was generated due to a disabled Invalid Operation Exception, that QNaN is stored as the result. If a QNaN is to be generated as a result, the QNaN generated has a sign bit of 0, an exponent field of all 1s, and a high-order fraction bit of 1 with all other fraction bits 0. Any instruction that generates a QNaN as the result of a disabled Invalid Operation must generate this QNaN (that is, 0x7FF8000000000000).

A double-precision NaN is representable in single format if and only if the low-order 29 bits of the double-precision NaNs fraction are zero.

5.4.4 Sign of Result

The following rules govern the sign of the result of an arithmetic, rounding, or conversion operation, when the operation does not yield an exception. They apply even when the operands or results are zeros or infinities.

- The sign of the result of an add operation is the sign of the operand having the larger absolute value. If both operands have the same sign, the sign of the result of an add operation is the same as the sign of the operands. The sign of the result of the subtract operation *x* – *y* is the same as the sign of the result of the add operation *x* + (–*y*).

When the sum of two operands with opposite sign, or the difference of two operands with the same sign, is exactly zero, the sign of the result is positive in all rounding modes except Round toward -Infinity, in which mode the sign is negative.

- The sign of the result of a multiply or divide operation is the Exclusive OR of the signs of the operands.
- The sign of the result of a *Square Root* or **frsqrte** instruction is always positive, except that the square root of –0 is –0 and the reciprocal square root of –0 is –Infinity.
- The sign of the result of an **frsp**[**.**], or **fctiw** operation is the sign of the operand being converted.

For the *Multiply-Add* instructions, the preceding rules are applied first to the multiply operation and then to the add or subtract operation (one of the inputs to the add or subtract operation is the result of the multiply operation).

5.4.5 Normalization and Denormalization

The intermediate result of an arithmetic or **frsp** instruction may require normalization and/or denormalization. Normalization and denormalization do not affect the sign of the result.

When an arithmetic or **frsp** instruction produces an intermediate result, consisting of a sign bit, an exponent, and a nonzero significand with a 0 leading bit, it is not a normalized number and must be normalized before it is stored.

A number is normalized by shifting its significand left while decrementing its exponent by 1 for each bit shifted, until the leading significand bit becomes 1. The G bit and the R bit (see *Execution Model for IEEE Operations* on page 172) participate in the shift with zeros shifted into the Round bit. The exponent is regarded as if its range were unlimited.

After normalization, or if normalization was not required, the intermediate result may have a nonzero significand and an exponent value that is less than the minimum value that can be represented in the format specified for the result. In this case, the intermediate result is said to be "Tiny" and the stored result is determined by the rules described in *Underflow Exception* on page 266. These rules may require denormalization.

A number is denormalized by shifting its significand right while incrementing its exponent by 1 for each bit shifted, until the exponent is equal to the format's minimum value. If any significant bits are lost in this shifting process, "Loss of Accuracy" has occurred (see *Underflow Exception* on page 266) and an Underflow Exception is signaled.

5.4.6 Data Handling and Precision

Instructions are defined to move floating-point data between the FPRs and storage. For double format data, the data are not altered during the move. For single format data, a format conversion from single to double is performed when loading from storage into an FPR. A format conversion from double to single is performed when storing from an FPR to storage. The *Load/Store* instructions do not cause floating-point exceptions.

All computational, *Move*, and **fsel** instructions use the floating-point double format.

Floating-point single-precision values are obtained with the following types of instruction.

- Load Floating-Point Single

  This form of instruction accesses a single-precision operand in single format in storage, converts it to double format, and loads it into an FPR. No floating-point exceptions are caused by these instructions.
- Round to Floating-Point Single-Precision

The **frsp** instruction rounds a double-precision operand to single-precision, checking the exponent for single-precision range and handling any exceptions according to respective enable bits, and places that operand into an FPR as a double-precision operand. For results produced by single-precision arithmetic instructions, single-precision loads, and other instances of the **frsp** instruction, this operation does not alter the value.

Programming Note: The **frsp** instruction enables value conversion from double-precision to single-precision with appropriate exception checking and rounding. This instruction should be used to convert double-precision floating-point values (produced by double-precision load and arithmetic instructions) to single-precision values before storing them into single format storage elements or using them as operands for single-precision arithmetic instructions. Values produced by single-precision load and arithmetic instructions are already single-precision values and can be stored directly into single format storage elements, or used directly as operands for single-precision arithmetic instructions, without preceding the store, or the arithmetic instruction, by an **frsp** instruction.

- Single-Precision Arithmetic Instructions

  This form of instruction takes operands from the FPRs in double format, performs the operation as if it produced an intermediate result having infinite precision and unbounded exponent range, and then coerces this intermediate result to fit in single format. Status bits in the FPSCR are set to reflect the single-precision result. The result is then converted to double format and placed into an FPR. The result lies in the range supported by the single format.

  All input values must be representable in single format. If they are not, the result placed into the target FPR, and the setting of status bits in the FPSCR, are undefined.

- Store Floating-Point Single

  This form of instruction converts a double-precision operand to single format and stores that operand into storage. No floating-point exceptions are caused by these instructions. (The value being stored is effectively assumed to be the result of an instruction of one of the preceding three types.)

When the result of a *Load Floating-Point Single*, **frsp**, or single-precision arithmetic instruction is stored in an FPR, the low-order 29 fraction bits are zero.

Programming Note: A single-precision value can be used in double-precision arithmetic operations. The reverse is true only if the double-precision value is representable in single format.

5.4.7 Rounding

Rounding applies to operations that have numeric operands (operands that are not infinities or NaNs). Rounding the intermediate result of such operations may cause an Overflow Exception, an Underflow Exception, or an Inexact Exception. The following description assumes that the operations cause no exceptions and that the result is numeric. See *Value Representation* on page 166 and *Floating Point Unit Interrupts and Exceptions* on page 259 for the cases not covered here.

Execution Model for IEEE Operations on page 172 provides a detailed explanation of rounding.

The *Arithmetic* and *Rounding and Conversion* instructions produce intermediate results that can be regarded as having infinite precision and unbounded exponent range. Such intermediate results are normalized or denormalized if required, then rounded to the target format. The final result is then placed into the target FPR in double format or in integer format, depending on the instruction.

The *Arithmetic* and *Rounding and Conversion* instructions, which round intermediate results, set FPSCR[FR, FI]. If the fraction was incremented during rounding, FPSCR[FR] = 1; otherwise, FPSCR[FR] = 0. If the rounded result is inexact, FPSCR[FI] = 1; otherwise, FPSCR[FI] = 0.

The *Estimate* instructions set FPSCR[FR, FI] to undefined values. The remaining floating-point instructions do not alter FPSCR[FR, FI].

FPSCR[RN] specifies one of four programmable rounding modes.

Let *z* be the intermediate arithmetic result or the operand of a convert operation. If *z* can be represented exactly in the target format, then the result in all rounding modes is *z* as represented in the target format. If *z* cannot be represented exactly in the target format, let *z1* and *z2* bound *z* as the next larger and next smaller numbers representable in the target format. Then, *z1* or *z2* can be used to approximate the result in the target format.

Figure 5-4 shows the relation of *z*, *z1*, and *z2* in this case. The following rules specify the rounding in the four modes. 'lsb' means 'least-significant bit'.

Figure 5-4. Selection of z1 and z2

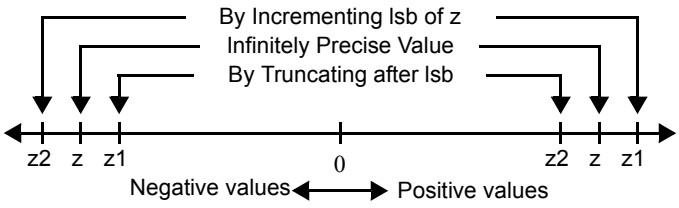


Table 5-8 describes the rounding modes.

Table 5-8. Rounding Modes

FPSCR[RN]	Rounding Mode	Description
00	Round to Nearest	Choose the value that is closest to *z*, either *z1* or *z2*. In case of a tie, choose the one that is even (the lsb is 0).
01	Round toward Zero	Choose the smaller in magnitude (*z1* or *z2*).
10	Round toward +Infinity	Choose *z1*.
11	Round toward –Infinity	Choose *z2*.

5.5 Floating-Point Execution Models

All implementations of this architecture must provide the equivalent of the following execution models to ensure that identical results are obtained.

Special rules are provided in the definition of the computational instructions for the infinities, denormalized numbers and NaNs. The material in the remainder of this section applies to instructions that have numeric operands and a numeric result (i.e., operands and result that are not infinities or NaNs), and that cause no exceptions. See *Value Representation* on page 166 and *Floating Point Unit Interrupts and Exceptions* on page 259 for the cases not covered here.

Although the double format specifies an 11-bit exponent, exponent arithmetic makes use of two additional bits to avoid potential transient overflow conditions. One extra bit is required when denormalized double-precision numbers are prenormalized. The second bit is required to permit the computation of the adjusted exponent value in the following cases when the corresponding exception enable bit is 1:

- Underflow during multiplication using a denormalized operand.
- Overflow during division using a denormalized divisor.

The IEEE standard includes 32-bit and 64-bit arithmetic. The standard requires that single-precision arithmetic be provided for single-precision operands. The standard permits double-precision floating-point operations to have either (or both) single-precision or double-precision operands, but states that single-precision floating-point operations should not accept double-precision operands. Book-E follows these guidelines: double-precision arithmetic instructions can have operands of either or both precisions, while single-precision arithmetic instructions require all operands to be single-precision. Double-precision arithmetic instructions produce double-precision values, while single-precision arithmetic instructions produce single-precision values.

For arithmetic instructions, conversions from double-precision to single-precision must be done explicitly by software, while conversions from single-precision to double-precision are done implicitly.

5.5.1 Execution Model for IEEE Operations

The following description uses 64-bit arithmetic as an example. 32-bit arithmetic is similar except that the FRACTION is a 23-bit field, and the single-precision Guard, Round, and Sticky bits (described in this section) are logically adjacent to the 23-bit FRACTION field.

IEEE-conforming significand arithmetic is considered to be performed with a floating-point accumulator having the following format, where bits 0:55 comprise the significand of the intermediate result.

Table 5-9. IEEE 64-bit Execution Model

S	C	L	FRACTION	G	R	X
		0	1 ... 52			55

The S bit is the sign bit.

The C bit is the carry bit, which captures the carry out of the significand.

The L bit is the leading unit bit of the significand, which receives the implicit bit from the operand.

The FRACTION is a 52-bit field that accepts the fraction of the operand.

The Guard (G), Round (R), and Sticky (X) bits are extensions to the low-order bits of the accumulator. The G and R bits are required for post-normalization of the result. The G, R, and X bits are required during rounding to determine if the intermediate result is equally near the two nearest representable values. The X bit serves as an extension to the G and R bits by representing the logical OR of all bits that may appear to the low-order side of the R bit, due either to shifting the accumulator right or to other generation of low-order result bits. The G and R bits participate in the left shifts with zeros being shifted into the R bit. *Table 5-10* shows the significance of the G, R, and X bits with respect to the intermediate result (IR), the representable number next lower in magnitude (NL), and the representable number next higher in magnitude (NH).

Table 5-10. Interpretation of the G, R, and X Bits

G	R	X	Interpretation
0	0	0	IR is exact
0 0 0	0 1 1	1 0 1	IR closer to NL
1	0	0	IR midway between NL and NH
1 1 1	0 1 1	1 0 1	IR closer to NH

After normalization, the intermediate result is rounded, using the rounding mode specified by FPSCR[RN]. If rounding results in a carry into C, the significand is shifted right one position and the exponent incremented by one. This yields an inexact result and possibly also exponent overflow. Fraction bits to the left of the bit position used for rounding are stored into the FPR and low-order bit positions, if any, are set to zero.

Four user-selectable rounding modes are provided through FPSCR[RN] as described in *Rounding* on page 170. For rounding, the conceptual Guard, Round, and Sticky bits are defined in terms of accumulator bits. *Table 5-11* shows the positions of the Guard, Round, and Sticky bits for double-precision and single-precision floating-point numbers in the IEEE execution model.

Table 5-11. Location of the Guard, Round, and Sticky Bits in the IEEE Execution Model

Format	Guard	Round	Sticky
Double Single	G bit 24	R bit 25	X bit OR of 26:52, G, R, X

Rounding can be treated as though the significand were shifted right, if required, until the least significant bit to be retained is in the low-order bit position of the FRACTION. If any of the Guard, Round, or Sticky bits is nonzero, then the result is inexact.

Z1 and *Z2*, as defined in *Rounding* on page 170, can be used to approximate the result in the target format when one of the following rules is used.

- Round to Nearest
  - Guard bit = 0

    The result is truncated. (Result exact (GRX = 000) or closest to next lower value in magnitude (GRX = 001, 010, or 011))
  - Guard bit = 1

    Depends on Round and Sticky bits:
    - Case a

      If the Round or Sticky bit is 1 (inclusive), the result is incremented. (Result closest to next higher value in magnitude (GRX = 101, 110, or 111))
    - Case b

If the Round and Sticky bits are 0 (result midway between closest representable values), then if the low-order bit of the result is 1 the result is incremented. Otherwise (the low-order bit of the result is 0) the result is truncated (this is the case of a tie rounded to even).

- Round toward Zero

  Choose the smaller in magnitude of *Z1* or *Z2*. If the Guard, Round, or Sticky bit is nonzero, the result is inexact.

- Round toward +Infinity

  Choose *Z1*.

- Round toward –Infinity

  Choose *Z2*.

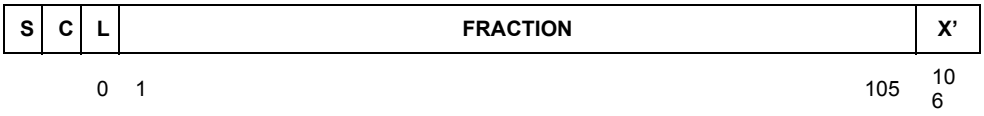
Where the result is to have fewer than 53 bits of precision because the instruction is a *Floating Round to Single-Precision* or single-precision arithmetic instruction, the intermediate result is either normalized or placed in correct denormalized form before being rounded.

5.5.2 Execution Model for Multiply-Add Type Instructions

The PPC440 FPU provides a special form of instruction that performs up to three operations in one instruction (a multiplication, an addition, and a negation). With this added capability comes the special ability to produce a more exact intermediate result as input to the rounder. 32-bit arithmetic is similar except that the FRACTION field is smaller.

Multiply-add significand arithmetic is considered to be performed with a floating-point accumulator having the following format, where bits 0:106 comprise the significand of the intermediate result.

Table 5-12. Multiply-Add 64-bit Execution Model



The first part of the operation is a multiplication. The multiplication has two 53-bit significands as inputs, which are assumed to be prenormalized, and produces a result conforming to the above model. If there is a carry out of the significand (into the C bit), then the significand is shifted right one position, shifting the L bit (leading unit bit) into the most significant bit of the FRACTION and shifting the C bit (carry out) into the L bit. All 106 bits (L bit, the FRACTION) of the product take part in the add operation. If the exponents of the two inputs to the adder are not equal, the significand of the operand with the smaller exponent is aligned (shifted) to the right by an amount that is added to that exponent to make it equal to the other input's exponent. Zeros are shifted into the left of the significand as it is aligned and bits shifted out of bit 105 of the significand are ORed into the X' bit. The add operation also produces a result conforming to the above model with the X' bit taking part in the add operation.

The result of the addition is then normalized, with all bits of the addition result, except the X' bit, participating in the shift. The normalized result serves as the intermediate result that is input to the rounder.

For rounding, the conceptual Guard, Round, and Sticky bits are defined in terms of accumulator bits. *Table 5-13* shows the positions of the Guard, Round, and Sticky bits for double-precision and single-precision floating-point numbers in the multiply-add execution model.

Table 5-13. Location of Guard, Round, and Sticky Bits in the Multiply-Add Execution Model

Format	Guard	Round	Sticky
Double	53	54	OR of 55:105, X'
Single	24	25	OR of 26:105, X'

The rules for rounding the intermediate result are the same as those given in *Execution Model for IEEE Operations* on page 172.

If the instruction is *Floating Negative Multiply-Add* or *Floating Negative Multiply-Subtract*, the final result is negated.

5.6 Floating-Point Instructions

Primary opcode 63 is used for the double-precision arithmetic instructions and miscellaneous instructions, such as the *Floating-Point Status and Control Register Manipulation* instructions. Primary opcode 59 is used for the single-precision arithmetic instructions.

The single-precision instructions for which there is a corresponding double-precision instruction have the same format and extended opcode as the corresponding double-precision instruction.

Instructions are provided to perform arithmetic, rounding, conversion, comparison, and other operations in floating-point registers; to move floating-point data between storage and these registers; and to manipulate the FPSCR explicitly.

These instructions are divided into two categories.

- Computational instructions

  The computational instructions are those that perform addition, subtraction, multiplication, division, extracting the square root, rounding, conversion, comparison, and combinations of these operations. These instructions provide the floating-point operations. They place status information into the FPSCR. They are the instructions described in *Floating-Point Arithmetic Instructions* on page 180, *Floating-Point Rounding and Conversion Instructions* on page 181, and *Floating-Point Compare Instructions* on page 181.

- Noncomputational instructions

  The noncomputational instructions that perform loads and stores, move the contents of a floating-point register to another floating-point register possibly altering the sign, manipulate the FPSCR explicitly, and select a value from one of two floating-point registers based on the value in a third floating-point register. These operations are not considered floating-point operations. With the exception of the instructions that manipulate the FPSCR explicitly, they do not alter the FPSCR. Those instructions are described in *Floating-Point Status and Control Register Instructions* on page 181.

A floating-point number consists of a signed exponent and a signed significand. The quantity expressed by this number is the product of the significand and the number $2^{exponent}$. Encodings are provided in the data format to represent finite numeric values, ±Infinity, and values that are "not a number" (NaN). Operations involving infinities produce results following traditional mathematical conventions. NaNs have no mathematical interpretation, but their encoding supports a variable diagnostic information field. NaNs may be used to indicate such things as uninitialized variables, and can be produced by certain invalid operations.

One class of exceptions that occur during floating-point instruction execution is unique to floating-point operations: the Floating-point exception. Bits set in the FPSCR indicate floating-point exceptions. They can cause an Enabled exception type Program interrupt to be taken, precisely or imprecisely, if the proper control bits are set.

5.6.1 Instructions By Category

The floating-point instructions can be classified into computational and noncomputational categories. The computational instructions include those that perform arithmetic operations or conversions on operands. Noncomputational instructions perform loads/stores and moves (with possible sign changes), or select data. Additionally, some noncomputational instructions can write directly to the FPSCR. All instructions executed in the Load/Store Pipeline are noncomputational, while most executed in the Arithmetic pipe are computational.

All floating-point operands are stored internally in Double Precision Format. Arithmetic operations specified as Single, require that the internal data is representable as Single (that is, having an unbiased exponent between –126 and 127 and a significand accurately representable in 24 bits). If the data cannot be represented in this way, the results stored in FPR, and the status bits set in FPSCR and CR (as appropriate), are undefined.

For consistency, to reduce the likelihood of causing a serious malfunction due to user error, and to enable random testing, single-precision operations are performed on double-precision operands. For all cases except for **fdivs**, the operation is performed as if it were double-precision; the result is then rounded to single-precision. For **fdivs**, the appropriate number of iterations are performed to accomplish a single-precision result (potentially with early out); the quotient is then properly rounded.

In all cases, result exceptions (overflow, underflow, and inexact) are detected and reported based on the result, not on the source operands. Default (masked exception) results are the same as for the single-precision instructions. In the case of masked overflow or underflow exceptions, the least significant 11 bits of the adjusted true exponent are returned.

The results of all single-precision operations are rounded to Single Precision. These results are stored in Double-Precision format, but are restricted to Single-Precision range (exponent and fraction). All status bits are set based upon the single-precision result.

5.6.2 Load and Store Instructions

The PPC440 FPU instruction set includes instructions to load from memory to an FPR, and to store from an FPR to memory.

For load instructions, the function of the LSC is to receive data from the 16 byte bus from the PPC440EP and present it to the FPRs. Data received from PPC440EP could be single or double precision, and in the big or little endian formats. Also, the data received is word aligned. Data to the FPR must be in the big endian, double precision format.

For store instructions one operand from the FPR, or a bypass path, is received. Data is to be word aligned on the output bus, not all cases can be handled with a throughput of one per cycle. FPR data (or bypassed data) for single precision stores that needs to be denormalized to fit in the single precision format requires multiple cycles. Also data for double precision stores may need to be normalized if the original data source was a single precision denormalized number. There are two basic forms of load instruction: single-precision and double-precision. Because the FPRs support only floating-point double format, single-precision *Load Floating-Point* instructions convert single-precision data to double format prior to loading the operand into the target FPR. The conversion and loading steps are as follows.

Let $WORD_{0:31}$ be the floating-point single-precision operand accessed from storage.

Normalized Operand
if $WORD_{1:8} > 0$ and $WORD_{1:8} < 255$ then
$FPR(FRT)_{0:1} \leftarrow WORD_{0:1}$
$FPR(FRT)_{2} \leftarrow \neg WORD_{1}$
$FPR(FRT)_{3} \leftarrow \neg WORD_{1}$
$FPR(FRT)_{4} \leftarrow \neg WORD_{1}$
$FPR(FRT)_{5:63} \leftarrow WORD_{2:31} \,||\, {}^{29}0$
Denormalized Operand
if $WORD_{1:8} = 0$ and $WORD_{9:31} \neq 0$ then
sign $\leftarrow WORD_{0}$
exp $\leftarrow$ -126
$frac_{0:52} \leftarrow 0b0 \,||\, WORD_{9:31} \,||\, {}^{29}0$
normalize the operand
do while $frac_{0} = 0$
frac $\leftarrow frac_{1:52} \,||\, 0b0$
exp $\leftarrow$ exp - 1
$FPR(FRT)_{0} \leftarrow$ sign
$FPR(FRT)_{1:11} \leftarrow$ exp + 1023
$FPR(FRT)_{12:63} \leftarrow frac_{1:52}$
Zero / Infinity / NaN
if $WORD_{1:8} = 255$ or $WORD_{1:31} = 0$ then
$FPR(FRT)_{0:1} \leftarrow WORD_{0:1}$
$FPR(FRT)_{2} \leftarrow WORD_{1}$
$FPR(FRT)_{3} \leftarrow WORD_{1}$
$FPR(FRT)_{4} \leftarrow WORD_{1}$
$FPR(FRT)_{5:63} \leftarrow WORD_{2:31} \,||\, {}^{29}0$

For double-precision *Load Floating-Point* instructions no conversion is required, as the data from storage are copied directly into the FPR.

Some of the *Floating-Point Load* instructions update GPR(RA) the effective address. For these forms, if RA≠0, the effective address is placed into GPR(RA) and the storage element (byte, half word, word, or double word) addressed by EA is loaded into FPR(RT). If RA=0, the instruction form is invalid.

Floating-Point Load storage accesses cause Data Storage exceptions if the program is not allowed to read the storage location. *FLoating-Point Load* storage accesses cause Data TLB Error exceptions if the program attempts to access storage that is unavailable.

Note: RA and RB denote GPRs, while FRT denotes an FPR.

Both big endian and little endian byte orderings are supported.

Table 5-14. Floating-Point Load Instructions

Mnemonic	Operands	Instruction	Page
lfd	FRT, D(RA)	Load Floating-Point Double	788
lfdu	FRT, D(RA)	Load Floating-Point Double with Update	789
lfdux	FRT, RA, R	Load Floating-Point Double with Update Indexed	790
lfdx	FRT, RA, R	Load Floating-Point Double Indexed	791
lfs	FRT, D(RA)	Load Floating-Point Single	792
lfsu	FRT, D(RA)	Load Floating-Point Single with Update	793
lfsux	FRT, RA, RB	Load Floating-Point Single with Update Indexed	794
lfsx	FRT, RA, RB	Load Floating-Point Single Indexed	795

5.6.3 Floating-Point Store Instructions

There are three basic forms of store instruction: single-precision, double-precision, and integer. The integer form is provided by the **stfiwx** instruction, described on page 806. Because the FPRs support only floating-point double format for floating-point data, single-precision *Store Floating-Point* instructions convert double-precision data to single format before storing the operand in storage. The conversion steps are as follows.

Let $WORD_{0:31}$ be the word in storage written to.

No Denormalization Required (includes Zero / Infinity / NaN)
if $FPR(FRS)_{1:11} > 896$ or $FPR(FRS)_{1:63} = 0$ then
$WORD_{0:1} \leftarrow FPR(FRS)_{0:1}$
$WORD_{2:31} \leftarrow FPR(FRS)_{5:34}$
Denormalization Required
if $874 \leq FRS_{1:11} \leq 896$ then
$sign \leftarrow FPR(FRS)_0$
$exp \leftarrow FPR(FRS)_{1:11} - 1023$
$frac \leftarrow 0b1 \,||\, FPR(FRS)_{12:63}$
denormalize operand
do while $exp < -126$
$frac \leftarrow 0b0 \,||\, frac_{0:62}$
$exp \leftarrow exp + 1$
$WORD_0 \leftarrow sign$
$WORD_{1:8} \leftarrow 0x00$
$WORD_{9:31} \leftarrow frac_{1:23}$
else $WORD \leftarrow$ undefined

Notice that if the value to be stored by a single-precision *Store Floating-Point* instruction is larger in magnitude than the maximum number representable in single format, the first case above ("No Denormalization Required") applies. The result stored in WORD is then a well-defined value, but is not numerically equal to the value in the source register. The result of a single-precision *Load Floating-Point* from WORD will not compare equal to the contents of the original source register.

For double-precision *Store Floating-Point* instructions and for the *Store Floating-Point as Integer Word* instruction no conversion is required, as the data from the FPR are copied directly into storage.

Some of the *Floating-Point Store* instructions update GPR(RA) with the effective address. For these forms, if $RA \neq 0$, the effective address is placed into GPR(RA).

Floating-Point Store storage accesses will cause a Data Storage interrupt if the program is not allowed to write to the storage location. *Integer Store* storage accesses will cause a Data TLB Error interrupt if the program attempts to access storage that is unavailable.

Note: RA and RB denote GPRs, while FRS denotes an FPR.

Both big endian and little endian byte orderings are supported.

Table 5-15. Floating-Point Store Instructions

Mnemonic	Operands	Instruction	Page
stfd	FRS, D(RA)	Store Floating-Point Double	802
stfdu	FRS, D(RA)	Store Floating-Point Double with Update	803
stfdux	FRS, RA, RB	Store Floating-Point Double with Update Indexed	804
stfdx	FRS, RA, RB	Store Floating-Point Double Indexed	805
stfiwx	FRS, RA, RB	Store Floating-Point as Integer Word Indexed	806
stfs	FRS, D(RA)	Store Floating-Point Single	807
stfsu	FRS, D(RA)	Store Floating-Point Single with Update	808
stfsux	FRS, RA, RB	Store Floating-Point Single with Update Indexed	809
stfsx	FRS, RA, RB	Store Floating-Point Single Indexed	810

5.6.4 Floating-Point Move Instructions

These instructions copy data from one floating-point register to another, altering the sign bit (bit 0) as described in the instruction descriptions in *Floating Point Instruction Set* on page 755 for **fneg**, **fabs**, and **fnabs**. These instructions treat NaNs just like any other kind of value (for example, the sign bit of an NaN may be altered by **fneg**, **fabs**, and **fnabs**). These instructions do not alter the FSPCR.

Table 5-16. Floating-Point Move Instructions

Mnemonic	Operands	Instruction	Page
fabs	FRT, FRB	Floating Absolute Value	759
fmr	FRT, FRB	Floating Move Register	770
fnabs	FRT, FRB	Floating Negative Absolute Value	775
fneg	FRT, FRB	Floating Negate	776

5.6.5 Floating-Point Arithmetic Instructions

These instructions perform elementary arithmetic operations.

Table 5-17. Floating-Point Elementary Arithmetic Instructions

Mnemonic	Operands	Instruction	Page
fadd	FRT, FRA, FRB	Floating Add	760
fadds	FRT, FRA, FRB	Floating Add Single	761
fdiv	FRT, FRA, FRB	Floating Divide	766
fdivs	FRT, FRA, FRB	Floating Divide Single	767
fmul	FRT, FRA, FRB	Floating Multiply	773
fmuls	FRT, FRA, FRB	Floating Multiply Single	774
fres	FRT, FRB	Floating Reciprocal Estimate Single	781
frsqrte	FRT, FRB	Floating Reciprocal Square Root Estimate	783
fsub	FRT, FRA, FRB	Floating Subtract	786
fsubs	FRT, FRA, FRB	Floating Subtract Single	787

5.6.5.1 Floating-Point Multiply-Add Instructions

These instructions combine a multiply and an add operation without an intermediate rounding operation. The fraction part of the intermediate product is 106 bits wide (L bit, FRACTION), and all 106 bits take part in the add/subtract portion of the instruction.

FPSCR bits are set as follows.

- Overflow, Underflow, and Inexact Exception bits, the FR and FI bits, and the FPRF field are set based on the final result of the operation, and not on the result of the multiplication.
- Invalid Operation exception bits are set as if the multiplication and the addition were performed using two separate instructions (**fmul**[**s**], followed by **fadd**[**s**] or **fsub**[**s**]. That is, multiplication of infinity by 0 or of anything by an SNaN, and addition of an SNaN, cause the corresponding exception bits to be set.

Table 5-18. Floating-Point Multiply-Add Instructions

Mnemonic	Operands	Instruction	Page
fmadd	FRT, FRA, FRB, FRC	Floating Multiply-Add	768
fmadds	FRT, FRA, FRB, FRC	Floating Multiply-Add Single	769
fmsub	FRT, FRA, FRB, FRC	Floating Multiply-Subtract	771
fmsubs	FRT, FRA, FRB, FRC	Floating Multiply-Subtract Single	772
fnmadd	FRT, FRA, FRB, FRC	Floating Negative Multiply-Add	777
fnmadds	FRT, FRA, FRB, FRC	Floating Negative Multiply-Add Single	778
fnmsub	FRT, FRA, FRB, FRC	Floating Negative Multiply-Subtract	779
fnmsubs	FRT, FRA, FRB, FRC	Floating Negative Multiply-Subtract Single	780

5.6.6 Floating-Point Rounding and Conversion Instructions

Examples of uses of these instructions to perform various conversions can be found in Appendix X, "Floating-Point Conversions", on page XREF TBD.

Table 5-19. Floating-Point Rounding and Conversion Instructions

Mnemonic	Operand	Instruction	Page
fctiw	FRT, FRB	Floating Convert To Integer Word	764
fctiwz	FRT, FRB	Floating Convert To Integer Word and round to Zero	765
frsp	FRT, FRB	Floating Round to Single-Precision	782

5.6.7 Floating-Point Compare Instructions

The floating-point *Compare* instructions compare the contents of two floating-point registers. Comparison ignores the sign of zero (+0 is treated as equal to –0). The comparison result can be ordered or unordered.

The comparison sets one bit in the designated CR field to 1 and the other three bits to 0. FPSCR[FPCC] is set in the same way.

The CR field and FPSCR[FPCC] are set as follows.

Table 5-20. Comparison Sets

Bit	Name	Description
0	FL	(FRA) < (FRB)
1	FG	(FRA) > (FRB)
2	FE	(FRA) = (FRB)
3	FU	(FRA) ? (FRB) (unordered)

Table 5-21. Floating-Point Compare and Select Instructions

Mnemonic	Operands	Instruction	Page
fcmpo	BF, FRA, FRB	Floating Compare Ordered	762
fcmpu	BF, FRA, FRB	Floating Compare Unordered	763
fsel	FRT, FRA, FRB, FRC	Floating Select	785

5.6.8 Floating-Point Status and Control Register Instructions

Every *Floating-Point Status and Control Register* instruction synchronizes the effects of all floating-point instructions executed by a given processor. Executing a *Floating-Point Status and Control Register* instruction ensures that all floating-point instructions previously initiated by the given processor have completed before the *Floating-Point Status and Control Register* instruction is initiated, and that no subsequent floating-point instructions are initiated by the given processor until the *Floating-Point Status and Control Register* instruction has completed. In particular:

- All exceptions that will be caused by the previously initiated instructions are recorded in the FPSCR before the *Floating-Point Status and Control Register* instruction is initiated.

- All invocations of the Enabled exception type Program interrupt that will be caused by the previously initiated instructions have occurred before the *Floating-Point Status and Control Register* instruction is initiated.
- No subsequent floating-point instruction that depends on or alters the settings of any FPSCR bits is initiated until the *Floating-Point Status and Control Register* instruction has completed.

Floating-Point Load and *Floating-Point Store* instructions are not affected.

Table 5-22. Floating-Point Status and Control Register Instructions

Mnemonic	Operands	Instruction	Page
mcrfs		Move To Condition Register From FPSCR	796
mffs	FRT	Move From FPSCR	797
mtfsb0	BT	Move To FPSCR Bit 0	798
mtfsb1	BT	Move To FPSCR Bit 1	799
mtfsf	FLM, FRB	Move To FPSCR Fields	800
mtfsfi	BF,U	Move To FPSCR Field Immediate	801

6. Instruction and Data Caches

The PPC440EP provides separate instruction and data cache controllers and arrays, which allow concurrent access and minimize pipeline stalls. The storage capacity of both cache arrays is 32KB. Both cache controllers have 32-byte lines, and both are highly associative, having 64-way set-associativity. The PowerPC instruction set provides a rich set of cache management instructions for software-enforced coherency. The PPC440EP implementation also provides special debug instructions that can directly read the tag and data arrays. The cache controllers interface to the processor local bus (PLB) for connection to the IBM CoreConnect system-on-a-chip environment.

Both the data and instruction caches are parity protected against soft errors. If such errors are detected, the CPU will vector to the machine check interrupt handler, where software can take appropriate action.

Refer to the *PPC440 Processor User's Manual* for details.

7. Memory Management

The 440EP supports a uniform, 4 GB effective address (EA) space, and a 64 GB (36-bit) real address (RA) space. The 440EP memory management unit (MMU) performs address translation between effective and real addresses, as well as protection functions. With appropriate system software, the MMU supports:

- Translation of effective addresses into real addresses
- Software control of the page replacement strategy
- Page-level access control for instruction and data accesses
- Page-level storage attribute control

Refer to the *PPC440 Processor User's Manual* for details.

Part III. PPC440EP System Operations

8. Reset and Initialization

This chapter describes the initial state of the PPC440EP after a hardware reset, and contains a description of the initialization software required to complete initialization so that the PPC440EP can begin executing application code. Initialization of other on-chip and off-chip system components is described in the appropriate chapter.

8.1 Reset Signals

The PPC440EP provides two reset signals, System_Reset_N and External_Reset_N. System_Reset_N is bidirectional and External_Reset_N is an output.

When the System_Reset_N signal is an input (asserted by an off-chip device), such as during power on reset (POR), the chip responds by performing a system reset as described in a following section. An external assertion of System_Reset_N is not extended by an assertion of the open drain bidirectional System_Reset_N driver.

As an output, the PPC440EP asserts the System_Reset_N signal when a system reset request is detected. The System_Reset_N open drain driver is activated and the signal is driven low for at least 16K SysClk periods. This enables the PPC440EP to reset itself and other devices attached to the same reset network.

The External_Reset_N signal is used by synchronous peripheral devices served by the external bus clock, such as ROM and external masters. During chip and system resets, External_Reset_N is asserted until the PerClk signal is stable and all internal resets are released. Its duration in effect is the same as that of signal RstTopAllLogicRst which resets all on-chip peripherals.

8.2 Reset Types

Three types of reset, each with different scope, are possible in the PPC440EP. A CPU reset affects only the processor core and the FPU core. Chip resets affect the processor core, FPU core and all on-chip peripherals. System resets affect the processor core, FPU core, all on-chip peripherals, and any off-chip devices connected to the PPC440EP System_Reset_N signal. The effects of system, chip and CPU resets on the processor core and FPU core are identical. To determine which reset type occurred, the most-recent reset (MRR) field of the Debug Status Register (DBSR) can be examined.

8.2.1 CPU Reset

A CPU reset results in a reset of the processor core and FPU core. No other on-chip logic is affected. Reset_top logic, outside the processor core, detects the CPU reset request and asserts the reset input to the processor core and FPU core. The duration of a CPU reset will last 10 system clock cycles beyond the CPU core reset request, which typically lasts 4 CPU clock cycles.

8.2.2 Chip Reset

A chip reset results in the reset of the processor core, FPU core and on-chip peripherals. A chip reset request is generated by the processor core so that the processor core, FPU core and all on-chip peripherals reset for an extended period while the CPR PLL is allowed to re-lock. The duration of a chip reset will last 10 system clocks beyond the CPR PLL locking.

During chip reset, the External_Reset_N signal is driven low to ensure the reset of synchronous devices that use the external bus clock signal, PerClk.

8.2.3 System Reset

A system reset results in a reset of all PPC440EP logic, including the processor core, FPU core, internal phase-locked loops (CPR PLL), and on-chip peripherals. A system reset can be initiated externally or internally. External system resets are initiated by the assertion of the System_Reset_N signal for at least 16 SysClk cycles. Internal system resets are initiated by either the processor core the duration of a system reset will last 10 system clocks beyond the CPR PLL locking.

When a system reset is requested internally, the bidirectional open drain System_Reset_N signal is asserted to enable other chips to be reset at the same time. In this case, the System_Reset_N signal is driven low for 16384 SysClk cycles, resulting in a System reset of the PPC440EP chip and all other devices attached to the reset network connected to System_Reset_N. During this time all internal clocks and PerClk clock toggle.

After the System_Reset_N signal is de asserted, the CPR PLL begins its locking process, which requires about 8350 SysClk cycles when CPU feedback is selected and 16700 SysClk cycles when remote PerClk feedback is selected. During this time all internal clocks and PerClk clock toggle.

When the PLL locking process is complete, internal resets are released by Reset_top logic, and the processor core begins its initial instruction fetch.

During system reset, the External_Reset_N signal is driven low to ensure the reset of synchronous devices that use the external bus clock signal, PerClk.

Figure 8-1. PPC440EP Power-on Reset Process

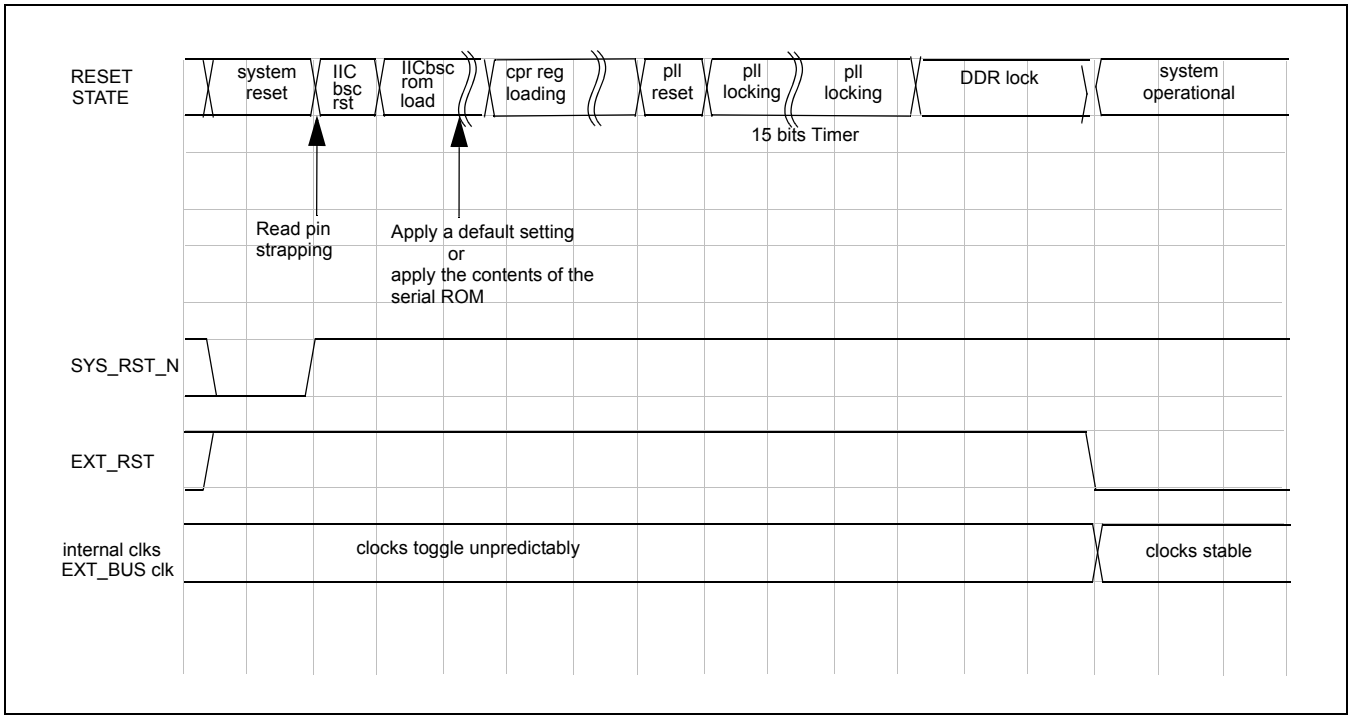
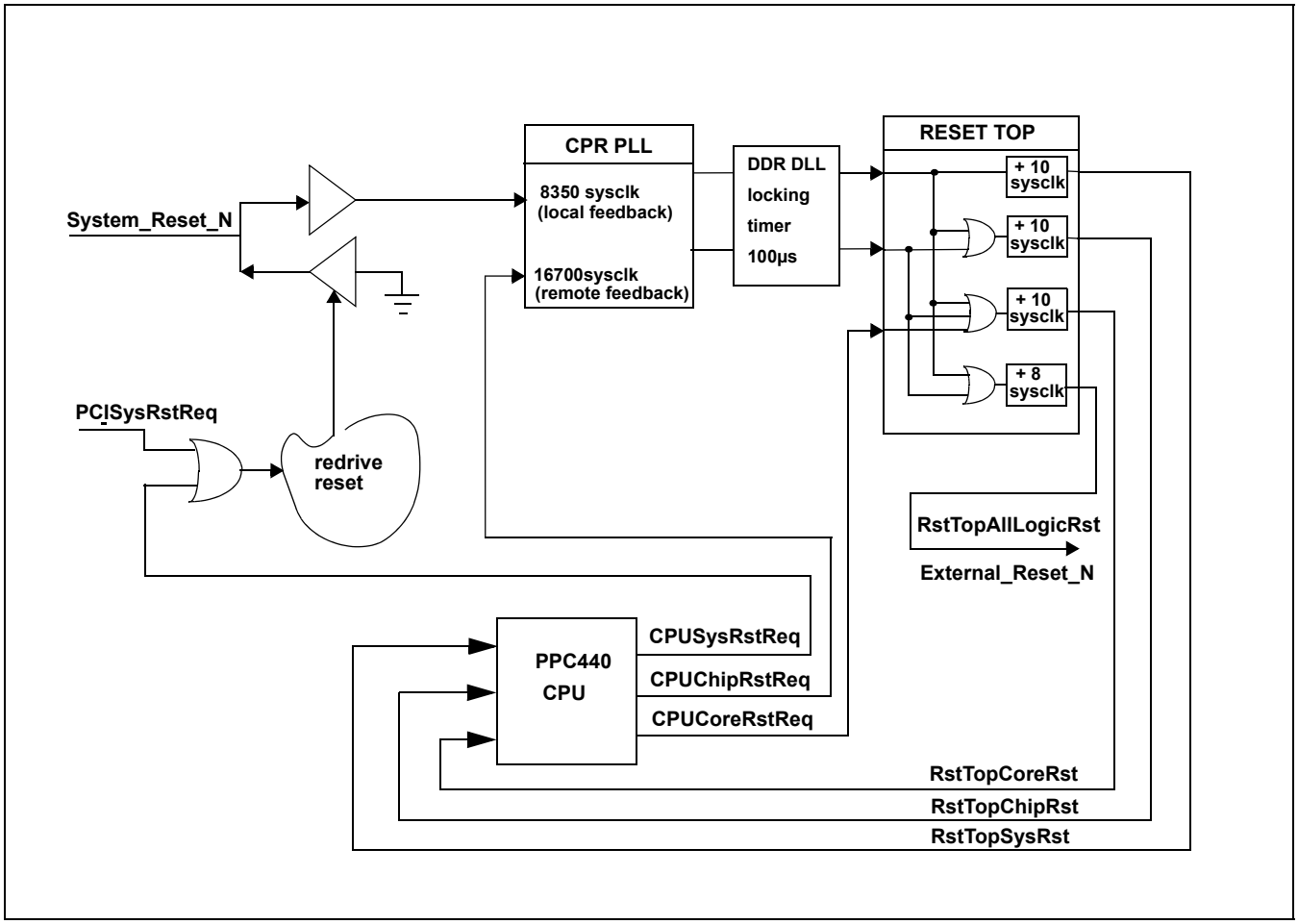


Figure 8-2. PPC440EP Reset



8.3 Processor Core State After Reset

In general, the contents of registers and other facilities within the PPC440EP processor core are undefined after a hardware reset. Reset is defined to initialize only the minimal resources required such that instructions can be fetched and executed from the initial program memory page, and so that repeatable, deterministic behavior can be guaranteed provided that the proper software initialization sequence is followed. System software must fully configure the rest of the PPC440EP resources, as well as the other facilities within the chip and/or system.

The following list summarizes the requirements of the Book-E Enhanced PowerPC Architecture with regards to the processor state after reset, prior to any additional initialization by software.

- All fields of the MSR are set to 0, disabling all asynchronous interrupts, placing the processor in supervisor mode, and specifying that instruction and data accesses are to the system (as opposed to application) address space.
- DBCR0[RST] is set to 0, thereby ending any previous software-initiated reset operation.
- DBSR[MRR] records the type of the just ended reset operation (core, chip, or system; see *Reset Types* on page 189).
- TCR[WRC] is set to 0, thereby disabling the Watchdog timer reset operation.

- TSR[WRS] records the type of the just ended reset operation, if the reset was initiated by the Watchdog Timer (otherwise this field is unchanged from its pre-reset value).
- The PVR is defined, after reset and otherwise, to contain a value that indicates the specific processor implementation.
- The program counter (PC) is set to 0xFFFFFFFC, the effective address (EA) of the last word of the address space.

The memory management resources are set to values such that the processor is able to successfully fetch and execute instructions and read (but not write) data within the 4KB program memory page located at the end of the 32-bit effective address space. Exactly how this is accomplished is implementation-dependent. For example, it may or may not be the case that a TLB entry is established in a manner which is visible to software using the TLB management instructions. Regardless of how the implementation enables access to the initial program memory page, instruction execution starts at the effective address of 0xFFFFFFFC, the last word of the effective address space. The instruction at this address must be an unconditional branch backwards to the start of the initialization sequence, which must lie somewhere within the initial 4KB program memory page. The real address to which the initial effective address will be translated is also implementation- or system-dependent, as are the various storage attributes of the initial program memory page such as the caching inhibited and endian attributes.

Note: In the PPC440EP, a single entry is established in the instruction shadow TLB (ITLB) and data shadow TLB (DTLB) at reset with the properties described in *Table 8-1 Reset Values of Registers and Other 440EP Core Facilities* on page 192. It is required that initialization software insert an entry into the UTLB to cover this same memory region before performing any context synchronizing operation (including causing any exceptions which would lead to an interrupt), since a context synchronizing operation will invalidate the shadow TLB entries.

Initialization software should consider all other resources within the PPC440EP to be undefined after reset, in order for the initialization sequence to be compatible with other PowerPC implementations. There are, however, additional core resources which are initialized by reset, in order to guarantee correct and deterministic operation of the processor during the initialization sequence. *Table 8-1* shows the reset state of all PPC440EP resources which are defined to be initialized by reset. While certain other register fields and other facilities within the PPC440EP may be affected by reset, this is not an architectural nor hardware requirement, and software must treat those resources as undefined. Likewise, even those resources which are included in *Table 8-1* but which are not identified in the previous list as being architecturally required, should be treated as undefined by the initialization software.

Table 8-1. Reset Values of Registers and Other 440EP Core Facilities

Resource	Field	Reset Value	Comment
CCR0	DAPUIB	0	Enable broadcast of instruction data to auxiliary processor interface
	DTB	0	Enable broadcast of trace information
CCR1	ICDPEI	0	Disable Parity Error Insertion (enabled only for s/w testing)
	ICTPEI	0	
	DCTPEI	0	
	DCDPEI	0	
	DCUPEI	0	
	DCMPEI	0	
	FCOM	0	Do not force cache operations to miss.
	MMUPEI	0	Disable Parity Error Insertion (enabled only for s/w testing)
	FFF	0	Flush only as much data from dirty lines as needed.

Table 8-1. Reset Values of Registers and Other 440EP Core Facilities (continued)

Resource	Field	Reset Value	Comment
DBCR0	EDM	0	External Debug mode disabled
	RST	0b00	Software-initiated debug reset disabled
	ICMP	0	Instruction completion debug events disabled
	BRT	0	Branch taken debug events disabled
	IAC1	0	Instruction Address Compare 1 (IAC1) debug events disabled
	IAC2	0	IAC2 debug events disabled
	IAC3	0	IAC3 debug events disabled
	IAC4	0	IAC4 debug events disabled
DBSR	UDE	0	Unconditional debug event has not occurred
	MRR	Reset dependent	Indicates most recent type of reset as follows: 00 No reset has occurred since this field last cleared by software 01 Core reset 10 Chip reset 11 System reset
	ICMP	0	Instruction completion debug event has not occurred
	BRT	0	Branch taken debug event has not occurred
	IRPT	0	Interrupt debug event has not occurred
	TRAP	0	Trap debug event has not occurred
	IAC1	0	IAC1 debug event has not occurred
	IAC2	0	IAC2 debug event has not occurred
	IAC3	0	IAC3 debug event has not occurred
	IAC4	0	IAC4 debug event has not occurred
	DAC1R	0	Data address compare 1 (DAC1) read debug event has not occurred
	DAC1W	0	DAC1 write debug event has not occurred
	DAC2R	0	DAC2 read debug event has not occurred
	DAC2W	0	DAC2 write debug event has not occurred
	RET	0	Return debug event has not occurred
ESR	MCI	0	Synchronous Instruction Machine Check exception has not occurred
MCSR	MCS	0	Asynchronous Instruction Machine Check exception has not occurred

Table 8-1. Reset Values of Registers and Other 440EP Core Facilities (continued)

Resource	Field	Reset Value	Comment
MSR	WE	0	Wait state disabled
	CE	0	Asynchronous critical interrupts disabled
	EE	0	Asynchronous non-critical interrupts disabled
	PR	0	Processor in supervisor mode
	FP	0	Floating-point Unavailable interrupts disabled
	ME	0	Machine Check interrupts disabled
	FE0	0	Floating-point Enabled interrupts disabled
	DWE	0	Debug Wait mode disabled
	DE	0	Debug interrupts disabled
	FE1	0	Floating-point Enabled interrupts disabled
	IS	0	Instruction fetch access is to system-level virtual address space
	DS	0	Data access is to system level virtual address space
PC		0xFFFFFFFC	Initial reset instruction fetched from last word of effective address space
PVR	OWN	System dependent	PVR[OWN] value (after reset and otherwise) is specified by core input signals
	PVN	System dependent	PVR[PVN] value (after reset and otherwise) is specified by core input signals
RSTCFG	U0	System dependent	All RSTCFG fields are specified by core input signals
	U1	System dependent	
	U2	System dependent	
	U3	System dependent	
	E	System dependent	
	EPRN	System dependent	
PVR	0:31	System dependent	Refer to *PowerPC 440EP Embedded Processor Data Sheet* for the PVR value
RSTCFG	U0	0	The U0 storage attribute has no effect in the PPC440EP.
	U1	0	Memory page contain normal instructions or data
	U2	0	Storage misses do not cause a line to allocated in the data cache
	U3	0	The U3 storage attribute has no effect in the PPC440EP.
	E	0	Memory pages are big endian
	ERPN	0	This value is hard wired to zero.
TCR	WRC	0b00	Watchdog Timer reset disabled

Table 8-1. Reset Values of Registers and Other 440EP Core Facilities (continued)

Resource	Field	Reset Value	Comment
TLBentry[1]	$EPN_{0:19}$	0xFFFFF000	Match EA of initial reset instruction ($EPN_{20:21}$ are undefined, as they are not compared to the EA because the page size is 4KB).
	V	1	Translation table entry for the initial program memory page is valid.
	TS	0	Initial program memory page is in system-level virtual address space.
	SIZE	0b0001	Initial program memory page size is 4KB.
	TID	0x00	Initial program memory page is globally shared; no match required against PID register.
	$RPN_{0:21}$	0xFFFFF || 0b00	Initial program memory page mapped effective = real.
	ERPN	0b0000	Extended real page number of the initial program memory page is set to 0b0000. Shadow TLB entry at reset is 0x0FFFFF000. The reset vector is 0x0FFFFFFFC for all boot configurations.
	U0	0	The U0 storage attribute has no effect in the PPC440EP.
	U1	0	Memory page contains normal instructions or data.
	U2	0	Storage misses do not cause a line to be allocated in the data cache.
	U3	0	The U3 storage attribute has no effect in the PPC440EP.
	W	0	Write-through storage attribute disabled.
	I	1	Caching inhibited storage attribute enabled.
	M	0	Memory coherent storage attribute disabled.
	G	1	Guarded storage attribute enabled.
	E	0	Reset value of endian storage attribute is specified by a core input signal.
	SX	1	Supervisor mode execution access enabled.
	SW	0	Supervisor mode write access disabled.
	SR	1	Supervisor mode read access enabled.
TSR	WRS	Copy of TCR[WRC]	If reset caused by Watchdog Timer
		Unchanged	If reset not caused by Watchdog Timer
		Undefined	After power-up

Note: "TLBentry" refers to an entry in the shadow instruction and data TLB arrays that is automatically configured by the PPC440EP to enable fetching and reading (but not writing) from the initial program memory page. This entry is not architecturally visible to software, and is invalidated upon any context synchronizing operation. Software must initialize a corresponding entry in the main unified TLB array before executing any operation which could lead to a context synchronization.

8.4 PPC440EP Chip Initialization

The PPC440EP chip configuration registers fall under two categories:

1. Clocking and power-on reset (CPR0) registers which are accessed indirectly through the CPR0_CFGADDR and CPR0_CFGDATA registers using the mtdcr and mfdcr instructions.
2. System device registers (SDR0) which are accessed indirectly through the SDR0_CFGADDR and SDR0_CFGDATA registers using the **mtdcr** and **mfdcr** instructions.

During PPC440EP initialization, some chip control registers must be initialized to ensure proper chip operation. Peripheral devices are also initialized as appropriate for the system design. Registers that control PPC440EP clocking and power-on reset are described in *Clocking* on page 283 while registers that control PPC440EP pin strapping are described in *Bootstrap Controller* on page 207. Initialization for peripheral devices is described in individual chapters as needed including chip level serial device control registers. The section that follows, lists registers that control miscellaneous PPC440EP devices that are not discussed in any other chapter.

8.4.1 Initial Configuration Register (CPR0_ICFG)

Figure 8-3. Initial Configuration Register (CPR0_ICFG)

0	RLI	Reload Inhibit 0 Reset CPR registers from configuration source defined by CPR0_ICFG[ICS] 1 Ignore CPR0_ICFG[ICS] configuration source and preserve current values in CPR0 registers (including CPR0_ICFG)	CPR0_ICFG[RLI] is used to preserve contents of CPR0 registers during a chip reset.
1:28		Reserved	
29:31	ICS	Initial values of the strap pins (UART0_DCD, UART0_DSR, UART0_CTS)	

8.4.2 System Device Control Registers

All SDR registers are accessed with the move-to-DCR (**mtdcr**) and move-from-DCR (**mfdcr**) instructions using indirect addressing. In indirect addressing, the **mtdcr** and **mfdcr** instructions access a given SDR register by means of its address offset which is contained in the SDR0_CFGADDR register. The data for the specified register is moved to or moved from the SDR0_CFGDATA register as described below. The SDR configuration registers are listed in *Table 8-2*. The offsets for all indirectly addressed DCR registers appear in *Table 4-3* on page 139.

Table 8-2. SDR Configuration Registers

Mnemonic	Register	Address	Access
SDR0_CFGADDR	SDR Configuration Address Register	0x000E	R/W
SDR0_CFGDATA	SDR Configuration Data Register	0x000F	R/W

Figure 8-4. SDR Configuration Address Register (SDR0_CFGADDR)

0:23			
24:31	DCRA	8-bit SDR Register Offset Value	This value can range from 0x00 to 0x7F. Its contents are used as the indirect DCR address for accessing an SDR register.

Figure 8-5. SDR Configuration Data Register (SDR0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contains the value of the SDR register as indicated by the SDR0_CFGADDR register.

8.4.2.1 Electronic Chip ID Register 0 (SDR0_ECID0)

The SDR0_ECIDx registers are for manufacturing use only. They uniquely identify each individual module manufactured using a 112-bit blown-fuse register.

Figure 8-6. Electronic Chip ID Register 0 (SDR0_ECID0)

0:31	ECID0	Electronic chip ID 0	Bits 0:31 of 112-bit ID

8.4.2.2 Electronic Chip ID Register 1 (SDR0_ECID1)

Figure 8-7. Electronic Chip ID Register 1 (SDR0_ECID1)

0:31	ECID1	Electronic chip ID 1	Bits 32:63 of 112-bit ID

8.4.2.3 Electronic Chip ID Register 2 (SDR0_ECID2)

Figure 8-8. Electronic Chip ID Register 1 (SDR0_ECID2)

0:31	ECID2	Electronic chip ID 2	Bits 64:95 of 112-bit ID

8.4.2.4 Electronic Chip ID Register 3 (SDR0_ECID3)

Figure 8-9. Electronic Chip ID Register 3 (SDR0_ECID3)

0:15	ECID3	Electronic chip ID 3	Bits 96:111 of 112-bit ID
16:31		Reserved	

8.4.2.5 Pin Function Control Register 0 (SDR0_PFC0)

Figure 8-10. Pin Function Control Register 0 (SDR0_PFC0)

0:22		Reserved	Bits 0:17 reset value = 0 Bits 18:22 reset value =1
23	TRE	GPIO Trace Enable 0 GPIO49-63 are enabled 1 GPIO49-63 are disabled	Trace interface cannot be used when GPIO is enabled. Trace interface can be used when GPIO is disabled. See the PPC440EP data sheet for pin multiplexing information. Reset value = SDR0_SDSTP1[CTE]
24:31		Reserved	Reset value = 0

8.4.2.6 Pin Function Control Register 1 (SDR0_PFC1)

Reset value = 0x0000100F.

Figure 8-11. Pin Function Control Register 1 (SDR0_PFC1)

0:5		Reserved	
6	U1ME	UART1 Mode Enable 0 Enable UART1DSR/CTS as DSR and UART1DTR/RTS as DTR 1 Enable UART1DSR/CTS as CTS and UART1DTR/RTS as RTS	See the PPC440EP data sheet for pin multiplexing information.
7:11		Reserved	
12	U0ME	UART0 Mode Enable 0 Enable UART0DSR/CTS as DSR and UART0DTR/RTS as DTR 1 Enable UART0DSR/CTS as CTS and UART0DTR/RTS as RTS	See the PPC440EP data sheet for pin multiplexing information. **Note:** U0ME must be set to 1 when U0IM = 0.
13	U0IM	UART0 Interface Mode 0 UART0 interface mode has 8 pins 1 UART0 interface mode has 4 pins	
14	SIS	SPI/IIC1 Selection 0 SPI enabled 1 IIC1 enabled	SIS controls the function of signals: [IIC1SClk]SCPClkOut [IIC1SData]SCPDI
15	UES	USB/EBC Selection 0 USB2D enabled 1 EBC enabled	UES controls the function of signals: [HoldPri]USB2LS1 [HoldReq]USB2RxAct
16	DUS	DMA/UIC Selection 0 DMA enabled 1 UIC enabled	DUS controls the function of signal: [DMAReq1]IRQ5
17	ERE	External Request Enable 0 Force 1b1 on EXT_REQ_EN input 1 GPIO0(27) input	To use the EBMI, set ERE to 1. To configure [ExtReq]USB2RxErr[GPIO27] as ExtReq see *Table 29-7* Alternate 1 Configuration for GPIO16-GPIO31.
18	UPR	Packet Reject Enable 0 USB2LS0 selected 1 RejectPkt selected	UPR controls the function of signal: [RejectPkt]USB2LS0
19	P3P4E	PLB Performance Monitor Selection: 0 PLB3 performance select 1 PLB4 performance select	The PLB Bus Monitor either monitors PLB3 or PLB4. P3P4E selects which bus is monitored. See Chapter 3 PLB Performance Monitor.
20:31		Reserved	

8.4.2.7 Slave Address Pipeline Register (SDR0_SLPIPE0)

Figure 8-12. Slave Address Pipeline Register (SDR0_SLPIPE0)

0		Reserved	
1	DAP	DDR Address Pipeline 0 DDR address pipeline disabled 1 DDR address pipeline enabled	
2	P4P3AP	PLB4 to PLB3 Address Pipeline 0 PLB4 to PLB3 address pipeline disabled 1 PLB4 to PLB3 address pipeline enabled	
3	P4OAP	PLB4 to OPB Address Pipeline 0 PLB4 to OPB address pipeline disabled 1 PLB4 to OPB address pipeline enabled	
4:27		Reserved	
28	P3OAP	PLB3 to OPB Address Pipeline 0 PLB3 to OPB address pipeline disabled 1 PLB3 to OPB address pipeline enabled	
29	EAP	EBC Address Pipeline 0 EBC address pipeline disabled 1 EBC address pipeline enabled	
30	PAP	PCI Address Pipeline 0 PCI address pipeline disabled 1 PCI address pipeline enabled	
31	P3P4AP	PLB3 to PLB4 Address Pipeline 0 PL3 to PLB4 address pipeline disabled 1 PLB3 to PLB4 address pipeline enabled	

8.4.2.8 Soft Reset Registers (SDR0_SRST0 and SDR0_SRST1)

Figure 8-13 and *Figure 8-14* describes the SDR0_SRST0 and SDR0_SRST1 register bits. For all bits, 1 = reset core, 0 = operational.

Reset value = 0 for SDR0_SRST0 and SDR0_SRST1.

Figure 8-13. Soft Reset Register 0 (SDR0_SRST0)

0	BGO	PLB to OPB bridge	
1	PLB4	PLB4 arbiter	
2	EBC	External bus controller	
3	OPB	OPB arbiter	
4	UART0	Universal asynchronous receiver/transmitter 0	
5	UART1	Universal asynchronous receiver/transmitter 1	
6	IIC0	Inter integrated circuit 0	
7	USB1H	USB1.1 host	
8	GPIO	General purpose I/O	
9	GPT	General purpose timer	
10	DMC	DDR SDRAM memory controller	

11	PCI	PCI	
12	EMAC0	Ethernet media access controller 0	
13	EMAC1	Ethernet media access controller 1	
14	CPM	Clock and power management	
15	ZMII	ZMII bridge	
16	UIC0	Universal interrupt controller 0	
17	UIC1	Universal interrupt controller 1	
18	IIC1	Inter integrated circuit 1	
19	SCP	Serial communications port	
20	BGI	OPB to PLB bridge	
21	DMA	Direct memory access controller	
22	DMAC	DMA channel	
23	MAL	Media access layer	
24	USB2D	USB2.0 device	
25	GPTR	General purpose timer	
26	PPM	PLB performance monitor	
27	P4P3	PLB4 to PLB3 bridge	
28	P3P4	PLB3 to PLB4 bridge	
29	PLB3	PLB3 arbiter	
30	UART2	Universal asynchronous receiver/transmitter 2	
31	UART3	Universal asynchronous receiver/transmitter 3	

Figure 8-14. Soft Reset Register 1(SDR0_SRST1)

0	NDFC	Nand flash controller	
1	OPBA1	OPB Arbiter attached to PLB4	
2	P42OPB	PLB4 to OPB	
3	DMA4	DMA to PLB4	
4	DMA4CH	DMA Channel to PLB4	
5:31		Reserved	

8.4.2.9 Miscellaneous Function Register (SDR0_MFR)

Reset value = 0 for bits 0:5 and 8:31.

Figure 8-15. Miscellaneous Function Register (SDR0_MFR)

0:3		Reserved	
4	E0CS	Ethernet 0 Clock Selection 0 Ethernet 0 selects external clock 1 Ethernet 0 selects internal reference clock (for internal loop back test mode)	SDR0_MFR[E0CS] is used in conjunction with EMAC0_MR1[ILE] **Note:** SDR0_MFR[E0CS] will change the clock even if it is not in internal loop back mode.

5	E1CS	Ethernet 1 Clock Selection 0 Ethernet 1 selects external clock 1 Ethernet 1 selects internal reference clock (for internal loop back test mode)	SDR0_MFR[E1CS] is used in conjunction with EMAC1_MR1[ILE] **Note:** SDR0_MFR[E1CS] will change the clock even if it is not in internal loop back mode.
6:7	ZMII	ZMII Mode 00 MII 01 SMII 10 RMII 10 11 RMII 100	Reset value = SDR0_SDSTP1[ZM]
8	E3E0	EMAC0 FIFO clock gating 0 Enable 1 Disable	Clock gating must be disabled (E3E0 = 1) when EMAC0 is enabled (EMAC0_MR0[TXE]=1, EMAC0_MR0[RXE] = 1). If clock gating (E3E0 = 0) is enabled and EMAC0 is inactive, the internal clock to the TX and RX FIFOs is gated. The default reset value is E3E0=0.
9	E3E1	EMAC1 FIFO clock gating 0 Enable 1 Disable	Clock gating must be disabled (E3E1 = 1) when EMAC1 is enabled (EMAC1_MR0[TXE] = 1, EMAC1_MR0[RXE] = 1). If clock gating is enabled (E3E1 = 0) and EMAC1 is inactive, the internal clock to the TX and RX FIFOs is gated The default reset value is E3E1=0.
10	PRP	Packet Reject Polarity 0 Packet reject signal is active low 1 Packet reject signal is active high	
11	PRE0	Packet Reject for EMAC0 Enable 0 Disable 1 Enable	
12	PRE1	Packet Reject for EMAC1 Enable 0 Disable 1 Enable	
13:31		Reserved	

8.4.2.10 Hardware Self Refresh Register (SDR0_HSF)

Figure 8-16. Hardware Self Refresh Register (SDR0_HSF)			
0:29		Reserved	
30	HSRR	Hardware Self Refresh Request	Reflects status of hardware self-refresh pin
31	HSRA	Hardware Self Refresh Acknowledge	Reflects status of DDR core hardware self-refresh ACK.

8.5 Initialization Software Requirements

After a reset operation occurs, the PPC440EP is initialized to a minimum configuration to enable the fetching and execution of the software initialization code, and to guarantee deterministic behavior of the core during the execution of this code. Initialization software is necessary to complete the configuration of the processor core and the rest of the on-chip and off-chip system.

The system must provide non-volatile memory (or memory initialized by some mechanism other than the PPC440EP) at the real address corresponding to effective address 0xFFFFFFFC, and at the rest of the initial program memory page. The instruction at the initial address must be an unconditional branch backwards to the beginning of the initialization software sequence.

The initialization software functions described in this section perform the configuration tasks required to prepare the PPC440EP to boot an operating system and subsequently execute an application program.

The initialization software must also perform functions associated with hardware resources that are outside the PPC440EP

Initialization software should perform the following tasks in order to fully configure the PPC440EP. For more information on the various functions referenced in the initialization sequence, see the corresponding chapters of this document.

1. Branch backwards from effective address 0xFFFFFFFC to the start of the initialization sequence
2. Invalidate the instruction cache (**iccci**)
3. Invalidate the data cache (**dccci**)
4. Synchronize memory accesses (**msync**)

   This step forces any data PLB operations that may have been in progress prior to the reset operation to complete, thereby allowing subsequent data accesses to be initiated and completed properly.
5. Clear DBCR0 register (disable all debug events)

   Although the PPC440EP is defined to reset some of the debug event enables during the reset operation (as specified in *Table 8-1* on page 192), this is not required by the architecture and hence the initialization software should not assume this behavior. Software should disable all debug events in order to prevent non-deterministic behavior on the trace interface to the core.
6. Clear DBSR register (initialize all debug event status)

   Although the PPC440EP is defined to reset the DBSR debug event status bits during the reset operation (as specified in *Table 8-1* on page 192), this is not required by the architecture and hence the initialization software should not assume this behavior. Software should clear all such status in order to prevent non-deterministic behavior on the JTAG interface to the core.
7. Initialize CCR0 register
   1. Enable/disable broadcast of instructions to auxiliary processor (save power if no AP attached)
   2. Enable/disable broadcast of trace information (save power if not tracing)
   3. Enable/configure or disable speculative instruction cache line prefetching
   4. Specify behavior for **icbt** and **dcbt**/**dcbtst** instructions
   5. Enable/disable gathering of separate store accesses
   6. Enable/disable hardware support for misaligned data accesses
   7. Enable/disable parity error recoverability (recoverability lowers load/store performance marginally.)
   8. Enable/disable cache read of parity bits depending on s/w compatibility requirements
8. Initialize CCR1 register
   1. Enable/disable full-line flushes as desired.
   2. Disable force cache-op miss (FCOM) and various parity error insertion (xxxPEI).
   3. Users may wish to initialize CCR1[TCS] here, or in the timer facilities section.
9. Configure instruction and data cache regions

   These steps must be performed prior to enabling the caches by setting the caching inhibited storage attribute of the corresponding TLB entry to 0.
   1. Clear the instruction and data cache normal victim index registers (INV0–INV3, DNV0–DNV3)

2. Clear the instruction and data cache transient victim index registers (ITV0–ITV3, DTV0–DTV3)
3. Set the instruction and data cache victim limit registers (IVLIM and DVLIM) according to the desired size of the normal, locked, and transient regions of each cache

10. Setup TLB entry to cover initial program memory page

Since the PPC440EP only initializes an architecturally-invisible shadow TLB entry during the reset operation, and since all shadow TLB entries are invalidated upon any context synchronization, special care must be taken during the initialization sequence to prevent any such context synchronizing operations (such as interrupts and the **isync** instruction) until after this step is completed, and an architected TLB entry has been established in the TLB. Particular care should be taken to avoid store operations, since write permission is disabled upon reset, and an attempt to execute any store operation would result in a Data Storage interrupt, thereby invalidating the shadow TLB entry.

1. Initialize MMUCR
   - Specify TID field to be written to TLB entries
   - Specify TS field to be used for TLB searches
   - Specify store miss allocation behavior
   - Enable/disable transient cache mechanism
   - Enable/disable cache locking exceptions
2. Write TLB entry for initial program memory page
   - Specify EPN, RPN, ERPN, and SIZE as appropriate for system
   - Set valid bit
   - Specify TID = 0 (disable comparison to PID) or else initialize PID register to matching value
   - Specify TS = 0 (system address space) or else MSR[IS,DS] must be set to correspond to TS=1
   - Specify storage attributes (W, I, M, G, E, U0–U3) as appropriate for system
   - Enable supervisor mode fetch, read, and write access (SX, SR, SW)
3. Initialize PID register to match TID field of TLB entry (unless using TID = 0)
4. Setup for subsequent MSR[IS,DS] initialization to correspond to TS field of TLB entry
   (This is necessary only if the TS field of the TLB entry is being set to 1 and MSR[IS,DS] is already reset to 0)
   - Write new MSR value into SRR1
   - Write address from which to continue execution into SRR0
5. Setup for subsequent change in instruction fetch address

   Only necessary if EPN field of TLB entry changed from the initial value ($EPN_{0:19} \neq 0xFFFFF$)
   - Write initial/new MSR value into SRR1
   - Write address from which to continue execution into SRR0
6. Fully initialize the TLB (tlbwe to all three words of each TLB entry; tlbre to TLB entries that are not fully initialized may result in parity exceptions).
7. Context synchronize to invalidate shadow TLB contents and cause new TLB contents to take effect
   - Use **isync** if not changing MSR contents and not changing the effective address of the rest of the initialization sequence
   - Use **rfi** if changing MSR to match new TS field of TLB entry (SRR1 will be copied into MSR, and program execution will resume at value in SRR0)

- Use **rfi** if changing next instruction fetch address to correspond to new EPN field of TLB entry (SRR1 will be copied into MSR, and program execution will resume at value in SRR0)

Instruction and data caches will now begin to be used, if the corresponding TLB entry has been setup with the caching inhibited storage attribute set to 0. Initialization software can now branch outside of the initial 4KB memory region as controlled by the address and size of the new TLB entry and/or any other TLB entries which have been setup.

11. Initialize interrupt resources
    1. Initialize IVPR to specify high-order address of the interrupt handling routines

       Make sure that the corresponding address region is covered by a TLB entry (or entries)
    2. Initialize IVOR0–IVOR15 registers (individual interrupt vector addresses)

       Make sure that the corresponding addresses are covered by a TLB entry (or entries)

       Because the low order four bits of IVOR0–IVOR15 are reserved, the values written to those bits are ignored when the registers are written, and are read as zero when the registers are used. Therefore, all interrupt vector offsets are implicitly aligned on quadword boundaries. Software must take care to assure that all interrupt handlers are quadword-aligned.
    3. Setup corresponding memory contents with the interrupt handling routines
    4. Synchronize any program memory changes as required. (Refer to the *PPC440 Processor User's Manual* for more information on the instruction sequence necessary to synchronize changes to program memory prior to executing the new instructions.)
12. Configure debug facilities as desired
    1. Write DBCR1 and DBCR2 to specify IAC and DAC event conditions
    2. Clear DBSR to initialize IAC auto-toggle status
    3. Initialize IAC1–IAC4, DAC1–DAC2, DVC1–DVC2 registers to desired values
    4. Write MSR[DWE] to enable Debug Wait mode (if desired)
    5. Write DBCR0 to enable desired debug mode(s) and event(s)
    6. Context synchronize to establish new debug facility context (**isync**)
13. Configure timer facilities as desired
    1. Write DEC to 0 to prevent Decrementer exception after TSR is cleared
    2. Write TBL to 0 to prevent Fixed Interval Timer and Watchdog Timer exceptions after TSR is cleared, and to prevent increment into TBH prior to full initialization
    3. CCR1[TCS] (Timer Clock Select) can be initialized here, or earlier with the rest of the CCR1.
    4. Clear TSR to clear all timer exception status
    5. Write TCR to configure and enable timers as desired

       Software must take care with respect to the enabling of the Watchdog Timer reset function, as once this function is enabled, it cannot be disabled except by reset itself
    6. Initialize TBH value as desired
    7. Initialize TBL value as desired
    8. Initialize DECAR to desired value (if enabling the auto-reload function)
    9. Initialize DEC to desired value

14. Initialize facilities outside the processor core which are possible sources of asynchronous interrupt requests (including DCRs and/or other memory-mapped resources)

    This must be done prior to enabling asynchronous interrupts in the MSR

15. Initialize the MSR to enable interrupts as desired
    1. Set MSR[CE] to enable/disable Critical Input and Watchdog Timer interrupts
    2. Set MSR[EE] to enable/disable External Input, Decrementer, and Fixed Interval Timer interrupts
    3. Set MSR[DE] to enable/disable Debug interrupts
    4. Set MSR[ME] to enable/disable Machine Check interrupts

       Software should first check the status of the ESR[MCI] field and MCSR[MCS] field to determine whether any Machine Check exceptions have occurred after these fields are cleared by reset and before Machine Check interrupts are enabled (by this step). Any such exceptions set ESR[MCI] or MCSR[MCS] to 1, and this status can only be cleared explicitly by software. After the MCSR[MCS] field is known to be clear, the MCSR status bits (MCSR[1:8]) should be cleared by software to avoid possible confusion upon later service of a machine check interrupt. Once MSR[ME] has been set to 1, subsequent Machine Check exceptions will result in a Machine Check interrupt.
    5. Context synchronize to establish new MSR context (**isync**)
16. Initialize any other processor core resources as required by the system (GPRs, SPRGs, and so on)
17. Initialize any other facilities outside the processor core as required by the system
18. Initialize system memory as required by the system software

    Synchronize any program memory changes as required. (Refer to the *PPC440 Processor User's Manual* for more information on the instruction sequence necessary to synchronize changes to program memory prior to executing the new instructions)
19. Start the system software

    System software is generally responsible for initializing and/or managing the rest of the MSR fields, including:
    1. MSR[FP] to enable or disable the execution of floating-point instructions
    2. MSR[FE0,FE1] to enable/disable Floating-Point Enabled exception type Program interrupts
    3. MSR[PR] to specify user mode or supervisor mode
    4. MSR[IS,DS] to specify application address space or system address space for instructions and data
    5. MSR[WE] to place the processor into Wait State (halt execution pending an interrupt)

9. Bootstrap Controller

In preparation for booting, the PPC440EP initializes several settings affecting system clocking, booting and other system features during a system reset. The initialization settings used in this process are provided by one of eight hardwired combinations on the three external bootstrap pins. Two of the hardwired strap combinations cause the initialization setting to be read by the IIC bootstrap controller, while the other six hardwired bootstrap combinations predefine the settings that are programmed into the initialization registers. The six hardwired bootstrap combinations that do not specify reading from the IIC interface configure the PPC440EP to operate at 333MHz or 400MHz and for booting from either an EBC, PCI, or NDFC attached ROM. If using the IIC bootstrap controller, the initialization settings are configured by the contents of an IIC serial ROM read by the IIC bootstrap controller.

9.1 Bootstrap Configuration

Bootstrap values include clock ratios, PLL parameters, boot settings and several other configurations affecting PCI, NDFC, Ethernet and pin sharing. Options selected during reset are used to initialize registers throughout the PPC440EP and are stored in the Serial Device Strap registers (SDR0_SDSTP0-SDR0_SDSTP3). *Table 9-1* lists the Serial Device Strap registers and the corresponding register bit fields initialized during reset.

Table 9-1. Bootstrappable Register Bit Fields

Serial Device Strap Register	Serial Device Strap Register Bit Field	Bit Field Initialized During Reset	Description
SDR0_SDSTP0	SDR0_SDSTP0[ENG]	CPR0_PLLC0[ENG]	Engage
	SDR0_SDSTP0[SRC]	CPR0_PLLC0[SRC]	PLL Feedback Source
	SDR0_SDSTP0[SEL]	CPR0_PLLC0[SEL]	Feedback Selection
	SDR0_SDSTP0[TUNE]	CPR0_PLLC0[TUNE]	Tune bits
	SDR0_SDSTP0[FBDV]	CPR0_PLLD0[FBDV]	PLL Feedback Divisor
	SDR0_SDSTP0[FWDVA]	CPR0_PLLD0[FWDVA]	PLL Forward Divisor A
	SDR0_SDSTP0[FWDVB]	CPR0_PLLD[FWDVB]	PLL Forward Divisor B
	SDR0_SDSTP0[PRBDV0]	CPR0_PRIMBD[PRBDV0]	PLL Primary Divisor B
	SDR0_SDSTP0[OPBDV0]	CPR0_OPBD[OPBDV0]	OPB Clock Divisor

Table 9-1. Bootstrappable Register Bit Fields (continued)

Serial Device Strap Register	Serial Device Strap Register Bit Field	Bit Field Initialized During Reset	Description
SDR0_SDSTP1	SDR0_SDSTP1[LFBDV]	CPR0_PLLD0[LFBDV]	PLL Local Feedback Divisor
	SDR0_SDSTP1[PERDV0]	CPR0_PERD[PERDV0]	Peripheral Clock Divisor 0
	SDR0_SDSTP1[MALD0]	CPR0_MALD[MALDV0]	MAL Clock Divisor 0
	SDR0_SDSTP1[SPCID0]	CPR0_SPCID[SPCID0]	PCI Clock Divisor
	SDR0_SDSTP1[PLLTIMER]		PLL TImer Length (CPR input)
	SDR0_SDSTP1[RW]	SDR0_EBC0[RW]	EBC ROM Width
	SDR0_SDSTP1[RL]	SDR0_CP440[RL]	ROM location (Boot source)
	SDR0_SDSTP1[PAE]	SDR0_PCI0[PAE]	PCI Arbiter Enable
	SDR0_SDSTP1[PHCE	SDR0_PCI0[PHCE]	PCI Host Configuration Enable
	SDR0_SDSTP1[ZM]	SDR0_MFR0[ZM]	ZMII Mode
	SDR0_SDSTP1[CTE]	SDR0_PFC0[CTE]	CPU Trace Enable
	SDR0_SDSTP1[Nto1]	SDR0_CP440[Nto1]	CPU PLB N to 1 clock ratio
	SDR0_SDSTP1[PAME]	SDR0_PCI0[PAME]	PCI Asynchronous Mode Enable
	SDR0_SDSTP1[RSV]	Reserved	Reserved
SDR0_SDSTP2	SDR0_SDSTP2[MEN]	SDR0_CUST0[MEN]	Multiplex EMAC or NDFC
	SDR0_SDSTP2[NE]	SDR0_CUST0[NE]	NDFC Enable
	SDR0_SDSTP2[NBW]	SDR0_CUST0[NBW]	NDFC Boot Width
	SDR0_SDSTP2[NBP]	SDR0_CUST0[NBP]	NDFC Boot Page
	SDR0_SDSTP2[NBAC]	SDR0_CUST0[NBAC]	NDFC Boot Address Cycle
	SDR0_SDSTP2[NARE]	SDR0_CUST0[NARE]	NDFC Auto read enable
	SDR0_SDSTP2[NRB]	SDR0_CUST0[NRB]	NDFC Ready/Busy bit
	SDR0_SDSTP2[NDRSC]	SDR0_CUST0[NDRSC]	NDFC Device Reset Count
	SDR0_SDSTP2[NCG]	SDR0_CUST0[NCG]	NDFC CS Gating
SDR0_SDSTP3	SDR0_SDSTP3[NDRDC]	SDR0_CUST1[NDRDC]	NDFC Device Read Count
	SDR0_SDSTP3[RSV]	SDR0_CUST1[RSV]	Reserved

9.2 Bootstrap Options

The PPC440EP has eight hardwired bootstrap options described in *Table 9-2*. The bootstrap pins are wired for the system reset default option. The six options A, B, C, D, E, and F provide predefined bootstrap configurations. Options G and H enable the IIC bootstrap controller and select the IIC address, 0b10101000 (0xA8) or 0b10100100 (0xA4) respectively.

The six hardwired bootstrap options A, B, C, D, E, and F include configurations for booting from either an EBC, PCI, or NDFC attached ROM. Options A, D, and E support booting from an EBC attached ROM with 8- or 16-bit data widths. Options B and F support booting over PCI. Option C supports booting over NDFC. *Table 9-3* contains descriptions of each bootstrap option.

If options G or H are wired, the controller reads 16 bytes from a serial ROM accessible from the IIC0 interface and stores them in the Serial Device Strap Registers. If the IIC bootstrap controller is unable to successfully read data from the serial ROM, the PPC440EP defaults to the X configuration. The settings for default configuration X are:

- SysClk - 33MHz
- VCO - 500MHz
- CPU - 250MHz
- PLB - 50MHz
- OPB - 25MHz
- PCI - 25MHz
- PerClk - 12.5MHz
- Boot ROM Location - EBC
- Boot width - 8 bits

The register settings for the default configuration X are defined in *Table 9-3*.

If the IIC bootstrap controller is disabled (strapping pins are wired for one of the options A—F), the PPC440EP is not limited to the selected bootstrap configuration. After setting of the default configuration is completed during the system reset, the software can change the initilization settings by following the procedure described in steps 1 through 3 below.

1. Configure the following CPR0 registers with the desired configuration:
   - CPR0_PLLC0
   - CPR0_PLLD0
   - CPR0_PRIMAD0
   - CPR0_PRIMAD
   - CPR0_PRIMBD
   - CPR0_OPBD
   - CPR0_PERD
   - CPR0_MALD
   - CPR0_SPCID
2. Set CPR0_ICFG[RLI] = 1. The Reload Inhibit bit preserves the configuration set in step 1 through a chip reset. The preserved register contents are used as the boot configuration after the chip reset.
3. Set DBCR0[RST] = 0b10 to generate a chip reset.

Table 9-2. Bootstrap Pins

$\overline{\text{UART0\_DCD}}$	$\overline{\text{UART0\_DSR}}$	$\overline{\text{UART0\_CTS}}$	Description
0	0	0	Bootstrap Option A SysClk - 33MHz VCO - 800MHz CPU - 400MHz PLB - 133MHz OPB - 66MHz PCI - 66MHz PerClk - 66MHz Boot ROM Location - EBC Boot width - 8 bits

Table 9-2. Bootstrap Pins (continued)

UART0_DCD	UART0_DSR	UART0_CTS	Description
0	0	1	Bootstrap Option B SysClk - 66MHz VCO - 666MHz CPU - 333MHz PLB - 133MHz OPB - 66MHz PCI - 66MHz PerClk - 33MHz Boot ROM Location - PCI
0	1	0	Bootstrap Option C SysClk - 33MHz VCO - 800MHz CPU - 400MHz PLB - 100MHz OPB - 50MHz PCI - 50MHz PerClk - 25MHz Boot ROM Location - NDFC
0	1	1	Bootstrap Option D SysClk - 33MHz VCO - 800MHz CPU - 400MHz PLB - 100MHz OPB - 50MHz PCI - 50MHz PerClk - 25MHz Boot ROM Location - EBC Boot width - 16 bits
1	0	0	Bootstrap Option E SysClk - 66MHz VCO - 666MHz CPU - 333MHz PLB - 133MHz OPB - 66MHz PCI - 66MHz PerClk - 33MHz Boot ROM Location - EBC Boot width - 16 bits
1	0	1	Bootstrap Option G IIC Bootstrap controller enabled, serial ROM address 0b10101000 (0xA8)
1	1	0	Bootstrap Option F SysClk - 33MHz VCO - 800MHz CPU - 400MHz PLB - 100MHz OPB - 50MHz PCI - 50MHz PerClk - 25MHz Boot ROM Location - PCI
1	1	1	Bootstrap Option H IIC Bootstrap controller enabled, serial ROM address 0b10100100 (0xA4)

Table 9-3. Bootstrap Options and Default X SDR0 Register Settings

Serial Device Strap Register Bit Field	Bootstrap Options						Default X	Description
	A	B	C	D	E	F		
SDR0_SDSTP0[ENG]	1	1	1	1	1	1	1	Engage 1 PLL's VCO is the source for the PLL forward dividers
SDR0_SDSTP0[SRC]	0	0	1	1	0	1	1	PLL Feedback Source 0 Feedback originates from PLLOutA 1 Feedback originates from PLLOutB
SDR0_SDSTP0[SEL]	000	000	000	000	000	000	000	Feedback Selection 000 PLL output (A or B)
SDR0_SDSTP0[TUNE]	11001 11100	01001 10110	11001 11100	11001 11100	01001 10110	11001 11100	10101 11000	PLL Tune bits 0100110110 9<M<=10 1100111100 22<M≤40
SDR0_SDSTP0[FBDV]	01100	00101	00011	00011	00101	00011	00011	PLL Feedback Divisor 01100 PLL Feedback Divisor = 12 00101 PLL Feedback Divisor = 5 00011 PLL Feedback Divisor = 3
SDR0_SDSTP0[FWDVA]	0010	0010	0010	0010	0010	0010	0010	PLL Forward Divisor A 0010 PLL Forward Divisor A = 2
SDR0_SDSTP0[FWDVB]	011	101	000	000	101	000	101	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 011 PLL Forward Divisor B = 3 101 PLL Forward Divisor B = 5
SDR0_SDSTP0[PRBDV0]	010	001	001	001	001	001	010	PLL Primary Divisor B 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 **Note:** Reset value for PLL Primary Divisor A is 1.
SDR0_SDSTP0[OPBDV0]	10	10	10	10	10	10	10	OPB Clock Divisor 0 10 OPB Clock Divisor 0 = 2
SDR0_SDSTP1[LFBDV]	000001	000001	000001	000001	000001	000001	000001	PLL Local Feedback Divisor 000001 PLL local feedback divisor = 1
SDR0_SDSTP1[PERDV0]	010	100	100	100	100	100	100	Peripheral Clock Divisor 0 010 Peripheral clock divisor = 2 100 Peripheral clock divisor = 4
SDR0_SDSTP1[MALDV0]	10	10	10	10	10	10	10	MAL Clock Divisor 0 10 MAL clock divisor 0 = 2
SDR0_SDSTP1[SPCID0]	10	10	10	10	10	10	10	SPCID0 10 (divide by 2)
SDR0_SDSTP1[PLLTIMER]	1111	1111	1111	1111	1111	1111	1111	Timer setting for PLL locking. Set the four upper bit of timer length value. The lower bits are 1
SDR0_SDSTP1[RW]	00	01	10	01	01	10	00	ROM boot width 00 ROM Boot Width = 8 bit 01 ROM Boot Width = 16 bit 10 ROM Boot Width = 32bit

Table 9-3. Bootstrap Options and Default X SDR0 Register Settings (continued)

Serial Device Strap Register Bit Field	Bootstrap Options						Default X	Description
	A	B	C	D	E	F		
SDR0_SDSTP1[RL]	00	01	10	00	00	01	00	ROM Location 00 EBC 01 PCI 10 NDFC
SDR0_SDSTP1[PAE]	0	0	0	0	0	0	0	PCI Internal Arbiter Enable 0 Disable
SDR0_SDSTP1[PHCE]	0	0	0	0	0	0	0	PCI Host Configuration Enable 0 Disable
SDR0_SDSTP1[ZM]	00	00	00	00	00	00	00	ZMII mode 00 MII mode
SDR0_SDSTP1[CTE]	0	0	0	0	0	0	0	CPU trace enable 0 Disable
SDR0_SDSTP1[N to 1]	1	0	1	1	0	1	1	CPU/PLB ration N/P 0 not N to 1 1 N to 1
SDR0_SDSTP1[PAME]	1	1	1	1	1	1	1	PCI Asynchronous mode enable 1 Enable
SDR0_SDSTP1[RSV]								Reserved (4 bits)
SDR0_SDSTP2[MEN]	01	01	10	01	01	01	01	Multiplex EMAC, NDFC, or GPIO 01 EMAC 10 NDFC 11 GPIO
SDR0_SDSTP2[NE]	0	0	1	0	0	0	0	NDFC Enable 0 Disable 1 Enable
SDR0_SDSTP2[NBW]	0	0	0	0	0	0	0	NDFC Boot Width 0 8 bits
SDR0_SDSTP2[NBP]	0000	0000	0000	0000	0000	0000	0000	NDFC Boot Page selection
SDR0_SDSTP2[NBAC]	00	00	01	00	00	00	00	NDFC Boot Address selection cycle 00 Three cycles, 512B per page 01 Four cycles, 512B per page
SDR0_SDSTP2[NARE]	0	0	1	0	0	0	1	NDFC Auto Read Disable 0 Disable 1 Enable
SDR0_SDSTP2[NRB]	0	0	0	0	0	0	0	NDFC Ready/Busy 0 Ready 1 Busy
SDR0_SDSTP2[NDRSC]	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 1011 0101	NDFC Device Reset Counter value
SDR0_SDSTP2[NCG]	0000	0000	1000	0000	0000	0000	0000	NDFC/EBC Chip select Gating

Table 9-3. Bootstrap Options and Default X SDR0 Register Settings (continued)

Serial Device Strap Register Bit Field	Bootstrap Options						Default X	Description
	A	B	C	D	E	F		
SDR0_SDSTP3[NDRDC]	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	NDFC Device Read Counter value
SDR0_SDSTP3[RSV]								Reserved (16 bits)
Note: If CPR0_ICGF[RLI] = 0, the reset value for PLL Primary Divisor A is 1 (CPR0_PRIMAD[PRADV0] = 1).								

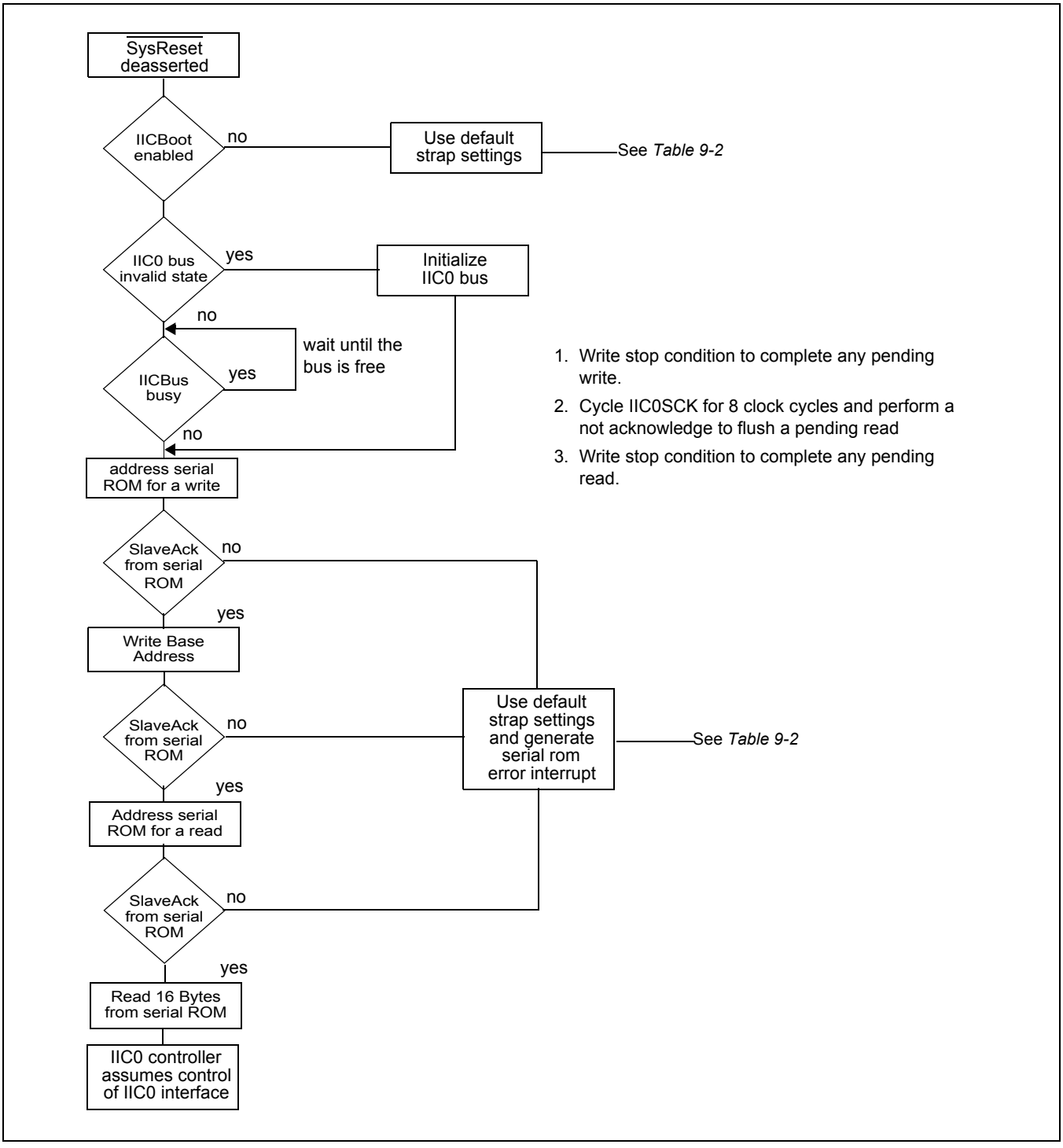
9.3 IIC Bootstrap Operations

After the deassertion of $\overline{\text{SysReset}}$, the IIC bootstrap controller tests the IIC bus to determine if the bus is in an invalid state or busy. If the bus is in an invalid state, it places the IIC0 bus in a known state through an initialization sequence and accesses the serial ROM as described in *Figure 9-1*. If the bus is busy, it waits until the bus is free before accessing the serial ROM. To access the ROM, the IIC bootstrap controller writes the base address (0x00) and then reads 16 bytes starting from the base address 0x00. The data in the serial ROM is organized as shown in *Table 9-4*.

If the IIC bootstrap controller is faster than the serial ROM, then the later can hold the IIC0 clock signal low until it is prepared for the next transaction. If the serial ROM does not acknowledge its address or the receipt of the base address, the PPC440EP defaults to configuration X as shown in *Table 9-3* and generates a serial ROM error interrupt, UIC1_SR [SRE]. Once interrupts are enabled, initialization software should check UIC1_SR[SRE] to ensure no errors occurred while accessing the serial ROM.

Figure 9-1 shows the IIC bootstrap controller flow.

Figure 9-1. IIC Bootstrap Controller Flow



9.3.1 Performance

The IIC bootstrap controller requires 188 IIC0_SCK clock cycles to read 16 bytes, 53 clock cycles for the first byte and 9 cycles for each of the remaining 15 bytes. The IIC0_SCK clock is generated directly from SysClk during system reset by dividing it by 768 (IIC0_SCK freq = SysClk/768).

Assuming a SysClk of 66.66 Mhz, the time needed to read the serial ROM is 2.1 mS. Serial ROM read time = 188*768/(SysClk freq)

Table 9-4. Serial ROM Memory Map

ROM Byte Address	Bit Number Within a Byte	Register[Field]
0	7 (MSb)	SDR0_SDSTP0[ENG]
	6	SDR0_SDSTP0[SRC]
	5:3	SDR0_SDSTP0[SEL]
	2:0 (LSb)	SDR0_SDSTP0[$TUNE_{5:7}$]
1	7:1	SDR0_SDSTP0[$TUNE_{8:14}$]
	0	SDR0_SDSTP0[$FBDV_{15}$]
2	7:4	SDR0_SDSTP0[$FBDV_{16:19}$]
	3:0	SDR0_SDSTP0[FBDVA]
3	7:5	SDR0_SDSTP0[FBDVB]
	4:2	SDR0_SDSTP0[PRBDV0]
	1:0	SDR0_SDSTP0[OPBDV0]
4	7:2	SDR0_SDSTP1[LFBDV]
	1:0	SDR0_SDSTP1[$PERDV0]_{0:1}$
5	7	SDR0_SDSTP1[$PERDV0]_2$
	6:5	SDR0_SDSTP1[MALDV0]
	4:3	SDR0_SDSTP1[PCID0]
	2:0	SDR0_SDSTP1[$PLLTIMER]_{0:2}$
6	7	SDR0_SDSTP1[$PLLTIMER]_3$
	6:5	SDR0_SDSTP1[RW]
	4:3	SDR0_SDSTP1[RL]
	2	SDR0_SDSTP1[PAE]
	1	SDR0_SDSTP1[PHCE]
	0	SDR0_SDSTP1[$ZM]_0$
7	7	SDR0_SDSTP1[$ZM]_1$
	6	SDR0_SDSTP1[CTE]
	5	SDR0_SDSTP1[Nto1]
	4	SDR0_SDSTP1[PAME]
	3:0	SDR0_SDSTP1[RSV]
8	7:6	SDR0_SDSTP2[MEN]
	5	SDR0_SDSTP2[NE]
	4	SDR0_SDSTP2[NBW]
	3:0	SDR0_SDSTP2[NBP]

Table 9-4. Serial ROM Memory Map (continued)

ROM Byte Address	Bit Number Within a Byte	Register[Field]
9	7:6	SDR0_SDSTP2[NBAC]
	5	SDR0_SDSTP2[NARE]
	4	SDR0_SDSTP2[NRB]
	3:0	SDR0_SDSTP2[NDRSC]$_{0:3}$
A	7:0	SDR0_SDSTP2[NDRSC]$_{4:11}$
B	7:4	SDR0_SDSTP2[NDRSC]$_{12:15}$
	3:0	SDR0_SDSTP2[NCG]
C	7:0	SDR0_SDSTP3[NDRDC]$_{0:7}$
D	7:0	SDR0_SDSTP3[NDRDC]$_{8:15}$
E	7:0	SDR0_SDSTP3[RSV]$_{0:7}$
F	7:0	SDR0_SDSTP3[RSV]$_{8:15}$

9.4 IIC Bootstrap Registers

All bootstrap registers are accessed indirectly through the SDR0_CFGADDR and SDR0_CFGDATA registers using the mtdcr and mfdcr instructions. *Table 9-5* lists the DCR addresses for the SDR0_CFGADDR and SDR0_CFGDATA registers.

Table 9-5. System DCR Register

Register	DCR Address	Access	Description
SDR0_CFGADDR	0x00E	R/W	System DCR Address Register
SDR0_CFGDATA	0x00F	R/W	System DCR Data Register

To read or write one of the bootstrap registers, software first writes the register address offset of the target register into the SDR0_CFGADDR register. The target register can then be read or written through the SDR0_CFGDATA DCR register. The following PowerPC code illustrates this procedure by reading the SDR0_SDSTP0 register.

```
li      r3, SDR0_CFGADDR      ! address offset of SDR0_CFGADDR
mtdcr   SDR0_CFGDATA, r3      ! set offset address
mfdcr   r4, SDR0_CFGDATA      ! read value of SDR0_CFGDATA
```

Note: *Table 9-6* lists the device control registers that are directly affected by the bootstrap initialization process. Other system device control registers that control PPC440EP miscellaneous devices are described in *Reset and Initialization* on page 189 while the clocking and power-on reset registers are described in *Clocking Registers* on page 294.

Table 9-6. IIC Bootstrap Controller Registers

Mnemonic	Register	Address	Access	Page
SDR0_PINSTP	Pin Strapping Register	0x0040	R	217
SDR0_SDCS0	Serial Device Controller Settings Register	0x0060	R	217

Table 9-6. IIC Bootstrap Controller Registers (continued)

Mnemonic	Register	Address	Access	Page
SDR0_SDSTP0	Serial Device Strap Register 0	0x0020	R	218
SDR0_SDSTP1	Serial Device Strap Register 1	0x0021	R	219
SDR0_SDSTP2	Read-Only Version of SDR0_CUST0	0x4001	R	220
SDR0_SDSTP3	Read-Only Version of SDR0_CUST1	0x4003	R	221
SDR0_CUST0	Customer Configuration Register 0	0x4000	R/W	221
SDR0_CUST1	Customer Configuration Register 1	0x4002	R/W	222
SDR0_EBC0	EBC Configuration Register	0x0100	R/W	222

9.4.1 Pin Strapping Register (SDR0_PINSTP)

The Pin Strapping Register (SDR0_PINSTP) records how the bootstrap pins are strapped during system reset. A copy of pin strap status is also maintained by *Initial Configuration Register (CPR0_ICFG)* on page 196.

Figure 9-2. Pin Strapping Register (SDR0_PINSTP)

0	U0DCD	$\overline{\text{UART0\_DCD}}$ Strapping 0 Strapped low 1 Strapped high	
1	U0DSR	$\overline{\text{UART0\_DSR}}$ Strapping 0 Strapped low 1 Strapped high	
2	U0CTS	$\overline{\text{UART0\_CTS}}$ Strapping 0 Strapped low 1 Strapped high	
3:31		Reserved	

9.4.2 Serial Device Controller Settings Register (SDR0_SDCS0)

The Serial Device Controller Settings Register contains the serial ROM address and status showing if the IIC Bootstrap Controller was enabled during system reset.

Figure 9-3. Serial Device Controller Settings Register (SDR0_SDCS0)

0:6	SBA	SEPROM Base Address: UART0_DSR=0 Base address 0xA8 0b1010100 UART0_DSR=1 Base address 0xA4 0b1010010	
7:29		Reserved	
30	SDC	Serial Device Control 0 Serial device control disabled 1 Serial device control enabled	
31	SDD	Serial Device Detection 0 Serial device detection disabled 1 Serial device detection enabled	SDR0_SDCR[SDD] = 1 if SDR0_SDCR[SDC] = 1

9.4.3 Serial Device Strap Register 0 (SDR0_SDSTP0)

SDR0_SDSTP0 is 32-bit read-only register. This register is reserved for system PLL and configuration information. The SDR0_SDSTP0 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap option A, B or C described in *Bootstrap Options* on page 208. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, configuration X is used as the default.

Figure 9-4. Serial Device Strap Register 0 (SDR0_SDSTP0)

0	ENG	Engage 0 SysClk is the source for PLL forward dividers. 1 PLL's VCO is the source for PLL forward dividers.	
1	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB	
2:4		Reserved	Only the default value 0b000 is allowed. Other values are for manufacturing use only.
5:14	TUNE	TUNE bits	
15:19	FBDV	PLL Feedback Divisor 00000 PLL Feedback Divisor = 32 00001 PLL Feedback Divisor = 1 00010 PLL Feedback Divisor = 2 00011 PLL Feedback Divisor = 3 00100 PLL Feedback Divisor = 4 .... 11111 PLL Feedback Divisor = 31	
20:23	FWDVA	PLL Forward Divisor A 0000 PLL Forward Divisor A = 16 0001 PLL Forward Divisor A = 1 0010 PLL Forward Divisor A = 2 0011 PLL Forward Divisor A = 3 0100 PLL Forward Divisor A = 4 0101 PLL Forward Divisor A = 5 0110 PLL Forward Divisor A = 6 0111 PLL Forward Divisor A = 7 1000 PLL Forward Divisor A = 8 1001 Reserved 1010 PLL Forward Divisor A = 10 1011 Reserved 1100 PLL Forward Divisor A = 12 1101 Reserved 1110 PLL Forward Divisor A = 14 1111 Reserved	
24:26	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 1 010 PLL Forward Divisor B = 2 011 PLL Forward Divisor B = 3 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 5 110 PLL Forward Divisor B = 6 111 PLL Forward Divisor B = 7	

27:29	PRBDV0	PLL Primary Divisor B 000 PLL Primary Divisor B = 8 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 011 PLL Primary Divisor B = 3 100 PLL Primary Divisor B = 4 101 PLL Primary Divisor B = 5 110 PLL Primary Divisor B = 6 111 PLL Primary Divisor B = 7	**Note:** If CPR0_ICFG[RLI] = 0, then the reset value for PLL Primary Divisor A is 1 (CPR0_PRIMAD[PRADV0]=1).
30:31	OPBDV0	OPB Clock Divisor 0 00 OPB clock divisor 0 = 4 01 OPB clock divisor 0 = 1 10 OPB clock divisor 0 = 2 11 OPB clock divisor 0 = 3	

9.4.4 Serial Device Strap Register 1 (SDR0_SDSTP1)

SDR0_SDSTP1 is 32-bit read-only register. This register is reserved for system PLL and configuration information. The SDR0_SDSTP1 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap option A, B or C described in *Bootstrap Options* on page 208. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, configuration X is used as the default.

Figure 9-5. Serial Device Strap Register 1 (SDR0_SDSTP1)

0:5	LFBDV	PLL Local Feedback Divisor 00_0000 PLL local feedback divisor = 64 00_0001 PLL local feedback divisor = 1 00_0010 PLL local feedback divisor = 2 .... 11_1111 PLL local feedback divisor = 63	The LFBDV is outside of PLL and is used to allow the PLL to lock using feedback directly from a PLL output. Program this divider to a value equivalent to the divide from the PLL to some clock to allow the PLL to be switched to use feedback from that clock. This allows the PLL to compensate for the latency associated with that clock's tree.
6:8	PERDV0	Peripheral Clock Divisor 0 000 Peripheral clock divisor 0 = 8 001 Peripheral clock divisor 0 = 1 010 Peripheral clock divisor 0 = 2 011 Peripheral clock divisor 0 = 3 100 Peripheral clock divisor 0 = 4 101 Peripheral clock divisor 0 = 5 110 Peripheral clock divisor 0 = 6 111 Peripheral clock divisor 0 = 7	
9:10	MALDV0	MAL Clock Divisor 0 00 MAL clock divisor 0 = 4 01 MAL clock divisor 0 = 1 10 MAL clock divisor 0 = 2 11 MAL clock divisor 0 = 3	
11:12	SPCID0	PCI Clock Divisor 0 00 PCI clock divisor 0 = 4 01 PCI clock divisor 0 = 1 10 PCI clock divisor 0 = 2 11 PCI clock divisor 0 = 3	
13:16	TS	Timer Setting (4 upper bits of the counter)	Length of timer for PLL locking. See also IIC bytes 5 & 6.

17:18	RW	EBC ROM Width 00 8-bit 01 16-bit 10 32-bit (Boot from Nand Flash) 11 Reserved	
19:20	RL	ROM Location 00 EBC 01 PCI 10 NDFC 11 Reserved	
21	PAE	PCI Internal Arbiter Enable 0 Disable 1 Enable	
22	PHCE	PCI Host Configuration Enable 0 Disable 1 Enable	
23:24	ZMII	ZMII mode 00 MII mode 01 SMII mode 10 RMII mode 10 11 RMII mode 100	
25	CTE	CPU trace enable 0 Disable 1 Enable	
26	Nto1	CPU/PLB ration N/P 0 not N to 1 1 N to 1	
27	PAME	PCI Asynchronous mode enable 0 Reserved 1 Enable	PAME = 0 is not supported.
28:31		Reserved	

9.4.5 Read-Only Equivalent of SDR0_CUST0 (SDR0_SDSTP2)

SDR0_SDSTP2 is a 32-bit read-only register. This register is reserved for user defined initialization data. The SDR0_SDSTP2 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap option A, B or C described in *Bootstrap Options* on page 208. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, configuration X is used as the default.

Figure 9-6. Serial Device Strap Register 2 (SDR0_SDSTP2)

0:1	MEN	Multiplex EMAC or NDFC or GPIO 01 EMAC 10 NDFC 11 GPIO	
2	NE	NDFC Enable 0 Disabled 1 Enabled	
3	NBW	NDFC Boot Width 0 8-bit 1 16-bit	Width of external device.
4:7	NBP	NDFC Boot Page Selection	
8:9	NBAC	NDFC Boot Address Selection Cycle	

10	NARE	NDFC Auto Read Disable 0 Disabled 1 Enabled	
11	NRB	NDFC Ready/Busy 0 Ready 1 Busy	
12:27	NDRSC	NDFC Device Reset Counter Value	
28:31	NCG	NDFC /EBC Chip select Gating Bit 28: CS0 (0 = EBC, 1= NDFC) Bit 29: CS1 Bit 30: CS2 Bit 31: CS3	Select either the EBC or the NDFC to control chip select pins.

9.4.6 Read-Only Version of SDR0_CUST1 (SDR0_SDSTP3)

SDR0_SDSTP3 is a 32-bit read-only register. This register is reserved for user defined initialization data. The SDR0_SDSTP3 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap option A, B or C described in *Bootstrap Options* on page 208. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, configuration X is used as the default.

Figure 9-7. Serial Device Strap Register 3 (SDR0_SDSTP3)

0:15	NDRSC	Device Read Count	
16:31		Reserved	

9.4.7 Customer Configuration Register 0 (SDR0_CUST0)

The Custom Configuration Register 0 (SDR0_CUST0) contains user defined initialization data. The bootstrap controller reads 8 bytes of user defined bootstraps and stores them in the SDR0_SDSTP2 register. The contents of read-only register SDR0_SDSTP2 initializes SDR0_CUST0.

Figure 9-8. Serial Device Strap Register 0 (SDR0_CUST0)

0:1	MEN	Multiplex EMAC or NDFC or GPIO 00 Reserved 01 EMAC 10 NDFC 11 GPIO	MEN selects the function of signals: EMCCD, EMC1RxErr[GPIO25][$\overline{\text{NFRdyBusy}}$] EMCDV, EMC1CrsDV[GPIO21][$\overline{\text{NFREn}}$] EMCTxD2, EMC1TxD0[GPIO18][NFCLE] EMCTxD3, EMC1TxD1[GPIO19][$\overline{\text{NFALE}}$] EMCTxErr, EMC1TxEn[GPIO23][$\overline{\text{NFWEn}}$] **Note:** NDFC is the NAND Flash Controller
2	NE	NDFC Enable 0 Disabled 1 Enabled	
3	NBW	NDFC Boot Width 0 8-bit 1 16-bit	Width of external device.
4:7	NBP	NDFC Boot Page Selection	
8:9	NBAC	NDFC Boot Address Selection Cycle	

10	NARE	NDFC Auto Read Disable 0 Disabled 1 Enabled	
11	NRB	NDFC Ready/Busy 0 Ready 1 Busy	
12:27	NDRSC	NDFC Device Reset Counter Value	
28	NCG0	NDFC /EBC Chip select (enable) gating Bit 28: CS0 (0 = EBC, 1= NDFC)	Select either the EBC or the NDFC to control chip select (enable) pins. **Note:** $\overline{\text{PerCSx}}$ (chip select) for EBC or $\overline{\text{NFCEx}}$ (chip enable) for NDFC
29	NCG1	NDFC /EBC Chip select (enable) gating Bit 29: CS1 (0 = EBC, 1= NDFC)	
30	NCG2	NDFC /EBC Chip select (enable) gating Bit 30: CS2(0 = EBC, 1= NDFC)	
31	NCG3	NDFC /EBC Chip select (enable) gating Bit 31: CS3 (0 = EBC, 1= NDFC)	

9.4.8 Custom Configuration Register 1 (SDR0_CUST1)

The Custom Configuration Register (SDR0_CUST1) contains user defined initialization data. The bootstrap controller reads bytes of user defined bootstraps and stores them in the SDR0_SDSTP3 register. The contents of read-only register SDR0_SDSTP3 initializes SDR0_CUST1.

Figure 9-9. Custom Configuration Register 1 (SDR0_CUST1)			
0:15	NDRSC	Device Read Count	
16:31		Reserved	Reserved for user's data

9.4.9 EBC Configuration Register (SDR0_EBC0)

The EBC Configuration Register (SDR0_EBC0) is initialized with values provided by the SDR0_SDSTP1[RW] bit field. This register configures EBC bank 0 peripheral width when booting from an EBC attached ROM.

Figure 9-10. EBC Configuration Register (SDR0_EBC0)			
0:1		Reserved	Reset value = 0
2:3	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 32-bit ROM 11 Reserved	When booting from NDFC, this value should be 10, as in boot configuration C. Since the actual device is hidden from the EBC, 32 bits indicates the width of the controller itself. Reset value = SDR0_SDSTP1[RW]
3:31		Reserved	Reset value = 0

10. Universal Interrupt Controller

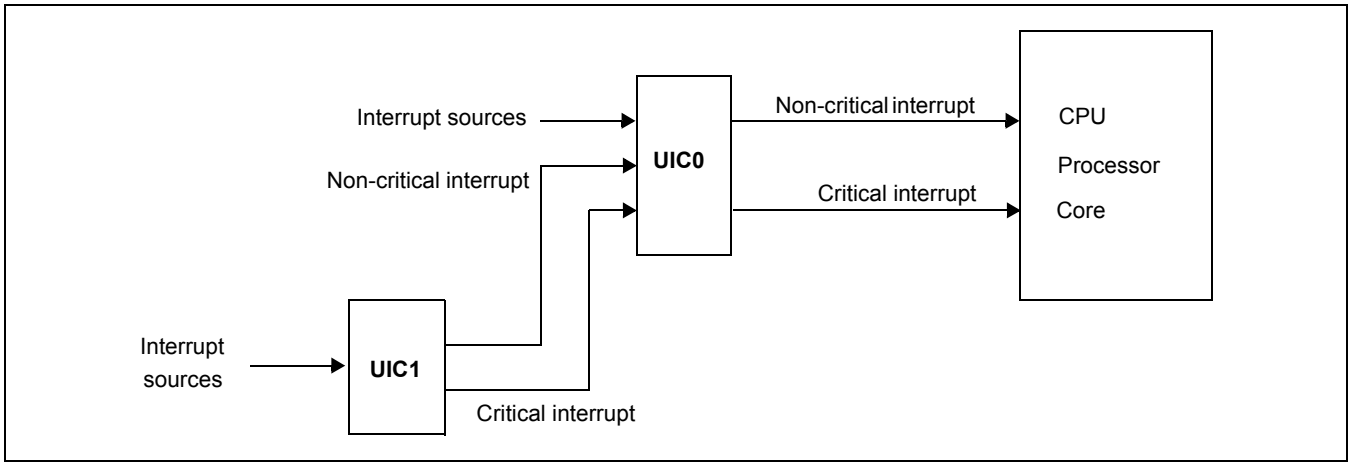
The PPC440EP contains two universal interrupt controllers (UIC0 and UIC1) that provide all necessary control, status, and communication between the various internal and external interrupt sources and the processor core.

10.1 UIC Overview

The UICs support 52 internal interrupts and 10 external interrupts. Status reporting (using the UIC Status Register [UIC*x*_SR]) is provided to ensure that systems software can determine the current and interrupting state of the system and respond appropriately. Software can generate interrupts to simplify software development and for diagnostics.

UIC0 collects interrupts from internal and external sources, including the critical and non-critical interrupt outputs of the secondary interrupt controller, UIC1. The UICs are cascaded as shown in *Figure 10-1* below:

Figure 10-1. Cascaded UIC Organization



The interrupts can be programmed, using the UIC Critical Register (UIC*x*_CR), to generate either a critical or a non-critical interrupt signal to the processor core.

The privileged **mtdcr** and **mfdcr** instructions, which are used by system software, are used to read and write the UIC registers.

An optional critical interrupt vector generator can reduce interrupt handling latency for critical interrupts. Vector calculation is described in detail in *UIC0 Vector Register (UIC0_VR)* on page 254.

10.2 UIC Features

- Support for 52 internal and 10 external interrupts
- Support for asynchronous level- or edge-sensitive interrupt types
- Programmable polarity for all interrupt types
- Programmable critical/non-critical interrupt selection for each interrupt bit
- Prioritized critical interrupt vector generation

- A UIC Status Register (UIC*x*_SR) providing the following information:
  - Current state of interrupts
  - Current state of all enabled interrupts (those masked using the UIC Enable Register (UIC*x*_ER))

10.3 UIC Interrupt Assignments

The UIC supports various internal and external interrupt sources as shown in *Table 10-1* and *Table 10-2*.

Table 10-1. UIC0 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Level	UART0
1	High	Level	UART1
2	High	Level	IIC0
3	High	Level	UART2
4	High	Level	UART3
5	High	Level	PCI CMD Write
6	High	Level	PCI Power Management
7	High	Level	IIC1
8	High	Level	SPI
9	Low	Level	External PCI SERR
10	High	Level	MAL TX EOB
11	High	Level	MAL RX EOB
12	High	Level	DMA Channel 0
13	High	Level	DMA Channel 1
14	High	Level	DMA Channel 2
15	High	Level	DMA Channel 3
16	High	Level	GPT Compare Timer 5
17	High	Level	GPT Compare Timer 6
18	High	Level	GPT Compare Timer 0
19	High	Level	GPT Compare Timer 1
20	High	Level	GPT Compare Timer 2
21	High	Level	GPT Compare Timer 3
22	High	Level	GPT Compare Timer 4
23	Programmable	Programmable	External IRQ 0
24	Programmable	Programmable	External IRQ 1
25	Programmable	Programmable	External IRQ 2

Table 10-1. UIC0 Interrupt Assignments (continued)

Interrupt	Polarity	Sensitivity	Interrupt Source
26	Programmable	Programmable	External IRQ 3
27	Programmable	Programmable	External IRQ 4
28	Programmable	Programmable	External IRQ 5
29	Programmable	Programmable	External IRQ 6
30	High	Level	UIC1 Non-Critical interrupt
31	High	Level	UIC1 Critical Interrupt

Table 10-2. UIC1 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Level	MAL SERR
1	High	Level	MAL TXDE
2	High	Level	MAL RXDE
3	High	Level	DDRSDRAM ECC Uncorrectable Error
4	High	Level	DDRSDRAM ECC Correctable Error
5	High	Edge	External Bus Controller Master
6	High	Edge	NDFC Interrupt
7	High	Level	OPB to PLB Bridge
8	High	Level	USB1.1 Host Interrupt 1
9	High	Level	USB 1.1 Host Interrupt 2 (SMI)
10	High	Level	PLB3x4x MIRQ 0
11	High	Level	PLB3x4x MIRQ 1
12	High	Level	PLB3x4x MIRQ 2
13	High	Level	PLB3x4x MIRQ 3
14	High	Level	PLB3x4x MIRQ 4
15	High	Level	PLB3x4x MIRQ 5
16	High	Level	UDMA Interrupt 0
17	High	Level	UDMA Interrupt 1
18	Programmable	Programmable	External IRQ 7
19	Programmable	Programmable	External IRQ 8
20	Programmable	Programmable	External IRQ 9
21	High	Level	UDMA Interrupt 2
22	High	Level	UDMA Interrupt 3
23	Low	Edge	USB1.1/USB 2.0 Device Interrupt
24	High	Level	Serial ROM
25	Low	Edge	GPT Decrement pulse
26	High	Level	PLB Performance Monitor
27	Low	Level	EXT_PCI_PERR (Parity)

Table 10-2. UIC1 Interrupt Assignments (continued)

Interrupt	Polarity	Sensitivity	Interrupt Source
28	High	Level	EMAC 0 Interrupt
29	High	Level	EMAC 0 Wake-up
30	High	Level	EMAC 1 Interrupt
31	High	Level	EMAC 1 Wake-up

10.4 Interrupt Programmability

The on-chip interrupts and the external IRQs are programmable. However, the polarity and sensitivity of the on-chip interrupts must be programmed as shown in *Table 10-1 UIC0 Interrupt Assignments* on page 224 and *Table 10-2 UIC1 Interrupt Assignments* on page 225, using the UIC Polarity Register (UIC*x*_PR) and UIC Trigger Register (UIC*x*_TR), respectively.

10.5 UIC Registers

The UIC includes the Device Control Registers (DCRs) listed in *Table 10-3*. The registers are accessed using the **mfdcr** and **mtdcr** instructions.

Table 10-3. UIC Device Control Registers

Mnemonic	Register	Address	Access	Page
UIC0_SR	UIC Status Register 0	0x0C0	Read/Clear	226
UIC1_SR	UIC Status Register 1	0x0D0	Read/Clear	228
UIC0_ER	UIC Enable Register 0	0x0C2	R/W	230
UIC1_ER	UIC Enable Register 1	0x0D2	R/W	233
UIC0_CR	UIC Critical Register 0	0x0C3	R/W	235
UIC1_CR	UIC Critical Register 1	0x0D3	R/W	237
UIC0_PR	UIC Polarity Register 0	0x0C4	R/W	239
UIC1_PR	UIC Polarity Register 1	0x0D4	R/W	241
UIC0_TR	UIC Trigger Register 0	0x0C5	R/W	244
UIC1_TR	UIC Trigger Register 1	0x0D5	R/W	246
UIC0_MSR	UIC Masked Status Register 0	0x0C6	Read-only	248
UIC1_MSR	UIC Masked Status Register 1	0x0D6	Read-only	250
UIC0_VR	UIC Vector Register 0	0x0C7	Read-only	254
UIC1_VR	UIC Vector Register 1	0x0D7	Read-only	255
UIC0_VCR	UIC Vector Configuration Register 0	0x0C8	Write-only	253
UIC1_VCR	UIC Vector Configuration Register 1	0x0D8	Write-only	253

10.5.1 UIC0 Status Register (UIC0_SR)

To report interrupt status, the UIC0_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC0_SR fields.

Figure 10-2. UIC0 Status Register (UIC0_SR)			
0	U0	UART0 Interrupt Status 0 UART0 interrupt has not occurred. 1 UART0 interrupt occurred.	
1	U1	UART1 Interrupt Status 0 UART1 interrupt has not occurred. 1 UART1 interrupt occurred.	
2	IIC0	IIC0 Interrupt Status 0 IIC0 interrupt has not occurred. 1 IIC0 interrupt occurred.	
3	U2	UART2 Interrupt Status 0 UART2 interrupt has not occurred. 1 UART2 interrupt occurred.	
4	U3	UART3 Interrupt Status 0 UART3 interrupt has not occurred. 1 UART3 interrupt occurred.	
5	PCRW	PCI Command Register Write Interrupt Status 0 PCI command register write interrupt has not occurred. 1 PCI command register write interrupt occurred.	
6	PPM	PCI Power Management Interrupt Status 0 PCI power management interrupt has not occurred. 1 PCI power management interrupt occurred.	
7	IIC1	IIC1 Interrupt Status 0 IIC1 interrupt has not occurred. 1 IIC1 interrupt occurred.	
8	SPI	SPI Interrupt Status 0 SPI interrupt has not occurred. 1 SPI interrupt occurred.	
9	EPS	Ext PCI SERR Interrupt Status 0 Ext PCI SERR interrupt has not occurred. 1 Ext PCI SERR interrupt occurred.	
10	MTE	MAL TX EOB Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.	
11	MRE	MAL RX EOB Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.	
12	D0	DMA2P30 Channel 0 Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.	
13	D1	DMA2P30 Channel 1 Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.	
14	D2	DMA2P30 Channel 2 Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.	
15	D3	DMA2P30 Channel 3 Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.	
16	CT5	GPT Compare Timer 5 Interrupt Status 0 Compare timer 5 interrupt has not occurred. 1 Compare timer 5 interrupt occurred.	

17	CT6	GPT Compare Timer 6 Interrupt Status 0 Compare timer 6 interrupt has not occurred. 1 Compare timer 6 interrupt occurred.	
18	CT0	GPT Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.	
19	CT1	GPT Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.	
20	CT2	GPT Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.	
21	CT3	GPT Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.	
22	CT4	GPT Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.	
23	EIR0	External IRQ 0 Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.	
24	EIR1	External IRQ 1 Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.	
25	EIR2	External IRQ 2 Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.	
26	EIR3	External IRQ 3 Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.	
27	EIR4	External IRQ 4 Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.	
28	EIR5	External IRQ 5 Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.	
29	EIR6	External IRQ 6 Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.	
30	UIC1NC	UIC1 Non-Critical Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.	
31	UIC1C	UIC1 Critical Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.	

10.5.2 UIC1 Status Register (UIC1_SR)

To report interrupt status, the UIC1_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC1_SR fields.

Figure 10-3. UIC1 Status Register (UIC1_SR)

0	MS	MAL SERR Interrupt Status 0 MAL SERR interrupt has not occurred. 1 MAL SERR interrupt occurred.	
1	MTDE	MAL TXDE Interrupt Status 0 MAL TXDE interrupt has not occurred. 1 MAL TXDE interrupt occurred.	
2	MRDE	MAL RXDE Interrupt Status 0 MAL RXDE interrupt has not occurred. 1 MAL RXDE interrupt occurred.	
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Status 0 DDRSDRAM ECC uncorrectable error interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error interrupt occurred.	
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Status 0 DDRSDRAM ECC correctable error interrupt has not occurred. 1 DDRSDRAM ECC correctable error interrupt occurred.	
5	EBC	EBC Interrupt Status 0 EBC interrupt has not occurred. 1 EBC interrupt occurred.	
6	NDFC	NDFC Interrupt Status 0 NDFC interrupt has not occurred. 1 NDFC interrupt occurred.	
7	OPB	OPB to PLB Bridge Interrupt Status 0 OPB to PLB bridge interrupt has not occurred. 1 OPB to PLB bridge interrupt occurred.	
8	USB1H1	USB1.1 Host 1 Interrupt Status 0 USB1.1 Host 1 interrupt has not occurred. 1 USB1.1 Host 1 interrupt occurred.	
9	USB1H2	USB1.1 Host 2 Interrupt Status 0 USB1.1 Host 2 interrupt has not occurred. 1 USB1.1 Host 2 interrupt occurred.	
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Status 0 PLB3 to PLB4 bridge 0 interrupt has not occurred. 1 PLB3 to PLB4 bridge 0 interrupt occurred.	
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Status 0 PLB3 to PLB4 bridge 1 interrupt has not occurred. 1 PLB3 to PLB4 bridge 1 interrupt occurred.	
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Status 0 PLB3 to PLB4 bridge 2 interrupt has not occurred. 1 PLB3 to PLB4 bridge 2 interrupt occurred.	
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Status 0 PLB3 to PLB4 bridge 3 interrupt has not occurred. 1 PLB3 to PLB4 bridge 3 interrupt occurred.	
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Status 0 PLB3 to PLB4 bridge 4 interrupt has not occurred. 1 PLB3 to PLB4 bridge 4 interrupt occurred.	
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Status 0 PLB3 to PLB4 bridge 5 interrupt has not occurred. 1 PLB3 to PLB4 bridge 5 interrupt occurred.	

16	UDMA0	UDMA0 Interrupt Status 0 UDMA0 interrupt has not occurred. 1 UDMA0 interrupt occurred.	DMA2P40 interrupt 0 (USB 2.0 device endpoint 1, DMA in).
17	UDMA1	UDMA1 Interrupt Status 0 UDMA1 interrupt has not occurred. 1 UDMA1 interrupt occurred.	DMA2P40 interrupt 1 (USB 2.0 device endpoint 2, DMA in).
18	EIR7	External IRQ 7 Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.	
19	EIR8	External IRQ 8 Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.	
20	EIR9	External IRQ 9 Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.	
21	UDMA2	UDMA2 Interrupt Status 0 UDMA2 interrupt has not occurred. 1 UDMA2 interrupt occurred.	DMA2P40 interrupt 2 (USB 2.0 device endpoint 1, DMA out).
22	UDMA3	UDMA3 Interrupt Status 0 UDMA3 interrupt has not occurred. 1 UDMA3 interrupt occurred.	DMA2P40 interrupt 3 (USB 2.0 device endpoint 2, DMA out).
23	USBD	USB1.1 USB2.0 Device Interrupt Status 0 USB1.1 USB 2.0 device interrupt has not occurred. 1 USB1.1 USB2.0 device interrupt occurred.	
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.	
25	GDP	GPT Decrement Pulse Interrupt Status 0 GPT decrement pulse interrupt has not occurred. 1 GPT decrement pulse interrupt occurred.	
26	PPM	PLB Performance Monitor Interrupt Status 0 PLB performance monitor interrupt has not occurred. 1 PLB performance monitor interrupt occurred.	
27	EPP	EXT_PCI_PERR (parity) Interrupt Status 0 EXT_PCI_PERR (parity) interrupt has not occurred. 1 EXT_PCI_PERR (parity) interrupt occurred.	
28	ETH0	Ethernet 0 Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.	
29	EWU0	Ethernet 0 Wake-up Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.	
30	ETH1	Ethernet 1 Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.	
31	EWU1	Ethernet 1 Wake-up Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up interrupt occurred.	

10.5.3 UIC0 Enable Register (UIC0_ER)

The fields of the UIC0_ER, which correspond to the fields of the UIC0_SR, enable or disable the reporting of the corresponding fields of the UIC0_SR.

If a UIC0_ER field is set to 1, the corresponding field of the UIC0_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC0_SR field is set to 1. If a UIC0_ER field is set to 0, the corresponding field of the UIC0_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC0_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC0_CR.

Figure 10-4. UIC0 Enable Register (UIC0_ER)

0	U0	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.	
1	U1	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.	
2	IIC0	IIC0 Interrupt Enable 0 IIC0 interrupt is disabled. 1 IIC0 interrupt is enabled.	
3	U2	UART2 Interrupt Enable 0 UART2 interrupt is disabled. 1 UART2 interrupt is enabled.	
4	U3	UART3 Interrupt Enable 0 UART3 interrupt is disabled. 1 UART3 interrupt is enabled.	
5	PCRW	PCI Command Register Write Interrupt Enable 0 PCI command register write interrupt is disabled. 1 PCI command register write interrupt is enabled.	
6	PPM	PCI Power Management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.	
7	IIC1	IIC1 Interrupt Enable 0 IIC1 interrupt is disabled. 1 IIC1 interrupt is enabled.	
8	SPI	SPI Interrupt Enable 0 SPI interrupt is disabled. 1 SPI interrupt is enabled.	
9	EPS	Ext PCI SERR Interrupt Enable 0 Ext PCI SERR interrupt is disabled. 1 Ext PCI SERR interrupt is enabled.	
10	MTE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt is enabled.	
11	MRE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt is enabled.	
12	D0	DMA2P30 Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.	
13	D1	DMA2P30 Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.	

14	D2	DMA2P30 Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.	
15	D3	DMA2P30 Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.	
16	CT5	GPT Compare Timer 5 Interrupt Enable 0 Compare timer 5 interrupt is disabled. 1 Compare timer 5 interrupt is enabled.	
17	CT6	GPT Compare Timer 6 Interrupt Enable 0 Compare timer 6 interrupt is disabled. 1 Compare timer 6 interrupt is enabled.	
18	CT0	GPT Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt is disabled. 1 Compare timer 0 interrupt is enabled.	
19	CT1	GPT Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt is disabled. 1 Compare timer 1 interrupt is enabled.	
20	CT2	GPT Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt is disabled. 1 Compare timer 2 interrupt is enabled.	
21	CT3	GPT Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt is disabled. 1 Compare timer 3 interrupt is enabled.	
22	CT4	GPT Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt is disabled. 1 Compare timer 4 interrupt is enabled.	
23	EIR0	External IRQ 0 Interrupt Enable 0 External IRQ 0 interrupt is disabled. 1 External IRQ 0 interrupt is enabled.	
24	EIR1	External IRQ 1 Interrupt Enable 0 External IRQ 1 interrupt is disabled. 1 External IRQ 1 interrupt is enabled.	
25	EIR2	External IRQ 2 Interrupt Enable 0 External IRQ 2 interrupt is disabled. 1 External IRQ 2 interrupt is enabled.	
26	EIR3	External IRQ 3 Interrupt Enable 0 External IRQ 3 interrupt is disabled. 1 External IRQ 3 interrupt is enabled.	
27	EIR4	External IRQ 4 Interrupt Enable 0 External IRQ 4 interrupt is disabled. 1 External IRQ 4 interrupt is enabled.	
28	EIR5	External IRQ 5 Interrupt Enable 0 External IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.	
29	EIR6	External IRQ 6 Interrupt Enable 0 External IRQ 6 interrupt is disabled. 1 External IRQ 6 interrupt is enabled.	
30	UIC1NC	UIC1 Non-Critical Interrupt Enable 0 UICNC interrupt is disabled. 1 UICNC interrupt is enabled.	
31	UIC1C	UIC1 Critical Interrupt Enable 0 UICC interrupt is disabled. 1 UICC interrupt is enabled.	

10.5.4 UIC1 Enable Register (UIC1_ER)

The fields of the UIC1_ER, which correspond to the fields of the UIC1_SR, enable or disable the reporting of the corresponding fields of the UIC1_SR.

If a UIC1_ER field is set to 1, the corresponding field of the UIC1_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC1_SR field is set to 1. If a UIC1_ER field is set to 0, the corresponding field of the UIC1_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC1_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC1_CR.

Figure 10-5. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.	
1	MTDE	MAL TXDE Interrupt Enable 0 MAL TXDE interrupt is disabled. 1 MAL TXDE interrupt is enabled.	
2	MRDE	MAL RXDE Interrupt Enable 0 MAL RXDE interrupt is disabled. 1 MAL RXDE interrupt is enabled.	
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Enable 0 DDRSDRAM ECC uncorrectable error interrupt is disabled. 1 DDRSDRAM ECC uncorrectable error interrupt is enabled.	
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Enable 0 DDRSDRAM ECC correctable error interrupt is disabled. 1 DDRSDRAM ECC correctable error interrupt is enabled.	
5	EBC	EBC Interrupt Enable 0 EBC interrupt is disabled. 1 EBC interrupt is enabled.	
6	NDFC	NDFC Interrupt Enable 0 NDFC interrupt is disabled. 1 NDFC interrupt is enabled.	
7	OPB	OPB to PLB Bridge Interrupt Enable 0 OPB to PLB bridge interrupt is disabled. 1 OPB to PLB bridge interrupt is enabled.	
8	USB1H1	USB1.1 Host 1 Interrupt Enable 0 USB1.1 Host 1 interrupt is disabled. 1 USB1.1 Host 1 interrupt is enabled.	
9	USB1H2	USB1.1 Host 2 Interrupt Enable 0 USB1.1 Host 2 interrupt is disabled. 1 USB1.1 Host 2 interrupt is enabled.	
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Enable 0 PLB3 to PLB4 bridge 0 interrupt is disabled. 1 PLB3 to PLB4 bridge 0 interrupt is enabled.	

11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Enable 0 PLB3 to PLB4 bridge 1 interrupt is disabled. 1 PLB3 to PLB4 bridge 1 interrupt is enabled.	
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Enable 0 PLB3 to PLB4 bridge 2 interrupt is disabled. 1 PLB3 to PLB4 bridge 2 interrupt is enabled.	
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Enable 0 PLB3 to PLB4 bridge 3 interrupt is disabled. 1 PLB3 to PLB4 bridge 3 interrupt is enabled.	
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Enable 0 PLB3 to PLB4 bridge 4 interrupt is disabled. 1 PLB3 to PLB4 bridge 4 interrupt is enabled.	
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Enable 0 PLB3 to PLB4 bridge 5 interrupt is disabled. 1 PLB3 to PLB4 bridge 5 interrupt is enabled.	
16	UDMA0	UDMA0 Interrupt Enable 0 UDMA0 interrupt is disabled. 1 UDMA0 interrupt is enabled.	DMA2P40 USB 2.0 device Endpoint 1 In.
17	UDMA1	UDMA1 Interrupt Enable 0 UDMA1 interrupt is disabled. 1 UDMA1 interrupt is enabled.	DMA2P40 USB 2.0 device Endpoint 2 In.
18	EIR7	External IRQ 7 Interrupt Enable 0 External IRQ 7 interrupt is disabled. 1 External IRQ 7 interrupt is enabled.	
19	EIR8	External IRQ 8 Interrupt Enable 0 External IRQ 8 interrupt is disabled. 1 External IRQ 8 interrupt is enabled.	
20	EIR9	External IRQ 9 Interrupt Enable 0 External IRQ 9 interrupt is disabled. 1 External IRQ 9 interrupt is enabled.	
21	UDMA2	UDMA2 Interrupt Enable 0 UDMA2 interrupt is disabled. 1 UDMA2 interrupt is enabled.	DMA2P40 USB 2.0 device Endpoint 1 Out.
22	UDMA3	UDMA3 Interrupt Enable 0 UDMA3 interrupt is disabled. 1 UDMA3 interrupt is enabled.	DMA2P40 USB 2.0 device Endpoint 2 Out.
23	USBD	USB1.1 USB2.0 Device Interrupt Enable 0 USB1.1 USB2.0 Device interrupt is disabled. 1 USB1.1 USB2.0 Device interrupt is enabled.	
24	SRE	Serial ROM Error Interrupt Enable 0 Serial ROM error interrupt is disabled. 1 Serial ROM error interrupt is enabled.	
25	GDP	GPT Decrement Pulse Interrupt Enable 0 GPT decrement pulse interrupt is disabled. 1 GPT decrement pulse interrupt is enabled.	
26	PPM	PLB Performance Monitor Interrupt Enable 0 PLB performance monitor interrupt is disabled. 1 PLB performance monitor interrupt is enabled.	
27	EPP	EXT_PCI_PERR (parity) Interrupt Enable 0 EXT_PCI_PERR (parity) interrupt is disabled. 1 EXT_PCI_PERR (parity) interrupt is enabled.	
28	ETH0	Ethernet 0 Interrupt Enable 0 Ethernet 0 interrupt is disabled. 1 Ethernet 0 interrupt is enabled.	

29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Ethernet 0 wake-up interrupt is disabled. 1 Ethernet 0 wake-up interrupt is enabled.	
30	ETH1	Ethernet 1 Interrupt Enable 0 Ethernet 1 interrupt is disabled. 1 Ethernet 1 interrupt is enabled.	
31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Ethernet 1 wake-up interrupt is disabled. 1 Ethernet 1 wake-up interrupt is enabled.	

10.5.5 UIC0 Critical Register (UIC0_CR)

The fields of the UIC0_CR, which correspond to the fields of the UIC0_SR and UIC0_ER, determine whether an interrupt captured in the corresponding fields of the UIC0_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC0_ER. The processor handles critical interrupts when MSR[EE] = 1 and non-critical interrupts when MSR[CE]=1.

If a UIC0_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC0_SR and enabled in the corresponding field of the UIC0_ER) generates a non-critical interrupt signal to the processor core. If a UIC0_CR field is a 1, a critical interrupt signal is generated.

Figure 10-6. UIC0 Critical Register (UIC0_CR)

0	U0	UART0 Interrupt Class 0 UART0 interrupt is non-critical. 1 UART0 interrupt is critical.	
1	U1	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.	
2	IIC0	IIC0 Interrupt Class 0 IIC0 interrupt is non-critical. 1 IIC0 interrupt is critical.	
3	U2	UART2 Interrupt Class 0 UART2 interrupt is non-critical. 1 UART2 interrupt is critical.	
4	U3	UART3 Interrupt Class 0 UART3 interrupt is non-critical. 1 UART3 interrupt is critical.	
5	PCRW	PCI Command Register Write Interrupt Class 0 PCI command register write interrupt is non-critical. 1 PCI command register write interrupt is critical.	
6	PPM	PCI Power Management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.	
7	IIC1	IIC1 Interrupt Class 0 IIC1 interrupt is non-critical. 1 IIC1 interrupt is critical.	
8	SPI	SPI Interrupt Class 0 SPI interrupt is non-critical. 1 SPI interrupt is critical.	
9	ESP	Ext PCI SERR Interrupt Class 0 Ext PCI SERR interrupt is non-critical. 1 Ext PCI SERR interrupt is critical.	

10	MTE	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt is critical.	
11	MRE	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt is critical.	
12	D0	DMA2P30 Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.	
13	D1	DMA2P30 Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.	
14	D2	DMA2P30 Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.	
15	D3	DMA2P30 Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.	
16	CT5	GPT Compare Timer 5 Interrupt Class 0 Compare interrupt is non-critical. 1 Compare interrupt is critical.	
17	CT6	GPT Compare Timer 6 Interrupt Class 0 Compare interrupt is non-critical. 1 Compare interrupt is critical.	
18	CT0	GPT Compare Timer 0 Interrupt Class 0 Compare interrupt has is non-critical. 1 Compare interrupt is critical.	
19	CT1	GPT Compare Timer 1 Interrupt Class 0 Compare interrupt is non-critical. 1 Compare interrupt is critical.	
20	CT2	GPT Compare Timer 2 Interrupt Class 0 Compare interrupt is non-critical. 1 Compare interrupt is critical.	
21	CT3	GPT Compare Timer 3 Interrupt Class 0 Compare 6 interrupt is non-critical. 1 Compare interrupt is critical.	
22	CT4	GPT Compare Timer 4 Interrupt Class 0 Compare interrupt is non-critical. 1 Compare interrupt is critical.	
23	EIR0	External IRQ 0 Interrupt Class 0 External IRQ 0 interrupt is non-critical. 1 External IRQ 0 interrupt is critical.	
24	EIR1	External IRQ 1 Interrupt Class 0 External IRQ 1 interrupt is non-critical. 1 External IRQ 1 interrupt is critical.	
25	EIR2	External IRQ 2 Interrupt Class 0 External IRQ 2 interrupt is non-critical. 1 External IRQ 2 interrupt is critical.	
26	EIR3	External IRQ 3 Interrupt Class 0 External IRQ 3 interrupt is non-critical. 1 External IRQ 3 interrupt is critical.	
27	EIR4	External IRQ 4 Interrupt Class 0 External IRQ 4 interrupt is non-critical. 1 External IRQ 4 interrupt is critical.	

28	EIR5	External IRQ 5 Interrupt Class 0 External IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.	
29	EIR6	External IRQ 6 Interrupt Class 0 External IRQ 6 interrupt is non-critical. 1 External IRQ 6 interrupt is critical.	
30	UIC1NC	UIC1 Non-Critical Interrupt Class 0 UICNC interrupt is non-critical. 1 UICNC interrupt is critical.	
31	UIC1C	UIC1 Critical Interrupt Class 0 UICC interrupt is non-critical. 1 UICC interrupt is critical.	

10.5.6 UIC1 Critical Register (UIC1_CR)

The fields of the UIC1_CR, which correspond to the fields of the UIC1_SR and UIC1_ER, determine whether an interrupt captured in the corresponding fields of the UIC1_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC1_ER. The processor handles critical interrupts when MSR[EE] = 1 and non-critical interrupts when MSR[CE]=1.

If a UIC1_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC1_SR and enabled in the corresponding field of the UIC1_ER) generates a non-critical interrupt signal to the processor core. If a UIC1_CR field is a 1, a critical interrupt signal is generated.

Figure 10-7. UIC1 Critical Register (UIC1_CR)

0	MS	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.	
1	MTDE	MAL TXDE Interrupt Class 0 MAL TXDE interrupt is non-critical. 1 MAL TXDE interrupt is critical.	
2	MRDE	MAL RXDE Interrupt Class 0 MAL RXDE interrupt is non-critical. 1 MAL RXDE interrupt is critical.	
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Class 0 DDRSDRAM ECC uncorrectable error interrupt is non-critical. 1 DDRSDRAM ECC uncorrectable error interrupt is critical.	
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Class 0 DDRSDRAM ECC correctable error interrupt is non-critical. 1 DDRSDRAM ECC correctable error interrupt is critical.	
5	EBC	EBC Interrupt Class 0 EBC interrupt is non-critical. 1 EBC interrupt is critical.	
6	NDFC	NDFC Interrupt Class 0 NDFC interrupt is non-critical. 1 NDFC interrupt is critical.	
7	OPB	OPB to PLB Bridge Interrupt Class 0 OPB to PLB bridge interrupt is non-critical. 1 OPB to PLB bridge interrupt is critical.	

8	USB1H1	USB1.1 Host 1 Interrupt Class 0 USB1.1 Host 1 interrupt is non-critical. 1 USB1.1 Host 1 interrupt is critical.	
9	USB1H2	USB1.1 Host 2 Interrupt Class 0 USB1.1 Host 2 interrupt is non-critical. 1 USB1.1 Host 2 interrupt is critical.	
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Class 0 PLB3 to PLB4 bridge 0 interrupt is non-critical. 1 PLB3 to PLB4 bridge 0 interrupt is critical.	
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Class 0 PLB3 to PLB4 bridge 1 interrupt is non-critical. 1 PLB3 to PLB4 bridge 1 interrupt is critical.	
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Class 0 PLB3 to PLB4 bridge 2 interrupt is non-critical. 1 PLB3 to PLB4 bridge 2 interrupt is critical.	
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Class 0 PLB3 to PLB4 bridge 3 interrupt is non-critical. 1 PLB3 to PLB4 bridge 3 interrupt is critical.	
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Class 0 PLB3 to PLB4 bridge 4 interrupt is non-critical. 1 PLB3 to PLB4 bridge 4 interrupt is critical.	
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Class 0 PLB3 to PLB4 bridge 5 interrupt is non-critical. 1 PLB3 to PLB4 bridge 5 interrupt is critical.	
16	UDMA0	UDMA0 Interrupt Class 0 UDMA0 interrupt is non-critical. 1 UDMA0 interrupt is critical.	DMA2P40 USB 2.0 device Endpoint 1 In.
17	UDMA1	UDMA1 Interrupt Class 0 UDMA1 interrupt is non-critical. 1 UDMA1 interrupt is critical.	DMA2P40 USB 2.0 device Endpoint 2 In.
18	EIR7	External IRQ 7 Interrupt Class 0 External IRQ 7 interrupt is non-critical. 1 External IRQ 7 interrupt is critical.	
19	EIR8	External IRQ 8 Interrupt Class 0 External IRQ 8 interrupt is non-critical. 1 External IRQ 8 interrupt is critical.	
20	EIR9	External IRQ 9 Interrupt Class 0 External IRQ 9 interrupt is non-critical. 1 External IRQ 9 interrupt is critical.	
21	UDMA2	UDMA2 Interrupt Class 0 UDMA2 interrupt is non-critical. 1 UDMA2 interrupt is critical.	DMA2P40 USB 2.0 device Endpoint 1 Out.
22	UDMA3	UDMA3 Interrupt Class 0 UDMA3 interrupt is non-critical. 1 UDMA3 interrupt is critical.	DMA2P40 USB 2.0 device Endpoint 2 Out.
23	USBD	USB1.1 and USB2.0 Device Interrupt Class 0 USB1.1 and USB2.0 device interrupt is non-critical. 1 USB1.1 and USB2.0 device interrupt is critical.	
24	SRE	Serial ROM Error Interrupt Class 0 Serial ROM error interrupt is non-critical. 1 Serial ROM error interrupt is critical.	
25	GDP	GPT Decrement Pulse Interrupt Class 0 GPT decrement pulse interrupt is non-critical. 1 GPT decrement pulse interrupt is critical.	

26	PPM	PLB Performance Monitor Interrupt Class 0 PLB performance monitor interrupt is non-critical. 1 PLB performance monitor interrupt is critical.	
27	EPP	EXT_PCI_PERR (parity) Interrupt Class 0 EXT_PCI_PERR (parity) interrupt is non-critical. 1 EXT_PCI_PERR (parity) interrupt is critical.	
28	ETH0	Ethernet 0 Interrupt Class 0 Ethernet 0 interrupt is non-critical. 1 Ethernet 0 interrupt is critical.	
29	EWU0	Ethernet 0 Wake-up Interrupt Class 0 Ethernet 0 wake-up interrupt is non-critical. 1 Ethernet 0 wake-up interrupt is critical.	
30	ETH1	Ethernet 1 Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.	
31	EWU1	Ethernet 1 Wake-up Interrupt Class 0 Ethernet 1 wake-up interrupt is non-critical. 1 Ethernet 1 wake-up interrupt is critical.	

10.5.7 UIC0 Polarity Register (UIC0_PR)

The fields of the UIC0_PR, which correspond to the fields of the UIC0_SR, determine whether the corresponding fields in the UIC0_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC0_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC0_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC0_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC0_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC0_PR) have positive polarity, the associated fields must be set to 1.

Figure 10-8. UIC0 Polarity Register (UIC0_PR)

0	U0	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.	Must be set to 1
1	U1	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.	Must be set to 1
2	IIC0	IIC0 Interrupt Polarity 0 IIC0 interrupt has negative polarity. 1 IIC0 interrupt has positive polarity.	Must be set to 1
3	U2	UART2 Interrupt Polarity 0 UART2 interrupt has negative polarity. 1 UART2 interrupt has positive polarity.	Must be set to 1
4	U3	UART3 Interrupt Polarity 0 UART3 interrupt has negative polarity. 1 UART3 interrupt has positive polarity.	Must be set to 1

5	PCRW	PCI Command Register Write Interrupt Polarity 0 PCI command register write interrupt has negative polarity. 1 PCI command register write interrupt has positive polarity.	Must be set to 1
6	PPM	PCI Power Management Interrupt Polarity 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.	Must be set to 1
7	IIC1	IIC1 Interrupt Polarity 0 IIC1 interrupt has negative polarity. 1 IIC1 interrupt has positive polarity.	Must be set to 1
8	SPI	SPI Interrupt Polarity 0 SPI interrupt has negative polarity. 1 SPI interrupt has positive polarity.	Must be set to 1
9	EPS	Ext PCI SERR Interrupt Polarity 0 Ext PCI SERR interrupt has negative polarity. 1 Ext PCI SERR interrupt has positive polarity.	Must be set to 0
10	MTE	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has positive polarity.	Must be set to 1
11	MRE	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has positive polarity.	Must be set to 1
12	D0	DMA2P30 Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.	Must be set to 1
13	D1	DMA2P30 Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.	Must be set to 1
14	D2	DMA2P30 Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.	Must be set to 1
15	D3	DMA2P30 Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.	Must be set to 1
16	CT5	GPT Compare Timer 5 Interrupt Polarity 0 Compare timer 5 interrupt has negative polarity. 1 Compare timer 5 interrupt has positive polarity.	Must be set to 1
17	CT6	GPT Compare Timer 6 Interrupt Polarity 0 Compare timer 6 interrupt has negative polarity. 1 Compare timer 6 interrupt has positive polarity.	Must be set to 1
18	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Compare timer 0 interrupt has negative polarity. 1 Compare timer 0 interrupt has positive polarity.	Must be set to 1
19	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Compare timer 1 interrupt has negative polarity. 1 Compare timer 1 interrupt has positive polarity.	Must be set to 1
20	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Compare timer 2 interrupt has negative polarity. 1 Compare timer 2 interrupt has positive polarity.	Must be set to 1
21	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Compare timer 3 interrupt has negative polarity. 1 Compare timer 3 interrupt has positive polarity.	Must be set to 1

22	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Compare timer 4 interrupt has negative polarity. 1 Compare timer 4 interrupt has positive polarity.	Must be set to 1
23	EIR0	External IRQ 0 Interrupt Polarity 0 External IRQ 0 interrupt has negative polarity. 1 External IRQ 0 interrupt has positive polarity.	
24	EIR1	External IRQ 1 Interrupt Polarity 0 External IRQ 1 interrupt has negative polarity. 1 External IRQ 1 interrupt has positive polarity.	
25	EIR2	External IRQ 2 Interrupt Polarity 0 External IRQ 2 interrupt has negative polarity. 1 External IRQ 2 interrupt has positive polarity.	
26	EIR3	External IRQ 3 Interrupt Polarity 0 External IRQ 3 interrupt has negative polarity. 1 External IRQ 3 interrupt has positive polarity.	
27	EIR4	External IRQ 4 Interrupt Polarity 0 External IRQ 4 interrupt has negative polarity. 1 External IRQ 4 interrupt has positive polarity.	
28	EIR5	External IRQ 5 Interrupt Polarity 0 External IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.	
29	EIR6	External IRQ 6 Interrupt Polarity 0 External IRQ 6 interrupt has negative polarity. 1 External IRQ 6 interrupt has positive polarity.	
30	UIC1NC	UIC1 Non-Critical Interrupt Polarity 0 UICNC interrupt has negative polarity. 1 UICNC interrupt has positive polarity.	Must be set to 1
31	UIC1C	UIC1 Critical Interrupt Polarity 0 UICC interrupt has negative polarity. 1 UICC interrupt has positive polarity.	Must be set to 1

10.5.8 UIC1 Polarity Register (UIC1_PR)

The fields of the UIC1_PR, which correspond to the fields of the UIC1_SR, determine whether the corresponding fields in the UIC1_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC1_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC1_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC1_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC1_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC1_PR) have positive polarity, the associated fields must be set to 1.

Figure 10-9. UIC1 Polarity Register (UIC1_PR)

0	MS	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.	Must be set to 1
1	MTDE	MAL TXDE Interrupt Polarity 0 MAL TXDE interrupt has negative polarity. 1 MAL TXDE interrupt has positive polarity.	Must be set to 1

2	MRDE	MAL RXDE Interrupt Polarity 0 MAL RXDE interrupt has negative polarity. 1 MAL RXDE interrupt has positive polarity.	Must be set to 1
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Polarity 0 DDRSDRAM ECC uncorrectable error interrupt has negative polarity. 1 DDRSDRAM ECC uncorrectable error interrupt has positive polarity.	Must be set to 1
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Polarity 0 DDRSDRAM ECC correctable error interrupt has negative polarity. 1 DDRSDRAM ECC correctable error interrupt has positive polarity.	Must be set to 1
5	EBC	EBC Interrupt Polarity 0 EBC interrupt has negative polarity. 1 EBC interrupt has positive polarity.	Must be set to 1
6	NDFC	NDFC Interrupt Polarity 0 NDFC interrupt has negative polarity. 1 NDFC interrupt has positive polarity.	Must be set to 1
7	OPB	OPB to PLB Bridge Interrupt Polarity 0 OPB to PLB bridge interrupt has negative polarity. 1 OPB to PLB bridge interrupt has positive polarity.	Must be set to 1
8	USB1H1	USB1.1 Host 1 Interrupt Polarity 0 USB1.1 Host 1 interrupt has negative polarity. 1 USB1.1 Host 1 interrupt has positive polarity.	Must be set to 1
9	USB1H2	USB1.1 Host 2 Interrupt Polarity 0 USB1.1 Host 2 interrupt has negative polarity. 1 USB1.1 Host 2 interrupt has positive polarity.	Must be set to 1
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Polarity 0 PLB3 to PLB4 bridge 0 interrupt has negative polarity. 1 PLB3 to PLB4 bridge 0 interrupt has positive polarity.	Must be set to 1
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Polarity 0 PLB3 to PLB4 bridge 1 interrupt has negative polarity. 1 PLB3 to PLB4 bridge 1 interrupt has positive polarity.	Must be set to 1
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Polarity 0 PLB3 to PLB4 bridge 2 interrupt has negative polarity. 1 PLB3 to PLB4 bridge 2 interrupt has positive polarity.	Must be set to 1
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Polarity 0 PLB3 to PLB4 bridge 3 interrupt has negative polarity. 1 PLB3 to PLB4 bridge 3 interrupt has positive polarity.	Must be set to 1
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Polarity 0 PLB3 to PLB4 bridge 4 interrupt has negative polarity. 1 PLB3 to PLB4 bridge 4 interrupt has positive polarity.	Must be set to 1

15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Polarity 0 PLB3 to PLB4 bridge 5 interrupt has negative polarity. 1 PLB3 to PLB4 bridge 5 interrupt has positive polarity.	Must be set to 1
16	UDMA0	UDMA0 Interrupt Polarity 0 UDMA0 interrupt has negative polarity. 1 UDMA0 interrupt has positive polarity.	Must be set to 1
17	UDMA1	UDMA1 Interrupt Polarity 0 UDMA1 interrupt has negative polarity. 1 UDMA1 interrupt has positive polarity.	Must be set to 1
18	EIR7	External IRQ 7 Interrupt Polarity 0 External IRQ 7 interrupt has negative polarity. 1 External IRQ 7 interrupt has positive polarity.	
19	EIR8	External IRQ 8 Interrupt Polarity 0 External IRQ 8 interrupt has negative polarity. 1 External IRQ 8 interrupt has positive polarity.	
20	EIR9	External IRQ 9 Interrupt Polarity 0 External IRQ 9 interrupt has negative polarity. 1 External IRQ 9 interrupt has positive polarity.	
21	UDMA2	UDMA2 Interrupt Polarity 0 UDMA2 interrupt has negative polarity. 1 UDMA2 interrupt has positive polarity.	Must be set to 1
22	UDMA3	UDMA3 Interrupt Polarity 0 UDMA3 interrupt has negative polarity. 1 UDMA3 interrupt has positive polarity.	Must be set to 1
23	USB2D	USB1.1 and USB2.0 Device Interrupt Polarity 0 USB1.1 and USB2.0 device interrupt has negative polarity. 1 USB1.1 and USB2.0 device interrupt has positive polarity.	Must be set to 0
24	SRE	Serial ROM Error Interrupt Polarity 0 Serial ROM error interrupt has negative polarity. 1 Serial ROM error interrupt has positive polarity.	Must be set to 1
25	GDP	GPT Decrement Pulse Interrupt Polarity 0 GPT decrement pulse interrupt has negative polarity. 1 GPT decrement pulse interrupt has positive polarity.	Must be set to 0
26	PPM	PLB Performance Monitor Interrupt Polarity 0 PLB performance monitor interrupt has negative polarity. 1 PLB performance monitor interrupt has positive polarity.	Must be set to 1
27	EPP	EXT_PCI_PERR (parity) Interrupt Polarity 0 EXT_PCI_PERR (parity) interrupt has negative polarity. 1 EXT_PCI_PERR (parity) interrupt has positive polarity.	Must be set to 0
28	ETH0	Ethernet 0 Interrupt Polarity 0 Ethernet 0 interrupt has negative polarity. 1 Ethernet 0 interrupt has positive polarity.	Must be set to 1
29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Ethernet 0 wake-up interrupt has negative polarity. 1 Ethernet 0 wake-up interrupt has positive polarity.	Must be set to 1
30	ETH1	Ethernet 1 Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.	Must be set to 1

31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Ethernet 1 wake-up interrupt has negative polarity. 1 Ethernet 1 wake-up interrupt has positive polarity.	Must be set to 1

10.5.9 UIC0 Trigger Register (UIC0_TR)

The fields of the UIC0_TR, which correspond to the fields of the UIC0_SR, determine whether corresponding fields in the UIC0_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC0_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC0_TR field is 0, the associated interrupt is level-sensitive. If the UIC0_TR field is 1, the interrupt is edge-sensitive.

Figure 10-10. UIC0 Trigger Register (UIC0_TR)

0	U0	UART0 Interrupt Trigger 0 UART0 interrupt is level sensitive. 1 UART0 interrupt is edge sensitive.	Must be set to 0
1	U1	UART1 Interrupt Trigger 0 UART1 interrupt is level sensitive. 1 UART1 interrupt is edge sensitive.	Must be set to 0
2	IIC0	IIC0 Interrupt Trigger 0 IIC0 interrupt is level sensitive. 1 IIC0 interrupt is edge sensitive.	Must be set to 0
3	U2	UART2 Interrupt Trigger 0 UART2 interrupt is level sensitive. 1 UART2 interrupt is edge sensitive.	Must be set to 0
4	U3	UART3 Interrupt Trigger 0 UART3 interrupt is level sensitive. 1 UART3 interrupt is edge sensitive.	Must be set to 0
5	PCRW	PCI Command Register Write Interrupt Trigger 0 PCI command register write interrupt is level sensitive. 1 PCI command register write interrupt is edge sensitive.	Must be set to 0
6	PPM	PCI Power Management Interrupt Trigger 0 PCI power management interrupt is level sensitive. 1 PCI power management interrupt is edge sensitive.	Must be set to 0
7	IIC1	IIC1 Interrupt Trigger 0 IIC1 interrupt is level sensitive. 1 IIC1 interrupt is edge sensitive.	Must be set to 0
8	SPI	SPI Interrupt Trigger 0 SPI interrupt is level sensitive. 1 SPI interrupt is edge sensitive.	Must be set to 0
9	EPS	Ext PCI SERR Interrupt Trigger 0 Ext PCI SERR interrupt is level sensitive. 1 Ext PCI SERR interrupt is edge sensitive.	Must be set to 0

10	MTE	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level sensitive. 1 MAL TX EOB interrupt is edge sensitive.	Must be set to 0
11	MRE	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level sensitive. 1 MAL RX EOB interrupt is edge sensitive.	Must be set to 0
12	D0	DMA2P30 Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level sensitive. 1 DMA channel 0 interrupt is edge sensitive.	Must be set to 0
13	D1	DMA2P30 Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level sensitive. 1 DMA channel 1 interrupt is edge sensitive.	Must be set to 0
14	D2	DMA2P30 Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level sensitive. 1 DMA channel 2 interrupt is edge sensitive.	Must be set to 0
15	D3	DMA2P30 Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level sensitive. 1 DMA channel 3 interrupt is edge sensitive.	Must be set to 0
16	CT5	GPT Compare Timer 5 Interrupt Trigger 0 Compare timer 5 interrupt is level sensitive. 1 Compare timer 5 interrupt is edge sensitive.	Must be set to 0
17	CT6	GPT Compare Timer 6 Interrupt Trigger 0 Compare timer 6 interrupt is level sensitive. 1 Compare timer 6 interrupt is edge sensitive.	Must be set to 0
18	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Compare timer 0 interrupt has is level sensitive. 1 Compare timer 0 interrupt is edge sensitive.	Must be set to 0
19	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Compare timer 1 interrupt is level sensitive. 1 Compare timer 1 interrupt is edge sensitive.	Must be set to 0
20	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Compare timer 2 interrupt is level sensitive. 1 Compare timer 2 interrupt is edge sensitive.	Must be set to 0
21	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Compare timer 3 interrupt is level sensitive. 1 Compare timer 3 interrupt is edge sensitive.	Must be set to 0
22	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Compare timer 4 interrupt is level sensitive. 1 Compare timer 4 interrupt is edge sensitive.	Must be set to 0
23	EIR0	External IRQ 0 Interrupt Trigger 0 External IRQ 0 interrupt is level sensitive. 1 External IRQ 0 interrupt is edge sensitive.	
24	EIR1	External IRQ 1 Interrupt Trigger 0 External IRQ 1 interrupt is level sensitive. 1 External IRQ 1 interrupt is edge sensitive.	
25	EIR2	External IRQ 2 Interrupt Trigger 0 External IRQ 2 interrupt is level sensitive. 1 External IRQ 2 interrupt is edge sensitive.	
26	EIR3	External IRQ 3 Interrupt Trigger 0 External IRQ 3 interrupt is level sensitive. 1 External IRQ 3 interrupt edge sensitive.	
27	EIR4	External IRQ 4 Interrupt Trigger 0 External IRQ 4 interrupt is level sensitive. 1 External IRQ 4 interrupt is edge sensitive.	

28	EIR5	External IRQ 5 Interrupt Trigger 0 External IRQ 5 interrupt is level sensitive. 1 An external IRQ 5 interrupt is edge sensitive.	
29	EIR6	External IRQ 6 Interrupt Trigger 0 External IRQ 6 interrupt is level sensitive. 1 External IRQ 6 interrupt is edge sensitive.	
30	UIC1NC	UIC1 Non-Critical Interrupt Trigger 0 UICNC interrupt is level sensitive. 1 UICNC interrupt is edge sensitive.	Must be set to 0
31	UIC1C	UIC1 Critical Interrupt Trigger 0 UICC interrupt is level sensitive. 1 UICC interrupt is edge sensitive.	Must be set to 0

10.5.10 UIC1 Trigger Register (UIC1_TR)

The fields of the UIC1_TR, which correspond to the fields of the UIC1_SR, determine whether corresponding fields in the UIC1_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC1_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC1_TR field is 0, the associated interrupt is level-sensitive. If the UIC1_TR field is 1, the interrupt is edge-sensitive.

Figure 10-11. UIC1 Trigger Register (UIC1_TR)

0	MS	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level sensitive. 1 MAL SERR interrupt is edge sensitive.	Must be set to 0
1	MTDE	MAL TXDE Interrupt Trigger 0 MAL TXDE interrupt is level sensitive. 1 MAL TXDE interrupt is edge sensitive.	Must be set to 0
2	MRDE	MAL RXDE Interrupt Trigger 0 MAL RXDE interrupt is level sensitive. 1 MAL RXDE interrupt is edge sensitive.	Must be set to 0
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Trigger 0 DDRSDRAM ECC uncorrectable error interrupt is level sensitive. 1 DDRSDRAM ECC uncorrectable error interrupt is edge sensitive.	Must be set to 0
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Trigger 0 DDRSDRAM ECC correctable error interrupt is level sensitive. 1 DDRSDRAM ECC correctable error interrupt is edge sensitive.	Must be set to 0
5	EBC	EBC Interrupt Trigger 0 EBC interrupt is level sensitive. 1 EBC interrupt is edge sensitive.	Must be set to 1
6	NDFC	NDFC Interrupt Trigger 0 NDFC interrupt is level sensitive. 1 NDFC interrupt is edge sensitive.	Must be set to 1

7	OPB	OPB to PLB Bridge Interrupt Trigger 0 OPB to PLB bridge interrupt is level sensitive. 1 OPB to PLB bridge interrupt is edge sensitive.	Must be set to 0
8	USB1H1	USB1.1 Host 1 Interrupt Trigger 0 USB1.1 Host 1 interrupt is level sensitive. 1 USB1.1 Host 1 interrupt is edge sensitive.	Must be set to 0
9	USB1H2	USB1.1 Host 2 Interrupt Trigger 0 USB1.1 Host 2 interrupt is level sensitive. 1 USB1.1 Host 2 interrupt is edge sensitive.	Must be set to 0
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Trigger 0 PLB3 to PLB4 bridge 0 interrupt is level sensitive. 1 PLB3 to PLB4 bridge 0 interrupt is edge sensitive.	Must be set to 0
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Trigger 0 PLB3 to PLB4 bridge 1 interrupt is level sensitive. 1 PLB3 to PLB4 bridge 1 interrupt is edge sensitive.	Must be set to 0
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Trigger 0 PLB3 to PLB4 bridge 2 interrupt is level sensitive. 1 PLB3 to PLB4 bridge 2 interrupt is edge sensitive.	Must be set to 0
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Trigger 0 PLB3 to PLB4 bridge 3 interrupt is level sensitive. 1 PLB3 to PLB4 bridge 3 interrupt is edge sensitive.	Must be set to 0
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Trigger 0 PLB3 to PLB4 bridge 4 interrupt is level sensitive. 1 PLB3 to PLB4 bridge 4 interrupt is edge sensitive.	Must be set to 0
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Trigger 0 PLB3 to PLB4 bridge 5 interrupt is level sensitive. 1 PLB3 to PLB4 bridge 5 interrupt is edge sensitive.	Must be set to 0
16	UDMA0	UDMA0 Interrupt Trigger 0 UDMA0 interrupt is level sensitive. 1 UDMA0 interrupt is edge sensitive.	Must be set to 0
17	UDMA1	UDMA1 Interrupt Trigger 0 UDMA1 interrupt is level sensitive. 1 UDMA1 interrupt is edge sensitive.	Must be set to 0
18	EIR7	External IRQ 7 Interrupt Trigger 0 External IRQ 7 interrupt is level sensitive. 1 External IRQ 7 interrupt is edge sensitive.	
19	EIR8	External IRQ 8 Interrupt Trigger 0 External IRQ 8 interrupt is level sensitive. 1 External IRQ 8 interrupt is edge sensitive.	
20	EIR9	External IRQ 9 Interrupt Trigger 0 External IRQ 9 interrupt is level sensitive. 1 External IRQ 9 interrupt is edge sensitive.	
21	UDMA2	UDMA2 Interrupt Trigger 0 UDMA2 interrupt is level sensitive. 1 UDMA2 interrupt is edge sensitive.	Must be set to 0
22	UDMA3	UDMA3 Interrupt Trigger 0 UDMA3 interrupt is level sensitive. 1 UDMA3 interrupt is edge sensitive.	Must be set to 0
23	USBD	USB1.1 and USB2.0 Device Interrupt Trigger 0 USB1.1 and USB2.0 device interrupt is level sensitive. 1 USB1.1 and USB2.0 device interrupt is edge sensitive.	Must be set to 1
24	SRE	Serial ROM Error Interrupt Trigger 0 Serial ROM error interrupt is level sensitive. 1 Serial ROM error interrupt is edge sensitive.	Must be set to 0

25	GDP	GPT Decrement Pulse Interrupt Trigger 0 GPT decrement pulse interrupt is level sensitive. 1 GPT decrement pulse interrupt is edge sensitive.	Must be set to 1
26	PPM	PLB Performance Monitor Interrupt Trigger 0 PLB performance monitor interrupt is level sensitive. 1 PLB performance monitor interrupt is edge sensitive.	Must be set to 0
27	EPP	EXT_PCI_PERR (parity) Interrupt Trigger 0 EXT_PCI_PERR (parity) interrupt is level sensitive. 1 EXT_PCI_PERR (parity) interrupt is edge sensitive.	Must be set to 0
28	ETH0	Ethernet 0 Interrupt Trigger 0 Ethernet 0 interrupt is level sensitive. 1 Ethernet 0 interrupt is edge sensitive.	Must be set to 0
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Ethernet 0 wake-up interrupt is level sensitive. 1 Ethernet 0 wake-up interrupt is edge sensitive.	Must be set to 0
30	ETH1	Ethernet 1 Interrupt Trigger 0 Ethernet 1 interrupt is level sensitive. 1 Ethernet 1 interrupt is edge sensitive.	Must be set to 0
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Ethernet 1 wake-up interrupt is level sensitive. 1 Ethernet 1 wake-up interrupt is edge sensitive.	Must be set to 0

10.5.11 UIC0 Masked Status Register (UIC0_MSR)

This read-only register contains the result of masking the UIC0_SR with the UIC0_ER. Reading this register, instead of the actual UIC0_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC0_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC0_SR, the UIC0_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC0_SR can be successfully cleared.

Figure 10-12. UIC0 Masked Status Register (UIC0_MSR)

0	U0	UART0 Masked Interrupt Status 0 UART0 masked interrupt has not occurred. 1 UART0 masked interrupt occurred.	
1	U1	UART1 Masked Interrupt Status 0 UART1 masked interrupt has not occurred. 1 UART1 masked interrupt occurred.	
2	IIC0	IIC0 Masked Interrupt Status 0 IIC0 masked interrupt has not occurred. 1 IIC0 masked interrupt occurred.	
3	U2	UART2 Masked Interrupt Status 0 UART2 masked interrupt has not occurred. 1 UART2 masked interrupt occurred.	
4	U3	UART3 Masked Interrupt Status 0 UART3 masked interrupt has not occurred. 1 UART3 masked interrupt occurred.	

5	PCRW	PCI Command Register Write Masked Interrupt Status 0 PCI command register write masked interrupt has not occurred. 1 PCI command register write masked interrupt occurred.	
6	PPM	PCI Power Management Masked Interrupt Status 0 PCI power management masked interrupt has not occurred. 1 PCI power management masked interrupt occurred.	
7	IIC1	IIC1 Masked Interrupt Status 0 IIC1 masked interrupt has not occurred. 1 IIC1 masked interrupt occurred.	
8	SPI	SPI Masked Interrupt Status 0 SPI masked interrupt has not occurred. 1 SPI masked interrupt occurred.	
9	EPS	Ext PCI SERR Masked Interrupt Status 0 Ext PCI SERR masked interrupt has not occurred. 1 Ext PCI SERR masked interrupt occurred.	
10	MTE	MAL TX EOB Masked Interrupt Status 0 MAL TX EOB masked interrupt has not occurred. 1 MAL TX EOB masked interrupt has occurred.	
11	MRE	MAL RX EOB Masked Interrupt Status 0 MAL RX EOB masked interrupt has not occurred. 1 MAL RX EOB masked interrupt has occurred.	
12	D0	DMA2P30 Channel 0 Masked Interrupt Status 0 DMA channel 0 masked interrupt has not occurred. 1 DMA channel 0 masked interrupt occurred.	
13	D1	DMA2P30 Channel 1 Masked Interrupt Status 0 DMA channel 1 masked interrupt has not occurred. 1 DMA channel 1 masked interrupt occurred.	
14	D2	DMA2P30 Channel 2 Masked Interrupt Status 0 DMA channel 2 masked interrupt has not occurred. 1 DMA channel 2 masked interrupt occurred.	
15	D3	DMA2P30 Channel 3 Masked Interrupt Status 0 DMA channel 3 masked interrupt has not occurred. 1 DMA channel 3 masked interrupt occurred.	
16	CT5	GPT Compare Timer 5 Masked Interrupt Status 0 Compare timer 5 masked interrupt has not occurred. 1 Compare timer 5 masked interrupt occurred.	
17	CT6	GPT Compare Timer 6 Masked Interrupt Status 0 Compare timer 6 masked interrupt has not occurred. 1 Compare timer 6 masked interrupt occurred.	
18	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Compare timer 0 masked interrupt has not occurred. 1 Compare timer 0 masked interrupt occurred.	

19	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Compare timer 1 masked interrupt has not occurred. 1 Compare timer 1 masked interrupt occurred.	
20	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Compare timer 2 masked interrupt has not occurred. 1 Compare timer 2 masked interrupt occurred.	
21	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Compare timer 3 masked interrupt has not occurred. 1 Compare timer 3 masked interrupt occurred.	
22	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Compare timer 4 masked interrupt has not occurred. 1 Compare timer 4 masked interrupt occurred.	
23	EIR0	External IRQ 0 Masked Interrupt Status 0 External IRQ 0 masked interrupt has not occurred. 1 External IRQ 0 masked interrupt occurred.	
24	EIR1	External IRQ 1 Masked Interrupt Status 0 External IRQ 1 masked interrupt has not occurred. 1 External IRQ 1 masked interrupt occurred.	
25	EIR2	External IRQ 2 Masked Interrupt Status 0 External IRQ 2 masked interrupt has not occurred. 1 External IRQ 2 masked interrupt occurred.	
26	EIR3	External IRQ 3 Masked Interrupt Status 0 External IRQ 3 masked interrupt has not occurred. 1 External IRQ 3 masked interrupt occurred.	
27	EIR4	External IRQ 4 Masked Interrupt Status 0 External IRQ 4 masked interrupt has not occurred. 1 External IRQ 4 masked interrupt occurred.	
28	EIR5	External IRQ 5 Masked Interrupt Status 0 External IRQ 5 masked interrupt has not occurred. 1 An external IRQ 5 masked interrupt occurred.	
29	EIR6	External IRQ 6 Masked Interrupt Status 0 External IRQ 6 masked interrupt has not occurred. 1 External IRQ 6 masked interrupt occurred.	
30	UIC1NC	UIC1 Non-Critical Masked Interrupt Status 0 UICNC masked interrupt has not occurred. 1 UICNC masked interrupt occurred.	
31	UIC1C	UIC1 Critical Masked Interrupt Status 0 UICC masked interrupt has not occurred. 1 UICC masked interrupt occurred.	

10.5.12 UIC1 Masked Status Register (UIC1_MSR)

This read-only register contains the result of masking the UIC1_SR with the UIC1_ER. Reading this register, instead of the actual UIC1_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC1_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC1_SR, the UIC1_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC1_SR can be successfully cleared.

Figure 10-13. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 MAL SERR masked interrupt has not occurred. 1 MAL SERR masked interrupt occurred.	
1	MTDE	MAL TXDE Masked Interrupt Status 0 MAL TXDE masked interrupt has not occurred. 1 MAL TXDE masked interrupt occurred.	
2	MRDE	MAL RXDE Masked Interrupt Status 0 MAL RXDE masked interrupt has not occurred. 1 MAL RXDE masked interrupt occurred.	
3	DEUE	DDRSDRAM ECC Uncorrectable Error Masked Interrupt Status 0 DDRSDRAM ECC uncorrectable error masked interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error masked interrupt occurred.	
4	DECE	DDRSDRAM ECC Correctable Error Masked Interrupt Status 0 DDRSDRAM ECC correctable error masked interrupt has not occurred. 1 DDRSDRAM ECC correctable error masked interrupt occurred.	
5	EBC	EBC Masked Interrupt Status 0 EBC masked interrupt has not occurred. 1 EBC masked interrupt occurred.	
6	NDFC	NDFC Masked Interrupt Status 0 NDFC masked interrupt has not occurred. 1 NDFC masked interrupt occurred.	
7	OPB	OPB to PLB Bridge Masked Interrupt Status 0 OPB to PLB bridge masked interrupt has not occurred. 1 OPB to PLB bridge masked interrupt occurred.	
8	USB1H1	USB1.1 Host 1 Masked Interrupt Status 0 USB1.1 Host 1 masked interrupt has not occurred. 1 USB1.1 Host 1 masked interrupt occurred.	
9	USB1H2	USB1.1 Host 2 Masked Interrupt Status 0 USB1.1 Host 2 masked interrupt has not occurred. 1 USB1.1 Host 2masked interrupt occurred.	
10	P2P0	PLB3 to PLB4 Bridge 0 Masked Interrupt Status 0 PLB3 to PLB4 bridge 0 masked interrupt has not occurred. 1 PLB3 to PLB4 bridge 0 masked interrupt occurred.	
11	P2P1	PLB3 to PLB4 Bridge 1 Masked Interrupt Status 0 PLB3 to PLB4 bridge 1 masked interrupt has not occurred. 1 PLB3 to PLB4 bridge 1 masked interrupt occurred.	
12	P2P2	PLB3 to PLB4 Bridge 2 Masked Interrupt Status 0 PLB3 to PLB4 bridge 2 masked interrupt has not occurred. 1 PLB3 to PLB4 bridge 2 masked interrupt occurred.	

13	P2P3	PLB3 to PLB4 Bridge 3 Masked Interrupt Status 0 PLB3 to PLB4 bridge 3 masked interrupt has not occurred. 1 PLB3 to PLB4 bridge 3 masked interrupt occurred.	
14	P2P4	PLB3 to PLB4 Bridge 4 Masked Interrupt Status 0 PLB3 to PLB4 bridge 4 masked interrupt has not occurred. 1 PLB3 to PLB4 bridge 4 masked interrupt occurred.	
15	P2P5	PLB3 to PLB4 Bridge 5 Masked Interrupt Status 0 PLB3 to PLB4 bridge 5 masked interrupt has not occurred. 1 PLB3 to PLB4 bridge 5 masked interrupt occurred.	
16	UDMA0	UDMA0 Masked Interrupt Status 0 UDMA0 masked interrupt has not occurred. 1 UDMA0 masked interrupt occurred.	DMA2P40 USB 2.0 device Endpoint 1 In.
17	UDMA1	UDMA1 Masked Interrupt Status 0 UDMA1 masked interrupt has not occurred. 1 UDMA1 masked interrupt occurred.	DMA2P40 USB 2.0 device Endpoint 2 In.
18	EIR7	External IRQ 7 Masked Interrupt Status 0 External IRQ 7 masked interrupt has not occurred. 1 External IRQ 7 masked interrupt occurred.	
19	EIR8	External IRQ 8 Masked Interrupt Status 0 External IRQ 8 masked interrupt has not occurred. 1 External IRQ 8 masked interrupt occurred.	
20	EIR9	External IRQ 9 Masked Interrupt Status 0 External IRQ 9 masked interrupt has not occurred. 1 External IRQ 9 masked interrupt occurred.	
21	UDMA2	UDMA2 Masked Interrupt Status 0 UDMA2 masked interrupt has not occurred. 1 UDMA2 masked interrupt occurred.	DMA2P40 USB 2.0 device Endpoint 1 Out.
22	UDMA3	UDMA3 Masked Interrupt Status 0 UDMA3 masked interrupt has not occurred. 1 UDMA3 masked interrupt occurred.	DMA2P40 USB 2.0 device Endpoint 2 Out.
23	USBD	USB1.1 and USB2.0 Device Masked Interrupt Status 0 USB1.1 and USB2.0 device masked interrupt has not occurred. 1 USB1.1 and USB2.0 device masked interrupt occurred.	
24	SRE	Serial ROM Error Masked Interrupt Status 0 Serial ROM error masked interrupt has not occurred. 1 Serial ROM error masked interrupt occurred.	
25	GDP	GPT Decrement Pulse Masked Interrupt Status 0 GPT decrement pulse masked interrupt has not occurred. 1 GPT decrement pulse masked interrupt occurred.	
26	PPM	PLB Performance Monitor Masked Interrupt Status 0 PLB performance monitor masked interrupt has not occurred. 1 PLB performance monitor masked interrupt occurred.	
27	EPP	EXT_PCI_PERR (parity) Masked Interrupt Status 0 EXT_PCI_PERR (parity) masked interrupt has not occurred. 1 EXT_PCI_PERR (parity) masked interrupt occurred.	

28	ETH0	Ethernet 0 Masked Interrupt Status 0 Ethernet 0 masked interrupt has not occurred. 1 Ethernet 0 masked interrupt occurred.	
29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Ethernet 0 wake-up masked interrupt has not occurred. 1 Ethernet 0 wake-up masked interrupt occurred.	
30	ETH1	Ethernet 1 Masked Interrupt Status 0 Ethernet 1 masked interrupt has not occurred. 1 Ethernet 1 masked interrupt occurred.	
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Ethernet 1 wake-up masked interrupt has not occurred. 1 Ethernet 1 wake-up masked interrupt occurred.	

10.5.13 UIC0 Vector Configuration Register (UIC0_VCR)

The write-only UIC0_VCR enables software control of interrupt vector generation for critical interrupts. UIC0_VCR contains an address, used as an interrupt vector base address, and specifies interrupt ordering priority. Vector generation is not performed for non-critical interrupts.

UIC0_VCR[VBA] can contain either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC0_VR, using UIC0_VCR[VBA]. Vector generation is described in *UIC0 Vector Register (UIC0_VR)* on page 254. Because the two lowest-order bits of an interrupt handler address are assumed to be 00 to ensure word alignment, 30 bits are sufficient to form the base address.

A general interrupt handler uses the vector to access a table of interrupt vectors. Each interrupt vector table entry contains the address of an interrupt handler for a specific interrupt. Alternatively, UIC0_VCR[VBA] can directly address the interrupt handlers for specific interrupts, which in memory are separated by an offset calculated in UIC0_VR.

UIC0_VCR[PRO] controls whether the interrupt associated with UIC0_SR[0] or UIC0_SR[31] has the highest priority. If UIC0_VCR[PRO] = 0, the interrupt associated with UIC0_SR[31] has the highest priority; if UIC0_VCR[PRO] = 1, the interrupt associated with UIC0_SR[0] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC0_CR as a interrupt has the second highest priority. Priority decreases across the UIC0_SR to the end opposite the highest priority field.

Figure 10-14. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address	
30		Reserved	
31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt.	**Note:** Vector generation is not performed for non-critical interrupts.

10.5.14 UIC1 Vector Configuration Register (UIC1_VCR)

The write-only UIC1_VCR enables software control of interrupt vector generation for critical interrupts. UIC1_VCR contains an address, used as an interrupt vector base address, and specifies interrupt ordering priority. Vector generation is not performed for non-critical interrupts.

UIC1_VCR[VBA] can contain either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC1_VR, using UIC1_VCR[VBA]. Vector generation is described in *UIC1 Vector Register (UIC1_VR)* on page 255. Because the two lowest-order bits of an interrupt handler address are assumed to be 00 to ensure word alignment, 30 bits are sufficient to form the base address.

A general interrupt handler uses the vector to access a table of interrupt vectors. Each interrupt vector table entry contains the address of an interrupt handler for a specific interrupt. Alternatively, UIC1_VCR[VBA] can directly address the interrupt handlers for specific interrupts, which in memory are separated by an offset calculated in UIC1_VR.

UIC1_VCR[PRO] controls whether the interrupt associated with UIC1_SR[0] or UIC1_SR[31] has the highest priority. If UIC1_VCR[PRO] = 0, the interrupt associated with UIC1_SR[31] has the highest priority; if UIC1_VCR[PRO] = 1, the interrupt associated with UIC1_SR[0] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC*x*_CR as a interrupt has the second highest priority. Priority decreases across the UIC1_SR to the end opposite the highest priority field.

Figure 10-15. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address	
30		Reserved	
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt.	**Note:** Vector generation is not performed for non-critical interrupts.

10.5.15 UIC0 Vector Register (UIC0_VR)

The read-only UIC0_VR contains an interrupt vector that can reduce interrupt handling latency for critical interrupts. Vector generation logic adds an offset to UIC0_VCR[VBA], and the sum is returned in the UIC0_VR. Vectors are not computed for non-critical interrupts.

The interrupt vector is based on the field position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC0_SR. The generated vectors can be programmed to point directly to the interrupt handlers.

Programming Note: Regardless of the programming of UIC0_VCR and UIC0_VR registers, the processor always vectors to IVPR and IVOR0 when a critical interrupt occurs.

The interrupt vector offset is based on the bit position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC0_SR. The offset has a fixed value of 512 per bit. The main critical interrupt handler can interpret the vector returned by UIC0_VR as the address of the interrupt handler for that interrupt, assuming the routine is 512 bytes or smaller. Alternatively, the main critical interrupt handler can interpret the vector as a look-up table entry for the address of the interrupt handler for that interrupt.

Figure 10-16. UIC Vector Register (UIC0_VR)

0:31	VR	Interrupt Vector	

The following example illustrates the generation of a UIC0_VR vector for external interrupt request IRQ2.

For the example, assume that UIC0_VCR[PRO] = 0, so that UIC0_SR[EIR6S] (UIC0_SR$_{31}$)has the highest interrupt priority, and that UIC0_SR[EIR2S] (UIC0_SR$_{27}$) is the current highest priority, enabled, active, critical interrupt. To generate the vector for the interrupt associated with UIC0_SR[EIR2S], internal logic multiplies the difference between the highest priority interrupt bit and the active enabled priority interrupt bit by 512. The interrupt vector offset is therefore $(31 - 27) \times 512 = 4 \times 512$. This offset is added to the base address in UIC0_VCR[VBA], and the UIC0_VR returns UIC0_VCR[VBA] + (4×512).

10.5.15.1 Using the Value in UIC0_VR as a Vector Address or Entry Table Lookup

If an interrupt handler is 512 bytes or smaller, system software can interpret the value returned in the UIC0_VR as an address. In this case, when the interrupt is received, the UIC0_VR is read and software simply jumps to the address represented by the UIC0_VR value. Alternatively, the routine can be at a different address, and system software can treat the value of the UIC0_VR as a pointer, storing the interrupt handler address in the UIC0_VR during system initialization. In this case, when the interrupt is handled, software must read the UIC0_VR, read the entry at the UIC0_VR value, and jump to the entry. Hardware has no knowledge of the method is used, which is determined by system software.

10.5.15.2 Vector Generation Scenarios

For the following sequence, assume that the interrupts are enabled and critical (vectors are not generated for disabled or non-critical interrupts). The sequence illustrates several scenarios for vector generation.

1. An intermediate priority interrupt goes active; its vector is stored in UIC0_VR.
2. A low priority interrupt goes active; UIC0_VR is unchanged.
3. Software reads the vector; UIC0_VR is unchanged.
4. Software resets the intermediate priority interrupt; UIC0_VR contains the vector for the low priority interrupt.
5. A high priority interrupt goes active; UIC0_VR contains the vector for the high priority interrupt.
6. Software resets the high priority interrupt; UIC0_VR contains the vector for the low priority interrupt.
7. Software resets the UIC0_ER field for the low priority interrupt, disabling it; UIC0_VR contains 0x00000000.
8. UIC0_CR is reprogrammed to make the low priority interrupt non-critical and UIC0_ER is reprogrammed to re-enable the low priority interrupt; UIC0_VR continues to contain 0x00000000.

10.5.16 UIC1 Vector Register (UIC1_VR)

The read-only UIC1_VR contains an interrupt vector that can reduce interrupt handling latency for critical interrupts. Vector generation logic adds an offset to UIC1_VCR[VBA], and the sum is returned in the UIC1_VR. Vectors are not computed for non-critical interrupts.

The interrupt vector is based on the field position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC1_SR. The generated vectors can be programmed to point directly to the interrupt handlers.

> **Programming Note:** Regardless of the programming of UIC1_VCR and UIC1_VR registers, the processor always vectors to IVPR and IVOR0 when a critical interrupt occurs.

The interrupt vector offset is based on the bit position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC1_SR. The offset has a fixed value of 512 per bit. The main critical interrupt handler can interpret the vector returned by UIC1_VR as the address of the interrupt handler for that interrupt, assuming the routine is 512 bytes or smaller. Alternatively, the main critical interrupt handler can interpret the vector as a look-up table entry for the address of the interrupt handler for that interrupt.

Figure 10-17. UIC1 Vector Register (UIC1_VR)			
0:31		Interrupt Vector	

The following example illustrates the generation of a UIC1_VR vector for external interrupt request IRQ2.

For the example, assume that UIC1_VCR[PRO] = 0, so that UIC1_SR[EIR6S] (UIC1_SR_{31})has the highest interrupt priority, and that UIC1_SR[EIR2S] (UIC1_SR_{27}) is the current highest priority, enabled, active, critical interrupt. To generate the vector for the interrupt associated with UIC1_SR[EIR2S], internal logic multiplies the difference between the highest priority interrupt bit and the active enabled priority interrupt bit by 512. The interrupt vector offset is therefore $(31 - 27) \times 512 = 4 \times 512$. This offset is added to the base address in UIC1_VCR[VBA], and the UIC1_VR returns UIC1_VCR[VBA] + (4×512).

10.5.16.1 Using the Value in UIC1_VR as a Vector Address or Entry Table Lookup

If an interrupt handler is 512 bytes or smaller, system software can interpret the value returned in the UIC1_VR as an address. In this case, when the interrupt is received, the UIC1_VR is read and software simply jumps to the address represented by the UIC1_VR value. Alternatively, the routine can be at a different address, and system software can treat the value of the UIC1_VR as a pointer, storing the interrupt handler address in the UIC1_VR during system initialization. In this case, when the interrupt is handled, software must read the UIC1_VR, read the entry at the UIC1_VR value, and jump to the entry. Hardware has no knowledge of the method is used, which is determined by system software.

10.5.16.2 Vector Generation Scenarios

For the following sequence, assume that the interrupts are enabled and critical (vectors are not generated for disabled or non-critical interrupts). The sequence illustrates several scenarios for vector generation.

1. An intermediate priority interrupt goes active; its vector is stored in UIC1_VR.
2. A low priority interrupt goes active; UIC1_VR is unchanged.
3. Software reads the vector; UIC1_VR is unchanged.
4. Software resets the intermediate priority interrupt; UIC1_VR contains the vector for the low priority interrupt.
5. A high priority interrupt goes active; UIC1_VR contains the vector for the high priority interrupt.
6. Software resets the high priority interrupt; UIC1_VR contains the vector for the low priority interrupt.
7. Software resets the UIC1_ER field for the low priority interrupt, disabling it; UIC1_VR contains 0x00000000.
8. UIC1_CR is reprogrammed to make the low priority interrupt non-critical and UIC1_ER is reprogrammed to re-enable the low priority interrupt; UIC1_VR continues to contain 0x00000000.

11. Interrupts and Exceptions

Interrupt and exception processing for all chip functions except the FPU is handled by the PPC440 processor. Refer to the *PPC440 Processor User's Manual* for details.

12. Floating Point Unit Interrupts and Exceptions

An *interrupt* is the action in which the processor saves its old context (Machine State Register (MSR) and next instruction address NIA)) and begins execution at a pre-determined interrupt-handler address, with a modified MSR. *Exceptions* are the events that may cause the processor to take an interrupt, if the corresponding interrupt type is enabled.

Exceptions may be generated by the execution of instructions, or by signals from devices external to the PPC440 processor core, the internal timer facilities, debug events, or error conditions.

12.1 Floating-Point Exceptions

Book-E requires all synchronous (precise and imprecise) interrupts to be reported in program order, as required by the sequential execution model. The only exception to this rule is the case of multiple synchronous imprecise interrupts. Upon a synchronizing event, all previously executed instructions are required to report any synchronous imprecise interrupt-generating exceptions, and the interrupt is then generated with all of those exception types reported cumulatively, in both the Exception Syndrome Register (ESR), and any status registers associated with the particular exception type, such as the Floating-Point Status and Control Register (FPSCR).

For any single instruction attempting to cause multiple exceptions for which the corresponding synchronous interrupt types are enabled, this section defines the priority order by which the instruction will be permitted to cause a *single* enabled exception, thus generating a particular synchronous interrupt. This exception priority mechanism, along with the requirement that synchronous interrupts must be generated in program order, guarantees that only one of the synchronous interrupt types is considered at any given time. The exception priority mechanism also prevents certain debug exceptions from existing in combination with certain other synchronous interrupt-generating exceptions.

This section does not define the permitted setting of multiple exceptions for which the corresponding interrupt types are disabled. The generation of exceptions for which the corresponding interrupt types are disabled has no effect on the generation of other exceptions for which the corresponding interrupt types are enabled. Conversely, if a particular exception for which the corresponding interrupt type is enabled is shown in the following sections to be of a higher priority than another exception, it will prevent the setting of that other exception, regardless of whether the corresponding interrupt type of the other exception is enabled or disabled.

Except as noted, only one of the exception types listed for a given instruction type can be generated at any given time. The priority of the exception types are listed in subsequent sections ranging from highest to lowest, within each instruction type.

Note: Some exception types may be mutually exclusive of each other and could otherwise be considered the same priority. In such cases, the exceptions are listed in the order suggested by the sequential execution model.

Computational instructions may cause exceptions. Aside from instructions that write the FPSCR, none of the noncomputational instructions can cause a floating-point exception.

All exceptions are handled precisely. Because this can affect performance adversely, it is strongly recommended that exceptions should be disabled when possible. This prevents the PPC440 FPU instruction stream from waiting for the execution of long latency instructions, such as **fdiv[s]**.

12.2 Exceptions List

Book-E defines the following floating-point exceptions:

Table 12-1. Invalid Operation Exception Categories

Category	FPSCR Field
SNaN	VXSNAN
Infinity – Infinity	VXISI
Infinity ÷ Infinity	VXIDI
Zero ÷ Zero	VXZDZ
Infinity × Zero	VXIMZ
Invalid Compare	VXVC
Software Request	VXSOFT
Invalid Square Root	VXSQRT
Invalid Integer Convert	VXCVI

- Invalid Operation exception (VX)
- Zero Divide exception (ZX)
- Overflow exception (OX)
- Underflow exception (UX)
- Inexact exception (XI)

These exceptions can occur during execution of computational instructions. In addition, an Invalid Operation exception occurs when a **mtsfs** or **mtsfsi** instruction sets FPSCR[VXSOFT] = 1.

Each floating-point exception, and each category of Invalid Operation exception, has an exception bit in the FPSCR. Each floating-point exception also has a corresponding enable bit in the FPSCR. The exception bit indicates the occurrence of the corresponding exception. If an exception occurs, the corresponding enable bit controls the result produced by the instruction and, with MSR[FE0, FE1] whether and how the Enabled exception type Program interrupt is taken. (See *Floating-Point Exceptions* on page 259 for more information.) In general, the enabling specified by an enable bit is to enable the invoking the interrupt, not to enable the exception to occur. The occurrence of an exception depends only on the instruction and its inputs, not on the setting of any enable bits. The only exceptions to this general rule are the occurrence of an Underflow exception, which may depend on the setting of the enable bit, and the occurrence of a Inexact exception, which may depend on the Overflow exception bit not being set.

A single instruction, other than **mtfsf** or **mtfsfi**, can set more than one exception bit only in the following cases:

- An Inexact exception may be set with an Overflow exception.
- An Inexact Exception may be set with an Underflow exception.
- An Invalid Operation exception (SNaN) is set with Invalid Operation exception ($\infty \times 0$) for *Multiply-Add* instructions for which the values being multiplied are infinity and 0, and the value being added is an SNaN.
- An Invalid Operation exception (SNaN) can be set with Invalid Operation exception (Invalid Compare) for *Compare Ordered* instructions.
- Invalid Operation exception (SNaN) can be set with Invalid Operation exception (Invalid Integer Convert) for *Convert To Integer* instructions.

When an exception occurs, instruction execution may be suppressed or a result may be delivered, depending on the exception.

Instruction execution is suppressed for the following kinds of exception, so that there is no possibility that one of the operands is lost:

- Enabled Invalid Operation
- Enabled Zero Divide

For the remaining exceptions, a result is generated and written to the target specified by the instruction causing the exception. The result may be a different value for the enabled and disabled conditions for some of these exceptions. The exceptions that deliver a result are:

- Disabled Invalid Operation
- Disabled Zero Divide
- Disabled Overflow
- Disabled Underflow
- Disabled Inexact
- Enabled Overflow
- Enabled Underflow
- Enabled Inexact

Subsequent sections define each of the floating-point exceptions and specify the action that is taken when they are detected.

IEEE 754 specifies the handling of exceptional conditions in terms of "traps" and "trap handlers." In Book-E, an FPSCR exception enable bit of 1 causes generation of the result value specified in the IEEE standard for the 'trap enabled' case. The exception is expected to be detected by software, which revises the result. An FPSCR exception enable bit of 0 causes generation of the "default result" value specified for the "trap disabled" (or "no trap occurs" or "trap is not implemented") case. Software is not expected to detect the exception, and simply uses the default result. The result to be delivered in each case for each exception is described in subsequent sections.

The IEEE 754 default behavior when an exception occurs is to generate a default value and to not notify software. In Book-E, if the IEEE 754 default behavior is desired for all exceptions, all FPSCR exception enable bits should be set to 0 and Ignore Exceptions Mode should be used (see *Table 12-2* on page 262). In this case, an Enabled exception type Program interrupt is not taken, even if floating-point exceptions occur. Software can inspect the FPSCR exception bits, if necessary, to determine whether exceptions have occurred.

If software is to be notified that a given kind of exception has occurred, the corresponding FPSCR exception enable bit must be set to 1 and a mode other than Ignore Exceptions Mode must be used. In this case, the Enabled exception type Program interrupt is taken if an enabled floating-point exception occurs. An Enabled exception type Program interrupt is also taken if an **mtsfs** or **mtsfsi** instruction sets an exception bit and its corresponding enable bit both to 1; the **mtsfs** or **mtsfsi** instruction is considered to cause the enabled exception.

MSR[FE0, FE1] control whether and how Enabled exception type Program interrupt are taken when an enabled floating-point exception occurs. An Enabled exception type Program interrupt is never taken because of a disabled floating-point exception.

Table 12-2. MSR[FE0, FE1] Modes

MSR[FE0]	MSR[FE1]	Mode
0	0	**Ignore Exceptions Mode** Floating-point exceptions do not cause an Enabled exception type Program interrupt to be taken.
1	1	**Precise Mode** An Enabled exception type Program interrupt is taken precisely at the instruction that caused the enabled exception.

If either MSR[FE0] or MSR[FE1] is 1, Enabled exception type Program interrupts are treated as in Precise Mode.

In all cases, the question of whether a floating-point result is stored, and what value is stored, is governed by the FPSCR exception enable bits, as described in subsequent sections, and is not affected by the value of MSR[FE0, FE1].

In all cases in which an Enabled exception type Program interrupt is taken, all instructions before the instruction at which the Enabled exception type Program interrupt is taken have completed, and no instruction after the instruction at which the Enabled exception type Program interrupt is taken has begun execution. The instruction at which the Enabled exception type Program interrupt is taken has not been executed unless it is the excepting instruction, in which case it has been executed if the exception is not an Enabled Invalid Operation exception or Enabled Zero Divide exception.

A **sync** instruction, or any other execution-synchronizing instruction or event, such as **isync**, also has the effects described above.

In order to obtain the best performance across the widest range of implementations, the programmer should follow these guidelines.

- If the IEEE 754 default results are acceptable to the application, Ignore Exceptions Mode should be used, with all FPSCR exception enable bits set to 0.
- Ignore Exceptions Mode should not, in general, be used when any FPSCR exception enable bits are set to 1.
- Precise Mode may degrade performance in some implementations, perhaps substantially, and therefore should be used only for debugging and other specialized applications.

12.3 Floating-Point Interrupts

The following interrupts are taken under the control of the PPC440 processor core, and are not enabled by or reported in FPSCR bits:

- Floating-Point Unavailable
- Floating-Point Assist

12.3.1 Floating-Point Unavailable Interrupt

A Floating-Point Unavailable interrupt occurs when no higher priority exception exists, an attempt is made to execute a floating-point instruction (including floating-point loads, stores, and moves), and MSR[FP] = 0.

When a Floating-Point Unavailable interrupt occurs, the processor suppresses the execution of the instruction causing the Floating-Point Unavailable interrupt.

12.4 Floating-Point Exception Behavior

The following sections describe the behavior that results from the floating-point exceptions. For each exception, the definition of the exception is given, followed by a description of the action caused by the exception.

In general, each exception can result in either of two types of action, depending on whether the exception is enabled by its associated exception enable bit in the FPSCR.

12.4.1 Invalid Operation Exception

An Invalid Operation exception occurs when an operand is invalid for the specified operation. The invalid operations are:

- Any floating-point operation on a signaling NaN (SNaN)
- For add or subtract operations, magnitude subtraction of infinities ($\infty - \infty$)
- Division of infinity by infinity ($\infty \div \infty$)
- Division of zero by zero ($0 \div 0$)
- Multiplication of infinity by zero ($\infty \times 0$)
- Ordered comparison involving a NaN (Invalid Compare)
- Square root or reciprocal square root of a negative and nonzero number (Invalid Square Root)
- Integer conversion involving a number too large in magnitude to be represented in the target format, or involving an infinity or a NaN (Invalid Integer Convert)

In addition, an Invalid Operation exception occurs if software explicitly requests this by executing an **mtfsf**, **mtfsfi**, or **mtfsb1** instruction that sets FPSCR[VXSOFT] = 1.

Programming Note: The purpose of FPSCR[VXSOFT] is to enable software to cause an Invalid Operation exception for a condition that is not necessarily associated with the execution of a floating-point instruction. For example, it may be set by a program that computes a square root, if the source operand is negative.

12.4.1.1 Action

The action taken depends on the setting of FPSCR[VE].

When Invalid Operation exception is enabled (FPSCR[VE] = 1) and an Invalid Operation exception occurs or software explicitly requests the exception, the following actions are taken:

- One or two FPSCR Invalid Operation exception bits, listed in *Table 12-3*, are set.

Table 12-3. Invalid Operation Exceptions

FPSCR Bit	Category
VXSNAN	SNaN
VXISI	Infinity – Infinity
VXIDI	Infinity ÷ Infinity
VXZDZ	Zero ÷ Zero
VXIMZ	Infinity × Zero
VXVC	Invalid Compare
VXSOFT	Software Request
VXSQRT	Invalid Square Root
VXCVI	Invalid Integer Convert

- If the operation is an arithmetic, **frsp**, or convert to integer operation, the target FPR is unchanged.
  - FPSCR[FR, FI] ← 0
  - FPSCR[FPRF] ← unchanged
- If the operation is a compare:
  - FPSCR[FR, FI, C] ← unchanged
  - FPSCR[FPCC] ← unordered
- If software explicitly requests the exception:
  - FPSCR[FR, FI, FPRF] are as set by the **mtfsf**, **mtfsfi**, or **mtfsb1** instruction.

When Invalid Operation exception is disabled (FPSCR[VE] = 0) and an Invalid Operation exception occurs, or software explicitly requests the exception, the following actions are taken:

- One or two FPSCR Invalid Operation exception bits, listed in *Table 12-3*, are set.
- If the operation is an arithmetic or *Floating Round to Single-Precision* operation, the target FPR is set to a Quiet NaN
  - FPSCR[FR, FI] ← 0
  - FPSCR[FPRF] ← the class of the result (Quiet NaN)
- If the operation is a convert to 32-bit integer operation, the target FPR is set as follows:
  - $FPR(FRT)_{0:31}$ ← undefined
  - $FPR(FRT)_{32:63}$ are set to the most positive 32-bit integer if the operand in FPR(FRB) is a positive number or +∞, and to the most negative 32-bit integer if the operand in FPR(FRB) is a negative number, -∞, or NaN.
  - FPSCR[FR, FI] ← 0
  - FPSCR[FPRF] ← undefined

- If the operation is a compare:
  - FPSCR[FR, FI, C] ← unchanged
  - FPSCR[FPCC] ← unordered
- If software explicitly requests the exception:
  - FPSCR[FR, FI, FPRF] are as set by the **mtfsf**, **mtfsfi**, or **mtfsb1** instruction.

12.4.2 Zero Divide Exception

A Zero Divide exception occurs when an **fdiv[s]** instruction is executed with a zero divisor value and a finite nonzero dividend value. This exception also occurs when a *Reciprocal Estimate* instruction (**fres** or **frsqrte**) is executed with an operand value of zero.

12.4.2.1 Action

The action to be taken depends on the setting of FPSCR[ZE].

When Zero Divide exception is enabled (FPSCR[ZE] = 1) and Zero Divide occurs, the following actions are taken:

- The Zero Divide exception bit is set.

  $FPSCR_{ZX} \leftarrow 1$
- $FPR(FRT)_{0:31} \leftarrow$ unchanged
- FPSCR[FR, FI] ← 0
- FPSCR[FPRF] ← unchanged

When Zero Divide exception is disabled (FPSCR[ZE] = 0) and zero divide occurs, the following actions are taken:

- The Zero Divide exception bit is set.

  $FPSCR_{ZX} \leftarrow 1$
- FPR(FRT) ← ±Infinity (the sign is determined by the XOR of the signs of the operands)
- FPSCR[FR, FI] ← 0
- FPSCR[FPRF] ← class and sign of the result (±Infinity)

12.4.3 Overflow Exception

Overflow occurs when the magnitude of what would have been the rounded result, if the exponent range were unbounded, exceeds that of the largest finite number of the specified result precision.

12.4.3.1 Action

The action to be taken depends on the setting of FPSCR[OE].

When Overflow exceptions re enabled (FPSCR[OE] = 1) and exponent overflow occurs, the following actions are taken:

- Overflow Exception is set

  FPSCR[OX] ← 1
- For double-precision arithmetic instructions, the exponent of the normalized intermediate result is adjusted by subtracting 1536.

- For single-precision arithmetic instructions and the **frsp** instruction, the exponent of the normalized intermediate result is adjusted by subtracting 192.
- FPR(FRT) ← adjusted rounded result
- FPSCR[FPRF] ← class and sign of the result (±Normal Number)

When Overflow Exception is disabled (FPSCR[OE] = 0) and overflow occurs, the following actions are taken:

- Overflow Exception is set

  FPSCR[OX] ← 1
- Inexact Exception is set

  FPSCR[XX] ← 1
- The result is determined by the rounding mode (FPSCR[RN]) and the sign of the intermediate result as follows:
  - Round to Nearest
    Store ± Infinity, where the sign is the sign of the intermediate result
  - Round toward Zero
    Store the format's largest finite number with the sign of the intermediate result
  - Round toward +Infinity
    For negative overflow, store the format's most negative finite number; for positive overflow, store +Infinity
  - Round toward –Infinity
    For negative overflow, store –Infinity; for positive overflow, store the largest finite number of the format
- FPR(FRT) ← result
- FPSCR[FR] ← undefined
- FPSCR[FI] ← 1
- FPSCR[FPRF] ← class and sign of the result (±Infinity or ±Normal Number)

12.4.4 Underflow Exception

Underflow Exception is defined separately for the enabled and disabled states:

- Enabled:
  Underflow occurs when the intermediate result is "Tiny."
- Disabled:
  Underflow occurs when the intermediate result is "Tiny" and there is "Loss of Accuracy."

A "Tiny" result is detected before rounding, when a nonzero intermediate result computed as though both the precision and the exponent range were unbounded would be less in magnitude than the smallest normalized number.

If the intermediate result is "Tiny" and Underflow Exception is disabled (FPSCR[UE] = 0), the intermediate result is denormalized (See *Normalization and Denormalization* on page 169) and rounded (See *Rounding* on page 170) before being placed into the target FPR.

"Loss of Accuracy" is detected when the delivered result value differs from what would have been computed were both the precision and the exponent range unbounded.

12.4.4.1 Action

The action to be taken depends on the setting of FPSCR[UE].

When Underflow exception is enabled (FPSCR[UE] = 1) and exponent underflow occurs, the following actions are taken:

- Underflow Exception is set
  FPSCR[UX] ← 1
- For double-precision arithmetic instructions, the exponent of the normalized intermediate result is adjusted by adding 1536
- For single-precision arithmetic instructions and the **frsp** instruction, the exponent of the normalized intermediate result is adjusted by adding 192
- The adjusted rounded result is placed into the target FPR

  FPSCR[FPRF] ← class and sign of the result (±Normalized Number)

  Programming Note: The FR and FI bits are provided to allow the Enabled exception type Program interrupt, when taken because of an Underflow Exception, to simulate a 'trap disabled' environment. That is, the FR and FI bits allow the Enabled exception type Program interrupt to unround the result, thus allowing the result to be denormalized.

When Underflow Exception is disabled (FPSCR[UE] = 0) and underflow occurs, the following actions are taken:

- Underflow Exception is set

  FPSCR[UX] ← 1
- FPR(FRT) ← rounded result
- FPSCR[FPRF] ← class and sign of the result (±Normalized Number, ±Denormalized Number, or ±Zero)

12.4.5 Inexact Exception

An Inexact Exception occurs when either of the following conditions occur during rounding:

- The rounded result differs from the intermediate result, assuming both the precision and the exponent range of the intermediate result to be unbounded. In this case, the result is said to be inexact. If the rounding causes an enabled Overflow Exception or an enabled Underflow Exception, an Inexact Exception also occurs only if the significands of the rounded result and the intermediate result differ.)
- The rounded result overflows and Overflow Exception is disabled.

12.4.5.1 Action

The action to be taken does not depend on the setting of FPSCR[XX].

When Inexact Exception occurs, the following actions are taken:

- Inexact Exception is set

  FPSCR[XX] ← 1
- FPR(FRT) ← rounded or overflowed result
- FPSCR[FPRF] ← class and sign of the result

  Programming Note: In some implementations, enabling Inexact Exceptions may degrade performance more than does enabling other types of floating-point exception.

12.5 Exception Priorities for Floating-Point Load and Store Instructions

The following prioritized list of exceptions may occur as a result of the attempted execution of any *Floating-Point Load* and *Store* instruction.

1. Debug (Instruction Address Compare)
2. Instruction TLB Error (all types)
3. Instruction Storage Interrupt (all types)
4. Program (Illegal Instruction)
5. Floating-Point Unavailable
6. Program (Unimplemented Operation)
7. Data TLB Error (all types)
8. Data Storage (all types)
9. Alignment
10. Debug (Data Address Compare, Data Value Compare)
11. Debug (Instruction Complete)

If an instruction causes both a Debug (Instruction Address Compare) exception, and a Debug (Data Address Compare) or Debug (Data Value Compare) exception, and does not cause any exception listed in items 2–9, both exceptions can be generated and recorded in the Debug Status Register (DBSR). A single Debug interrupt results.

12.6 Exception Priorities for other Floating-Point Instructions

The following prioritized list of exceptions may occur as a result of the attempted execution of any floating-point instruction other than a load or store.

1. Debug (Instruction Address Compare)
2. Instruction TLB Error (all types)
3. Instruction Storage Interrupt (all types)
4. Program (Illegal Instruction)
5. Floating-Point Unavailable
6. Program (Unimplemented Operation)
7. Program (Enabled)
8. Debug (Instruction Complete)

12.7 QNaN

If any of the source operands is a NaN, either signaling (SNaN) or quiet (QNaN), the result will be that NaN with the high-order fraction bit forced to 1 (that is, forced to a QNaN). The precedence, in decreasing order, is FRA, FRB, FRC. The resultant QNaN is only truncated on an **frsp**[**.**] instruction, in which case the most significant 35 bits are copied to the target, with the least significant 29 forced to zero.

Table 12-4. QNaN result

R_a	R_b	R_c	Resultant QNaN[a]
NaN	X	X	R_a
—	—	X	R_b[b]
—	—	NaN	R_c

a.High-order fraction bit is forced to a 1
b.**frsp**: Result is $(FRB)_{0:11}$ || 1 || $(FRB)_{13:34}$|| $^{29}0$?

12.8 Updating FPRs on Exceptions

The target FPR is never updated on enabled invalid exceptions and enabled divide by zero exceptions. This requirement exists because an instruction may potentially use one of the source registers as a target register, yet it is necessary that the trap handler be able to examine and act upon the source operands.

In all other cases, a floating-point exception does not block the writing of the target FPR.

12.9 Floating-Point Status and Control Register

The computational instructions modify the FPSCR. With the exception of instructions which write directly to the FPSCR, none of the noncomputational instructions modify the FPSCR.

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. $FPSCR_{0:23}$ are status bits. $FPSCR_{24:31}$ are control bits.

The exception bits in the FPSCR (bits 3:12, 21:23) are sticky; that is, once set to 1 they remain set to 1 until they are set to 0 by an **mcrfs**, **mtfsfi**, **mtfsf**, or **mtfsb0** instruction. The exception summary bits FPSCR[FX, FEX, VX] are not considered as "exception bits," and only FPSCR[FX] is sticky.

FPSCR[FEX, VX] are simply ORs of other FPSCR bits. Therefore, these bits are not listed among the FPSCR bits affected by the various instructions.

FPSCR[FPRF], which contains five result flag bits, is set for arithmetic, rounding, and conversion instructions based on the class of the result value placed into the target FPR. If any portion of a result is undefined, the value placed into FPSCR[FPRF] is undefined. *Table 12-5* describes how the values of the result flags in FPSCR[FPRF] correspond to the result value classes.

Table 12-5. FPSCR[FPRF] Result Flags

Result Flags					Result Value Class
C	<	>	=	?	
1	0	0	0	1	Quiet NaN
0	1	0	0	0	–Infinity

Table 12-5. FPSCR[FPRF] Result Flags

Result Flags					Result Value Class
C	<	>	=	?	
0	1	0	0	0	–Normalized Number
1	1	0	1	0	–Denormalized Number
1	0	0	1	0	–Zero
0	0	0	0	0	+Zero
1	0	1	0	0	+Denormalized Number
0	0	1	0	0	+Normalized Number
0	0	1	0	1	+Infinity

Figure 5-2 on page 162 illustrates the FPSCR.

12.10 Updating the CR

Architecturally, excepting floating-point instructions do not block the updating of the CR in the PPC440 processor core. However, the PPC440 FPU blocks CR updates and requires software assistance to make them.

12.10.1 CR Fields

The CR fields are modified by various floating-point instructions.

Figure 12-1. Condition Register (CR)

0:3	CR0	Condition Register Field 0	
4:7	CR1	Condition Register Field 1	
8:11	CR2	Condition Register Field 2	
12:15	CR3	Condition Register Field 3	
16:19	CR4	Condition Register Field 4	
20:23	CR5	Condition Register Field 5	
24:27	CR6	Condition Register Field 6	
28:31	CR7	Condition Register Field 7	

12.10.2 Updating CR Fields

The floating-point compare instructions **fcmpo** and **fcmpu** specify a CR field that is updated with the compare results. update a field (specified by the instruction) of the CR.

Table 12-6 illustrates the bit encodings for a CR field containing the results of an **fcmpo** and **fcmpu** instruction.

Table 12-6. Bit Encodings for a CR Field

CR Field (Bit)	Description
0	*Floating-Point Less Than* (FL) Floating-point compare: (FRA) < (FRB)
1	*Floating-Point Greater Than* (FG) Floating-point compare: (FRA) > (FRB)
2	*Floating-Point Equal* (FE) Floating-point compare: (FRA) = (FRB)
3	*Floating-Point Unordered* (FU) Floating-point compare: One or both of (FRA) or (FRB) is a NaN.

The **mcrfs** instruction moves a specified FPSCR field into a CR field.

12.10.3 Generation of QNaN Results

If a disabled Invalid Operation exception is caused by operating on a NaN, the value returned follows the rules indicated in *Table 12-4* on page 269.

If the exception was not caused by operating on a NaN, a QNaN must be generated. The generated QNaN has a sign bit of 0, an exponent of all 1s, a high-order fraction bit of 1 with all other fraction bits of 0: 0x7FF8000000000000.

13. Timer Facilities

The PPC440EP provides four timer facilities: a time base, a Decrementer (DEC), a Fixed Interval Timer (FIT), and a Watchdog Timer. These facilities, which share the same source clock frequency, can support:

- Time-of-day functions
- General software timing functions
- Peripherals requiring periodic service
- General system maintenance
- System error recovery

These timer facilities are part of the PPC440 processor core. Refer to the *PPC440 Processor User's Manual* for details.

14. General Purpose Timers

The General Purpose Timer (GPT) is a system timer with seven maskable compare registers and a 32-bit time base counter. Each compare register has a corresponding GPT interrupt to the UIC. GPT interrupts can be generated for a specific count by a match between the contents of a compare register and the time base counter. GPT interrupts may also be generated on a specific interval by masking individual bits of a compare register. The following sections provide an overview, programming steps and register descriptions.

14.1 GPT Features

- 32-bit time base counter driven by the OPB bus clock
- OPB slave interface for access to all control, timer and status registers which provide direct control of all GPT functions
- Seven compare timers and corresponding interrupt outputs
- Programmable time base register (sets the time base counter)
- Maskable time-base comparison support for each compare timer
- Programmable compare timer values
- Enable/disable control of all compare interrupts
- Mask control of interrupt status bits

14.2 Time Base Counter

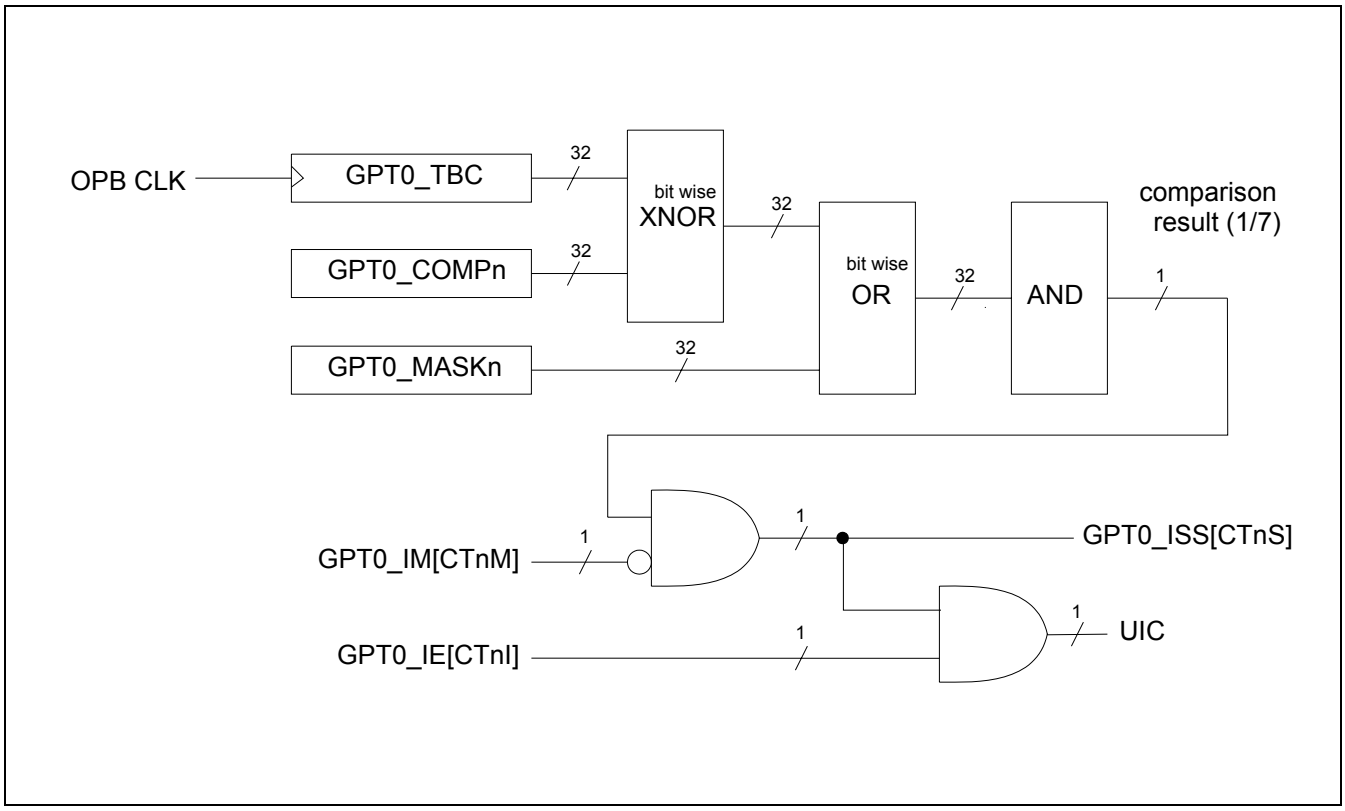
The Time Base Counter (TBC) is both an OPB register and an unsigned counter, and provides the reference time for all compare timers. It increments by one with each clock period and is 32-bit wide. When the time base counter is at its maximum value (all bits set to 1) it rolls back to zero upon the next clock.

The TBC is synchronously reset to zero upon a full chip reset. It may be read and written via software through the OPB interface. When written the new value is stored with the next rising edge of OPBClk.

14.3 Compare Timers

The time base counter, GPT0_TBC, is incremented each OPB clock period and evaluated for equivalence to the compare registers as illustrated in *Figure 14-1*. The XNOR identifies bits in the time base counter that are equivalent to corresponding bits in a compare register GTP0_COMP0:6. The 32-bit output of the XNOR is OR'ed with a 32-bit compare mask GPT0_MASK0:6. Bits set to 1 in the compare mask are masked and are not evaluated for equivalence. The comparison result of the 32-bit input AND reduces the masked result of the OR to a single bit indicating equivalence if set to 1. The status register GPT0_ISS records the comparison if the corresponding compare result is unmasked by the interrupt mask register field GPT0_IM[CTnM] = 0. To generate a GPT interrupt the interrupt enable bit GPT0_IE[CTnI] = 1 must also be enabled. *Figure 14-1* illustrates the comparison of the time base counter to a compare register.

Figure 14-1. Time Base Counter and Compare Register



14.3.1 Compare Timer Interrupt

The following are steps for enabling a GPT interrupt:

1. Set the corresponding compare register (GPT0_COMPn) to the desired compare value.
2. Set the corresponding compare mask register (GPT0_MASKn) with the desired mask bit pattern.
3. Unmask the GPT interrupt by clearing the interrupt mask bit (GPT0_IM[CTnM]=0).
4. Enable the GPT interrupt by setting the enable bit (GPT0_IE[CTnI]=1).
5. Configure the UIC to enable a GPT interrupt (see *Universal Interrupt Controller* on page 223).

Note: Seven separate interrupt lines, (UIC 16:22) one for each of the seven compare timers are implemented.

14.4 GPT Registers

The GPT device control registers listed in *Table 14-1* are accessed using the move from device control register (**mfdcr**) and move to device control register (**mtdcr**) instructions. All GPT registers are memory mapped and accessed via load/store instructions at the address of the register. The registers are accessed from the OPB on 32-

bit boundaries relative to the configurable base address. The GPT Interrupt Status Register (GPT0_IS) bits are either set or cleared when written, depending upon which one of two addresses are used. All other registers are both read and write accessible in the normal manner.

Table 14-1. GPT Registers

Mnemonic	Register	Address	Access	Page
GPT0_TBC	Time Base Counter	0x0 EF60 0000	R/W	277
GPT0_IM	GPT Interrupt Mask	0x0 EF60 0018	R/W	278
GPT0_ISS	GPT Interrupt Status (Set bits if write 1)	0x0 EF60 001C	R/W	278
GPT0_ISC	GPT Interrupt Status (Clear bits if write 1)	0x0 EF60 0020	R/W	278
GPT0_IE	GPT Interrupt Enable	0x0 EF60 0024	R/W	279
GPT0_COMP0	Compare Timer 0	0x0 EF60 0080	R/W	280
GPT0_COMP1	Compare Timer 1	0x0 EF60 0084	R/W	280
GPT0_COMP2	Compare Timer 2	0x0 EF60 0088	R/W	280
GPT0_COMP3	Compare Timer 3	0x0 EF60 008C	R/W	280
GPT0_COMP4	Compare Timer 4	0x0 EF60 0090	R/W	280
GPT0_COMP5	Compare Timer 5	0x0 EF60 0094	R/W	280
GPT0_COMP6	Compare Timer 6	0x0 EF60 0098	R/W	280
GPT0_MASK0	Compare Mask (Compare Timer 0)	0x0 EF60 00C0	R/W	280
GPT0_MASK1	Compare Mask (Compare Timer 1)	0x0 EF60 00C4	R/W	280
GPT0_MASK2	Compare Mask (Compare Timer 2)	0x0 EF60 00C8	R/W	280
GPT0_MASK3	Compare Mask (Compare Timer 3)	0x0 EF60 00CC	R/W	280
GPT0_MASK4	Compare Mask (Compare Timer 4)	0x0 EF60 00D0	R/W	280
GPT0_MASK5	Compare Mask (Compare Timer 5)	0x0 EF60 00D4	R/W	280
GPT0_MASK6	Compare Mask (Compare Timer 6)	0x0 EF60 00D8	R/W	280
GPT0_DCT0	Down Count Timer	0x0 EF60 0110	R/W	280
GPT0_DCIS	Down Count Timer Interrupt Status	0x0 EF60 011C	R/W	281

14.4.1 GPT Time Base Counter Register (GPT0_TBC)

GPT time base counter register (GPT0_TBC) is used by the compare timers as a reference for determining event occurrences and for software to use as a general timer.

Figure 14-2. GPT Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base

14.4.2 GPT Interrupt Mask Register (GPT0_IM)

GPT interrupt mask register (GPT0_IM) bits correspond to the compare timer interrupt masks. The register bits mask both the setting of the corresponding GPT0_IS bits and the interrupt output signals to the UIC. If masked, GPT0_IS bits are not set, and interrupt signals are not generated (even if the GPT0_IE bits are enabled). For interrupt signals to be active, the GPT0_IM bits must be reset (not masked) and the GPT0_IE bits must be enabled.

Figure 14-3. GPT Interrupt Mask Register (GPT0_IM)

0:15		Reserved	
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled	
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled	
18	CT2M	Compare Timer 2 Interrupt Mask 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled	
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled	
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled	
21	CT5M	Compare Timer 5 Interrupt Mask 0 Compare timer 5 interrupt mask disabled 1 Compare timer 5 interrupt mask enabled	
22	CT6M	Compare Timer 6 Interrupt Mask 0 Compare timer 6 interrupt mask disabled 1 Compare timer 6 interrupt mask enabled	
23:31		Reserved	

14.4.3 GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

GPT interrupt status register (GPT0_ISS/ICC) bits correspond to the compare timer interrupt status. The GPT Interrupt status bits for the compare timers are set when a valid comparison is made, and the compare interrupt is enabled.

GPT0_ISS can be accessed through address offset 0x1C, which provides a normal read access and a "Write-Set" access, allowing individual status bits to be set through a write access. Any status bits written to 1 are set (forced to 1), while bits written to 0 remain unchanged (0 or 1).

Offset 0x20 (GPT0_ISC) provides a normal read access and a "Write-Clear" access, which allows individual status bits to be reset through a write access. Any status bits written to 1 are cleared (forced to 0), while bits written to 0 remain unchanged (0 or 1).

Figure 14-4. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)			
0:15		Reserved	
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled	
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled	
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled	
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled	
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled	
21	CT5S	Compare Timer 5 Interrupt Status 0 Compare timer 5 interrupt status disabled 1 Compare timer 5 interrupt status enabled	
22	CT6S	Compare Timer 6 Interrupt Status 0 Compare timer 6 interrupt status disabled 1 Compare timer 6 interrupt status enabled	
23:31		Reserved	

14.4.4 GPT Interrupt Enable Register (GPT0_IE)

GPT interrupt enable register (GPT0_IE) bits correspond to the compare timer interrupt enable bits. When set, GPT0_IE bits prevent the corresponding compare interrupts from activating the GPT UIC interrupts, even if the interrupt mask (GPT0_IM) bits are set; however, these bits have no effect on the corresponding interrupt status (GPT0_IS) bits.

Figure 14-5. GPT Interrupt Enable Register (GPT0_IE)			
0:15		Reserved	
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled	
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled	
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled	
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled	

20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled	
21	CT5I	Compare Timer 5 Interrupt Enable 0 Compare timer 5 interrupt enable disabled 1 Compare timer 5 interrupt enable enabled	
22	CT6I	Compare Timer 6 Interrupt Enable 0 Compare timer 6 interrupt enable disabled 1 Compare timer 6 interrupt enable enabled	
23:31		Reserved	

14.4.5 GPT Compare Timer Registers (GPT0_COMP0:GPT0_COMP6)

Each GPT compare timer register (GPT0_COMP0:GPT0_COMP6) is programmed with the value that is continually compared to the TBC value, as filtered through each MASK register. The width of each GPT0_COMP0:GPT0_COMP6 is 32 bits.

Figure 14-6. GPT Compare Timer Register (GPT0_COMP0 - GPT0_COMP6)

0:31	COMP	Compare Timer	

14.4.6 GPT Compare Mask Registers (GPT0_MASK0:GPT0_MASK6)

GPT compare mask registers (GPT0_MASK0:GPT0_MASK6) bits are used by the compare timers to mask off the comparison (i.e., force a valid compare) of individual bits when the comparison function is performed. For bits that are set, a valid compare is always assumed, regardless of the actual value of these bits in the GPT0_COMP0:GPT0_COMP6 or GPT0_TBC registers. The width of each implemented Mask Register is 32 bits.

Figure 14-7. GPT Compare Mask Register (GPT0_MASK0:GPT0_MASK6)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.

14.4.7 GPT Down Count Timer (GPT0_DCT0)

The GPT0_DCT0 is loaded with the time or count value to be loaded into the down counter. It is a full 32-bit quantity that is decremented each TBCClk cycle until the counter reaches zero, where it stops until another value is loaded. If the register is loaded with 0xFFFFFFFF, the counter will not count.

Figure 14-8. Down Count Timer Register (GPT0_DCT0)

0:31	DCT	Time value

14.4.8 GPT Down Count Timer Interrupt Status (GPT0_DCIS)

The GPT0_DCIS is used to determine that the down count timer caused the interrupt.

Figure 14-9. Down Count Timer Register (GPT0_DCIS)

0	DCIS	Down Count Interrupt Status 0 Counter not expired 1 Counter expired	Writing a 1 resets the bit.
1:31		Reserved	

15. Clocking

Clocking in the PPC440EP is highly configurable and supports a wide range of clock ratios on the internal and external buses. PPC440EP clocking and power-on reset (CPR) registers allow for flexible power-on configuration using the IIC bootstrap controller, as well as for later adjustment after the PPC440EP has begun operation. See *Section 9.2 Bootstrap Options* on page 208 for bootstrap options and instructions on how to change clock setting after a system reset. Maximum performance at different operating frequencies is possible through the support of non integral relative frequencies for CPU and PLB clocks, including but not limited to ratios such as 3:1, 4:1, 5:1, 5:2, and 7:2.

Note: In Pass 1 only integer values are supported.

Two PLLs are used in the PPC440EP, which support both System and DDR clocking:

1. System PLL - source for CPU and PLB clocks, and indirectly the DDR SDRAM, OPB, serial and external bus clocks
2. DDR DLL - (fixed value PLL) source for DDR clock used by the DDR interface

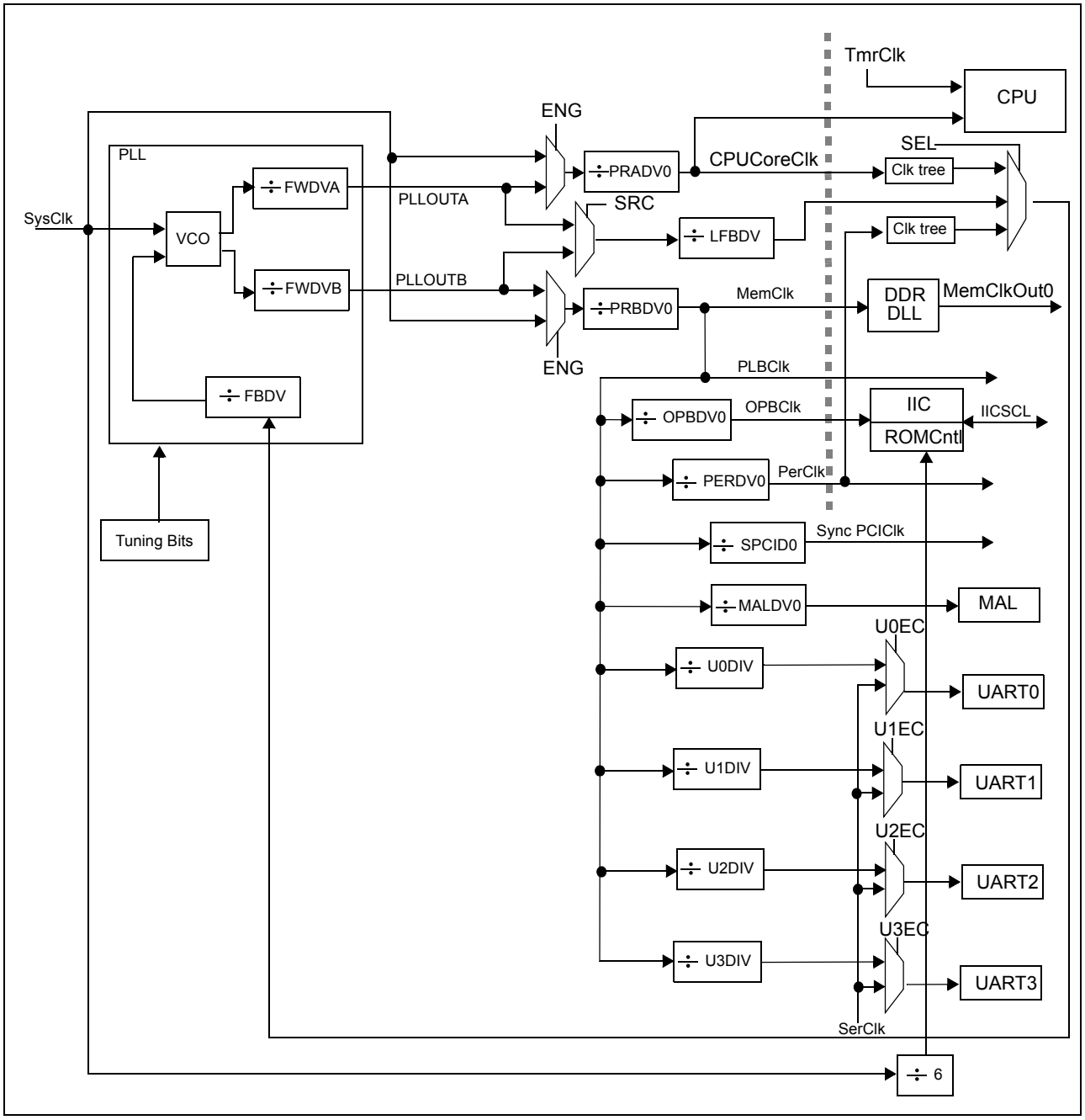
The control and configuration necessary for the system PLL is described in the following sections. Be sure to properly filter both the analog Vdd and GND inputs used by each PLL as described in detail in the PPC440EP data sheet.

PLL operation is controlled by a set of registers defined in *Clocking Registers* on page 294. The Voltage Controlled Oscillator (VCO) contained within each PLL is required to operate within the range of 600-1334 MHz. *System Clock Ratio Examples* on page 289 describe the VCO, CPU, PLB, OPB and EBC frequencies set by the clocking registers. *Figure 15-1* shows the PPC440EP clocking structure for the system PLL.

15.1 System Clocking

Figure 15-1 illustrates the clocks related to general system operation of the PPC440EP. Note that the CPU frequency is generated from PLLOUTA, while the PLB frequency is generated from PLLOUTB.

Figure 15-1. PPC440EP System Clocking



The system clock provided to the PPC440EP must be a minimum of 33MHz in order for the on-chip PLL to achieve a stable lock. An input clock above 67MHz is not supported, and the provided clock must be monotonic, and have acceptable jitter and slew rate as specified in the PPC440EP data sheet.

15.1.1 Feedback Selection

The PPC440EP clocking logic operates in four different modes: PLL bypass mode, PLL engaged with PLL local feedback (PLLOUTA or PLLOUTB), PLL engaged with CPU clock feedback, and PLL engaged with peripheral clock feedback. These modes are shown in *Table 15-1*.

Table 15-1. PPC440EP Clocking Modes

CPR0_PLLC0[ENG]	CPR0_PLLC0[SEL]	Mode
0	xxx	Bypass
1	000	PLL Local feedback
1	001	CPU clock feedback
1	101	PER clock feedback

The PLL operates in Bypass mode upon entering Reset (see *Reset and Initialization* on page 189 for additional information on reset). In this mode the system reference clock, SYSClk, replaces the output of the PLL. The frequencies of the various clocks are SYSclk divided by all the divisors in that clock's path. The PLL stays in Bypass mode until the initial configuration is determined from either the default serial rom straps or the values read in from the serial ROM device. When the initial configuration has been determined, set up, and the clocks are stable, reset is released to the rest of PPC440EP.

The PPC440EP system clocking logic is designed to use one of three feedback paths for the SYS PLL, selected by CPR0_PLLC0 [SEL]. Most users will find that local feedback provides the greatest flexibility in choosing CPU and PLB clock ratios. Users who require that the external bus clock, PerClk, be both frequency- and phase-aligned with SysClk will need to use PerClk for feedback. Phase alignment between SysClk and the PerClk clock permits the use of the external master interface with external masters that cannot use PerClk for synchronization. These external masters will normally source the external bus clock themselves and provide it to the PPC440EP SysClk input. CPU clock feedback is provided as a compatibility mode with the PPC440GP.

For PerClk and SysClk to be phase aligned, the clock divisors must be selected such that the frequencies are the same. Doing so requires that the CPR0_PLLD[FBDV] bits be set to a divide-by-one, and that PLB and other clocks derived from PLLOUTB are a multiple of PerClk. In the absence of any system requirement for phase alignment between PerClk and SysClk, users are encouraged to use PLL local feedback mode.

Note: In Bypass mode the DDR SDRAM interface may not work, because the DDR delay line is only designed to work between 100 and 133 MHz and allowed system reference clocks are 33 to 67MHz. See *DDR SDRAM Controller* on page 311 for additional information.

PPC440EP clocking mode can be changed by programming the clocking control registers to the desired values for the new clocking mode and then programming the CPU to issue a Chip Reset.

15.1.2 VCO Frequency and 'M' Value for SYS PLL

For any acceptable input SysClk frequency, the SYS PLL VCO frequency is set by the total product of divisor circuits used in the path from VCO output back to VCO input, multiplied by the system clock frequency. The product of the divisor circuits, both inside the PLL and in the external divisor circuits such as those that generate the PLB, OPB, and PerClk, is referred to as the M multiplier value. This value must be known in order to properly set the TUNE bits.

The M multiplier for the SYS PLL can be calculated using any of the equations shown in *Table 15-2*, depending upon the choice made for feedback as shown in the tables that follow.

Table 15-2. Clock Frequencies and M multiplier

Feedback Choice	M multiplier equation
PLL Local	M = FBDV x LFBDV x FWDVA (CPR0_PLLC0[SRC]=0)
PLL Local	M = FBDV x LFBDV x FWDVB (CPR0_PLLC0[SRC]=1)
CPU clock	M = FBDV x FWDVA x PRADV0
PerClk Clock	M = PERDV0 x FWDVB x PRBDV0 (FBDV=1)

As an example of the effect that various settings have on clock frequencies and the M multiplier, consider the System PLL configured first with PLL Local feedback as shown in *Table 15-3*, then with CPU feedback as shown in *Table 15-4*, and finally with PerClk feedback as shown in *Table 15-5*..

Table 15-3. System PLL Configuration Using PLL Local Feedback (CPR0_PLLC0[SEL] = 000)

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2 (PRADV0 =1)	CPU = 400MHz
FWDVB = 6 (PRBDV0 =1)	PLB = 133.3MHz
OPBDV0 = don't care	OPB = don't care
PERDV0 = don't care	PerClk = don't care
LFBDV = 3 (CPR0_PLLC0[SRC]=0)	
FBDV = 4	M = 24
M = 24	TUNE = 1100111100

Table 15-4. System PLL Configuration Using CPU Feedback (CPR0_PLLC0[SEL] = 001)

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2 (PRADV0 =1)	CPU = 400MHz
FWDVB = 6 (PRBDV0 =1)	PLB = 133.3MHz
OPBDV0 = don't care	OPB = don't care
PERDV0 = don't care	PerClk = don't care
FBDV = 12	M = 24
M = 24	TUNE = 1100111100

Table 15-5. System PLL Configuration Using PerClk Feedback (CPR0_PLLC0[SEL] = 101)

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2 (PRADV0 =1)	CPU = 400MHz
FWDVB = 6 (PRBDV0 =1)	PLB = 133.3MHz
OPBDV0= don't care	OPB = don't care
PERDV0 = 4	PerClk = 33.3 MHz
FBDV = 1	M = 24
M = 24	TUNE = 1100111100

15.1.3 SYS PLL TUNE Setting

The SYS PLL must be provided with different TUNE bit settings based upon the VCO frequency and 'M' value that result from the system configuration. *Table 15-6* describes how to determine the proper CPR0_PLLC0[TUNE] bit settings for appropriate application.

Table 15-6. CPR0_PLLC0[TUNE] Bit Settings

'M' Value Range	VCO Frequency Range	TUNE									
		0	1	2	3	4	5	6	7	8	9
$9 \leq M \leq 10$	$600MHz < VCO \leq 900MHz$	0	1	0	0	1	1	0	1	1	0
$10 < M \leq 22$	$600MHz < VCO \leq 900MHz$	1	0	0	0	1	1	1	0	0	0
	$900MHz < VCO \leq 1334MHz$	1	0	1	0	1	1	1	0	0	0
$22 < M \leq 32$	$600MHz < VCO \leq 900MHz$	1	1	0	0	1	1	1	1	0	0
	$900MHz < VCO \leq 1334MHz$	1	1	1	0	1	1	1	1	0	0

15.1.4 IIC Bootstrap Controller Clocking

SysClk is divided by 6 to provide a clock for the IIC bootstrap controller. This clock is internal to the PPC440EP and cannot be adjusted. See *Bootstrap Controller* on page 207 for additional information. Also, the IIC bootstrap controller further divides its clock by 128. Thus, for SysClk frequencies at or below 66.67MHz, a 100KHz serial ROM can be used.

15.1.5 Clocks For Off Chip Use

The DDR memory controller generates a differential MemClk to the DDR SDRAM chips at the PLB frequency. The PerClk is generated by the system clocking logic for use by an external bus master or other synchronous device. The UART serial clocks can be generated either from an internal clock divisor circuit or from the SerClk chip input. These clocks are presented internally to the UART cores, which can be further subdivided.

15.1.6 SYS PLL Strapping

System clocking is primarily controlled by the CPR0_PLLC0, CPR0_PLLD, CPR0_OPBD, CPR0_PERD, CPR0_SPCID, and SDR0_CP440 registers. These registers are normally initialized at power on time using the IIC bootstrap controller and an external serial ROM. The SDR0_SDSTP0 and SDR0_SDSTP1 registers are also initialized during this process. If a serial ROM is not present at power on time, then the pre-programmed defaults will be used.

System reset always re-loads the SDR0_SDSTP0, SDR0_SDSTP1, SDR0_CUST0 and SDR0_CUST1 registers using the values programmed into the serial bootstrap ROM.

Several different bit values in the clocking registers are required to correctly configure the chip in a way that will allow reliable operation of SYS PLL and clock divisor circuitry. When setting these values be careful to avoid accidentally configuring the PPC440EP in an unusable state. See the following section for assistance in selecting acceptable system clocking divisor ratios.

15.1.7 PLL Bypass (Emulation Mode)

The CPR0_PLLD[ENG] bit is provided to disable the SYS PLL if it is intentionally being used outside of its operating range. The most common reason for using this mode is in emulation systems used in product development, where the SYS CLK input to PPC440EP is set to a frequency well below 33MHz, perhaps as low as 1MHz. Forcing the SYS PLL into bypass mode directly feeds SYSClk to the divisor logic, allowing parts of the PPC440EP to operate well outside of its supported range (see the PPC440EP data sheet for supported range).

15.1.8 CPU / PLB Frequency N:1 Setting

The PPC440EP clocking logic allows for CPU:PLB clock frequencies that are in the ratio of N:X, where N = {1,2,...,etc} and X = {1, 2}. The CPU clock is considered to be an integral multiple of PLB when X = 1, as in the example where CPU = 400 and PLB = 133 (N:X = 3:1). A non-integral CPU to PLB relationship can be seen when X = 2, as in the example where CPU = 333 and PLB =133 (N:X = 5:2).

Note: a non-integral CPU to PLB is not supported in Pass1

These ratios are set by the values chosen for the divisors in the clocking tree. See *System Clock Ratio Examples* on page 289 to determine valid CPU and PLB frequencies and their resultant CPU:PLB, N:X, ratio.*Table 15-7* indicates the results of programming SDR0_CP440[Nto1] to either 0 or 1, depending upon whether the CPU:PLB ratio is N:1.

Table 15-7. SDR0_CP440[Nto1] Settings

CPU:PLB Ratio	SDR0_CP440[Nto1] = 0	SDR0_CP440[Nto1] = 1
CPU:PLB Ratio is N:1 (X = {1})	Slight performance loss as unnecessary clock resynchronization occurs at CPU:PLB interface	Optimal performance for N to 1 mode
CPU:PLB Ratio is not N:1 (X = {2})	Optimal performance for non N to 1 mode	System failure can occur

15.1.9 Choosing System Clock Ratios

Table 15-8 describes the rules that apply when choosing acceptable system clock ratios and frequencies for the PPC440EP. Please see the current data sheet as different PPC440EP revisions may have different limits than those described here.

Table 15-8. System Clock Rules

Rule	Value
CPU frequency must be less than or equal to	Maximum CPU frequency - see PPC440EP data sheet
PLB frequency must be less than or equal to	Maximum PLB frequency - see PPC440EP data sheet
CPU:PLB ratio is N:X with {N, X} being integer and X less than or equal to	2
VCO frequency must be greater than or equal to	600 MHz
VCO frequency must be less than or equal to	Maximum VCO frequency - see PPC440EP data sheet

Table 15-9 describes the equations that can be used to determine the resulting VCO, CPU, PLB frequencies based on selected divisor settings.

Table 15-9. Equations to Determine VCO, CPU, PLB Frequency

Feedback Selection	Equations
PLL Local	M = FBDV x LFBDV x FWDVA (CPR0_PLLC0[SRC]=0) or M = FBDV x LFBDV x FWDVB (CPR0_PLLC0[SRC]=1) VCO = SysClk x M CPU = VCO / FWDVA / PRADV0 PLB = VCO / FWDVB / PRBDV0
CPU clock	M = FBDV x FWDVA x PRADV0 VCO = SysClk x M CPU = VCO / FWDVA / PRADV0 PLB = VCO / FWDVB / PRBDV0
PerClk	M = PERDV0 x FWDVB x PRBDV0 VCO = SysClk x M CPU = VCO / FWDVA / PRADV0 PLB = VCO / FWDVB / PRBDV0

15.1.10 System Clock Ratio Examples

Example tables are provided in the following sections for the case where SysClk = 33.3MHz and PLL Local feedback, CPU feedback or PerClk feedback are used. These tables describe possible combinations of clocking ratios which can be achieved using the bit values in CPR0_PLLC0, CPR0_PLLD, CPR0_PRIMAD, CPR0_PRIMBD, CPR0_PERD,CPR0_OPBD and CPR0_SPCID.

Note: The value of PLLOUTA and PLLOUTB must not exceed the rated speed of the processor core.

15.1.10.1 PLL Local Feedback Example

As shown in *Table 15-10*, when using PLL local feedback, M and VCO can be determined once specific values for the FWDVA, LFBDV and FBDV divisors are chosen. Therefore, the PLL local feedback table is organized so that each row represents a specific combination of FWDVA, LFBDV and FBDV divisor. From these specified values, M and VCO are known and listed. Given a value of 1 for PRAVD0 the CPU frequency is also known and listed.

For each combination of these divisors, there are up to seven different columns on the right hand of the table which indicate the possible PLB frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVB and PRBDV0 divisor values.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 600 MHz or above 1334 MHz, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Table 15-10 provides clock ratio listing with PLL Local feedback, PLLOUTA (CPR0_PLLC0[SRC]=0), for CPR0_PRIMAD0[PRADV0] = 1, SysClk = 33.3 MHz, 600 MHz ≤ VCO ≤ 1334 MHz, CPU ≤ 667 MHz, 100 MHZ ≤ PLB ≤ 133 MHZ, with CPU to PLB clock ratios of 5:2, 3:1, 7:2 and 4:1. See the data sheet for maximum CPU and VCO frequencies.

Table 15-10. Clock Ratio Listing With PLL Local Feedback, SysClk = 33.3 MHz

FWDVB ====>						2	3	4	5	6	7	8
CPU:PLB Ratio ====>						1:1	3:2	2:1	5:2	3:1	7:2	4:1
PRVDB0	FWDVA	FBDV x LFBDV	M	VCO	CPU	PLB Clock Frequency (MHz)						
1	2	1-8	1-16	<600	-							
1	2	9	18	600.0	300.0				120.0	100.0		
1	2	10	20	666.7	333.3				133.3	111.1		
1	2	11	22	733.3	366.7					122.2	104.8	
1	2	12	24	800.0	400.0					133.3	114.3	100.0
1	2	13	26	866.7	433.3						123.8	108.3
1	2	14	28	933.3	466.7						133.33	116.7
1	2	15	30	1000.0	500.0							125.0
1	2	16	32	1066.7	533.3							133.3
1	1	20	20	666.7	666.7				133.3	111.1		
2	2	20	40	1333.3	666.7				133.3	111.1		

15.1.10.2 CPU Feedback Example

As shown in *Table 15-11*, when using CPU feedback, M, VCO and CPU can be determined once specific values for the FWDVA, PRADV0 and FBDV divisors are chosen. Therefore, the CPU feedback table is organized so that each row represents a specific combination of FWDVA, PRADV0 and FBDV divisor. From these specified values, M, VCO and CPU are known and listed.

For each combination of these divisors, there are up to seven different columns on the right hand of the table which indicate the possible PLB frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVB and PRBDV0 divisor values.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 600 MHz or above 1334 MHz, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Table 15-11 provides clock ratio listing with CPU feedback for CPR0_PRIMAD0[PRADV0]=1, SysClk = 33.3MHz, 600MHz ≤ VCO ≤ 1334MHz, CPU ≤ 667MHz, 133 MHZ ≤ PLB ≤ 166 MHZ, with CPU to PLB clock ratios of 5:2, 3:1, 7:2 and 4:1. See the data sheet for maximum CPU and VCO frequencies.

Table 15-11. Clock Ratio Listing With CPU Feedback, SysClk = 33.3MHz

FWDVB ====>						2	3	4	5	6	7	8
CPU:PLB Ratio ====>						1:1	3:2	2:1	5:2	3:1	7:2	4:1
PRVDB0	FWDVA	FBDV	M	VCO	CPU	PLB Clock Frequency (MHz)						
1	2	1-8	1-16	<600	-							
1	2	9	18	600.0	300.0			150				
1	2	10	20	666.7	333.3			166.5	133.3			
1	2	11	22	733.3	366.7				146.4			
1	2	12	24	800.0	400.0				160.0	133.3		
1	2	13	26	866.7	433.3					144.3		
1	2	14	28	933.3	466.7					155.3	133.33	
1	2	15	30	1000.0	500.0					166.6	142.8	
1	2	16	32	1066.7	533.3						152.4	133.3
1	1	20	20	666.7	666.7				133.3	111.1		
2	2	20	40	1333.3	666.7				133.3	111.1		

15.1.10.3 PerClk Feedback Example

As shown in the System Clock Rules table, when using PerClk feedback, M, VCO and PLB can be determined once specific values for the FWDVB, PRBDV0, PERDV0 divisors are chosen. Therefore, the PerClk feedback table is organized so that each row represents a specific combination of FWDVB and PERDV0 with PRBDV0 =1. From these specified values, M, VCO and CPU are known and listed.

Since PERDV0 can vary from a divide by 1 up to a divide by 4, the possible values are 1, 2, 3, 4,. Furthermore, when using external bus feedback, PerClk has the same frequency as SysClk. This in turn implies that FBDV is 1 and PLB = SysClk x PERDV0. *Table 15-12* demonstrates this fact by listing possible PLB frequencies which are all multiples of SysClk (that is, 100MHz, and 133.3MHz, assuming SysClk = 33.3MHz.

For each combination of these divisors, there are up to eight different columns on the right hand of the table which indicate the possible CPU frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVA divisor value.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 600MHz or above 1334MHZ, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Note: *Table 15-12* provides clock ratio listing with PerClk feedback for CPR0_PRIMAD0[PRADV0]=1, SysClk = 33.3MHz, 600MHz ≤ VCO ≤ 1334 MHz, CPU ≤ 667MHz, 100MHZ ≤ PLB ≤ 133MHZ, with CPU to PLB clock ratios of 5:2, 3:1, 7:2 and 4:1. See the data sheet for maximum CPU and VCO frequencies.

Table 15-12. Clock Ratio Listing With PerClk feedback, SysClk = 33.3MHz

FWDVA ====>						1	2	3	4	5	6	7	8
PRBDV0	PERDV0	FWDVB	M	VCO	PLB	CPU Clock Frequency (MHz)							
1	1	1-8		<600	-								
1	2	1-8		<600	-								
1	3	1-5		<600	-								
1	3	6	18	600.0	100.0		300.0						
1	3	7	21	700.0	100.0		350.0						
1	3	8	24	800.0	100.0		400.0						
1	4	1-4		<600									
1	4	5	20	666.7	133.3	666.7	333.3						
1	4	6	24	800.0	133.3		400.0						
1	4	7	28	933.3	133.3		466.7						
1	4	8	32	1066.7	133.3		533.3						
2	4	5	40	1333.3	133.3		666.7		333.3				

Table 15-13 shows the CPU:PLB ratios that result from the indicated FWDVA and FWDVB divisors when using PerClk feedback when SysClk is 33MHz. Several ratios are left blank because they are illegal.

Table 15-13. CPU:PLB Ratio

	FWDVA Divisor							
FWDVB	1	2	3	4	5	6	7	8
1								
2								
3	3:1							
4	4:1							
5		5:2						
6								
7		7:2						
8								

15.2 PCI Clocking

The following clocks are related to the PCI logic:

- The on-chip PLB clock
- The on-chip synchronous PCI clock

- An external asynchronous PCI clock, PCIClk

The PPC440EP supports only asynchronous mode. In asynchronous PCI mode, the asynchronous PCI clock attaches to logic that interfaces with the PCI bus. The synchronous clock must always be provided to the PCI logic, even when the asynchronous PCI clock is used. *Table 15-14* describes the relationships between the synchronous PCI clock, the asynchronous PCI clock, and the PLB clock.

In asynchronous PCI mode, the synchronous PCI clock must meet certain requirements. The following equation describes the relationship that must be maintained between the asynchronous PCI clock and synchronous PCI clock. Select an appropriate PCI:PLB ratio to maintain the relationship:

$$\text{AsyncPCIClk} - 1\text{MHz} \leq \text{SyncPCIclock} \leq (2 \times \text{AsyncPCIClk}) - 1\text{MHz}$$

Table 15-14 lists supported and commonly used combinations of synchronous and asynchronous PCI clocks. In general, higher synchronous PCI clock frequencies provides better performance, while lower synchronous PCI clock frequencies minimize power consumption.

Table 15-14. Example Synchronous PCI Clock Frequencies in Asynchronous Mode

Asynchronous PCI Frequency	Synchronous PCI Frequency	PLB Frequency	Sync PCI:PLB Ratio (CPR0_SPCID[SPCID0])
20 MHz	33.3 MHz	133.3 MHz	1:4
	33.3 MHz	100 MHz	1:3
	27.6 MHz	83.3 MHz	1:3
33.3 MHz	44.4 MHz	133.3 MHz	1:3
	33.3 MHz	133.3 MHz	1:4
	50 MHz	100 MHz	1:2
	41.6 MHz	83.3 MHz	1:2
40 MHz	44.4 MHz	133.3 MHz	1:3
	50 MHz	100 MHz	1:2
	41.6 MHz	83.3 MHz	1:2
66.6 MHz	66.6 MHz	133.3 MHz	1:2
	100 MHz	100 MHz	1:1
	83.3 MHz	83.3 MHz	1:1

15.2.1 PCI Adapter Applications

Because various systems run PCI expansion buses at different PCI frequencies, several PCI clock frequencies may need to be supported when the PPC440EP is used in PCI adapters.

Asynchronous PCI mode uses an externally provided PCI clock that does not interact with an on-chip PLL, so there is no lower frequency limit imposed by loss of PLL lock. However, the requirements resulting from the relationship between the synchronous and asynchronous PCI clocks must still be satisfied.

Note: Satisfying the equation in *Section 15.2* on page 292 presents a potential problem. The divisor selection needed to set an acceptable synchronous PCI clock for a 33 MHz asynchronous PCI clock differs from the selection for a 66 MHz asynchronous PCI clock. External logic is required to detect the state of the M66 pin on the PCI adapter interface and select appropriate PPC440EP divisor values during system reset.

15.3 Clocking Registers

Table 15-16 summarizes the indirectly accessed device control registers that control clocking and power on reset in PPC440EP.These registers are accessed by using the configuration address and data registers described in *Table 15-15*.

Table 15-15. PPC440EP Clocking Control Register Access

Mnemonic	Register	Address	Access	Page
CPR0_CFGADDR	Clocking Configuration Address Register	0x000C	R/W	294
CPR0_CFGDATA	Clocking Configuration Data Register	0x000D	R/W	294

Table 15-16. PPC440EP Clocking Control Registers

Mnemonic	Register	Offset	Access	Page
CPR0_CLKUPD	Clocking Update Register	0x0020	R/W	295
CPR0_PLLC0	PLL Control Register	0x0040	R/W	295
CPR0_PLLD0	PLL Divisor Register	0x0060	R/W	296
CPR0_PRIMAD0	Primary A Divisor Register	0x0080	R/W	297
CPR0_PRIMBD0	Primary B Divisor Register	0x00A0	R/W	298
CPR0_OPBD0	OPB Clock Divisor Register	0x00C0	R/W	298
CPR0_PERD0	Peripheral Clock Divisor Register	0x00E0	R/W	298
CPR0_MALD	MAL Clock Divisor Register	0x0100	R/W	299
CPR0_SPCID	Sync PCI Clock Divisor Register	0x0120	R/W	299
CPR0_ICFG	Clock/Power Configuration Register	0x0140	R/W	196

15.3.1 Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR)

The clock/power-on reset configuration address register (CPR0_CFGADDR) is a 32-bit register used to access the PPC440EP CPR0 configuration registers.

Figure 15-2. Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR)

0:16		Reserved	
17:31		Offset	

15.3.2 Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA)

The clock/power-on reset configuration data register (CPR0_CFGDATA) is a 32-bit register used to access the PPC440EP CPR0 configuration registers.

Figure 15-3. Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA)

0:31			

15.3.3 Clocking Update Register (CPR0_CLKUPD)

The clocking update register (CPR0_CLKUPD) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register.

Note: This register is documented here for reference purposes only and should not be used to change clocking modes. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-4. Clocking Update Register (CPR0_CLKUPD)

0	BSY	Clocking Subsystem Busy 0 Clocking subsystem is stable 1 Clocking subsystem is making changes.	In Read Access Mode. When CPR0_CLKUPD[BSY=1], software needs to wait until CPR0_CLKUPD[BSY]=0 before initiating any additional changes.
	CUD	Clocking Update Delay 0 Clocking update delay disabled 1 Clocking update delay enabled	In Write Access Mode. When CPR0_CLKUPD[CUD] is set to 1, then: Clocking update action is taken. Instruct the clocking logic to begin changing clocks to the newly programmed divider values. Result. System clocks will be momentarily stopped and then restarted using the currently programmed values in the CPR0 divider registers
1	CUI	Clocking Update Immediate 0 Clocking update immediate disabled 1 Clocking update immediate enabled	When you read it this bit is always 0. In Write Access Mode. When CPR0_CLKUPD[CUI]=1: Clocking update action taken. Allow the currently programmed values in all CPR0 registers to take effect immediately. Result. The clocking subsystem is now programmed according to the current settings of all CPR0 registers
2:31		Reserved	

Note: The CPR0_CLKUPD register is unique, in that writes will automatically result in an update to the clocking subsystem, while reads of the same bit(s) will report a status. Please note that this register is documented here for reference purposes only and should not be used to change clocking modes.

15.3.4 PLL Control Register (CPR0_PLLC0)

The PLL control register (CPR0_PLLC0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-5. PLL Control Register (CPR0_PLLC0)

0	RST	Reset 0 PLL allowed to lock 1 PLL is forced into reset.	For the CPR clocking logic, the reset value of the RST bit is the complement of the ENG bit
1	ENG	Engage 0 SysClk is the source for primary forward divisor. 1 PLL's VCO is the source for primary forward divisor.	
2	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB	

3:4		Reserved	
5:7	SEL	Feedback Selection 000 PLL output (A or B) 001 CPU 101 PERClk	All other combinations are reserved Selection of PLL output A or B implies that the feedback clock is taken exactly at or close to the PLL output, and not at the end of a repowered clock tree. Using this feedback source implies that the PLL is not being used to adjust the generated clocks to be phase aligned with SysClk. The specific clock used for feedback is controlled by CPR0_PLLCR[SRC] bit 2
8:21		Reserved	
22:31	TUNE	TUNE bits	

15.3.5 PLL Divisor Register (CPR0_PLLD0)

The PLL divisor register (CPR0_PLLD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-6. PLL Divisor Register (CPR0_PLLD)

0:2		Reserved	
3:7	FBDV	PLL Feedback Divisor 00000 PLL Feedback Divisor = 32 00001 PLL Feedback Divisor = 1 00010 PLL Feedback Divisor = 2 00011 PLL Feedback Divisor = 3 .... 11111 PLL Feedback Divisor = 31	
8:11		Reserved	
12:15	FWDVA	PLL Forward Divisor A 0000 PLL Forward Divisor A = 16 0001 PLL Forward Divisor A = 1 0010 PLL Forward Divisor A = 2 0011 PLL Forward Divisor A = 3 0100 PLL Forward Divisor A = 4 0101 PLL Forward Divisor A = 5 0110 PLL Forward Divisor A = 6 0111 PLL Forward Divisor A = 7 1000 PLL Forward Divisor A = 8 1001 Reserved 1010 PLL Forward Divisor A = 10 1011 Reserved 1100 PLL Forward Divisor A = 12 1101 Reserved 1110 PLL Forward Divisor A = 14 1111 Reserved	
16:20		Reserved	

21:23	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 1 010 PLL Forward Divisor B = 2 011 PLL Forward Divisor B = 3 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 5 110 PLL Forward Divisor B = 6 111 PLL Forward Divisor B = 7	
24:25		Reserved	
26:31	LFBDV	PLL Local Feedback Divisor 00_0000 PLL local feedback divisor = 64 00_0001 PLL local feedback divisor = 1 00_0010 PLL local feedback divisor = 2 .... 11_1111 PLL local feedback divisor = 63	The LFBDV is outside of PLL and is used to allow the PLL to lock using feedback directly from a PLL output. Program this divider to a value equivalent to the divide from the PLL to some clock to allow the PLL to be switched to use feedback from that clock. This allows the PLL to compensate for the latency associated with that clock's tree.

15.3.6 Primary A Divisor Register (CPR0_PRIMAD0)

The primary A divisor register (CPR0_PRIAMD) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-7. Primary A Divisor Register (CPR0_PRIMAD0)

0:4		Reserved	
5:7	PRADV0	PLL Primary Divisor A 000 PLL Primary Divisor A = 8 001 PLL Primary Divisor A = 1 010 PLL Primary Divisor A = 2 011 PLL Primary Divisor A = 3 100 PLL Primary Divisor A = 4 101 PLL Primary Divisor A = 5 110 PLL Primary Divisor A = 6 111 PLL Primary Divisor A = 7	**Note:** If CPR0_ICFG[RLI] = 0, then the reset value for PLL Primary Divisor A is 1
8:31		Reserved	

15.3.7 Primary B Divisor Register (CPR0_PRIMBD0)

The primary B divisor register (CPR0_PRIMBD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-8. Primary B Divisor Register (CPR0_PRIMBD0)

0:4		Reserved	
5:7	PRBDV0	PLL Primary Divisor B 000 PLL Primary Divisor B = 8 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 011 PLL Primary Divisor B = 3 100 PLL Primary Divisor B = 4 101 PLL Primary Divisor B = 5 110 PLL Primary Divisor B = 6 111 PLL Primary Divisor B = 7	
8:31		Reserved	

15.3.8 OPB Clock Divisor Register (CPR0_OPBD0)

The OPB clock divisor register (CPR0_OPBD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-9. OPB Clock Divisor Register (CPR0_OPBD0)

0:5		Reserved	
6:7	OPBDV0	OPB Clock Divisor 0 00 OPB clock divisor 0 = 4 01 OPB clock divisor 0 = 1 10 OPB clock divisor 0 = 2 11 OPB clock divisor 0 = 3	
8:31		Reserved	

15.3.9 Peripheral Clock Divisor Register (CPR0_PERD0)

The peripheral clock divisor register (CPR0_PERD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-10. Peripheral Clock Divisor Register (CPR0_PERD0)			
0:4		Reserved	
5:7	PERDV0	Peripheral Clock Divisor 0 000 Peripheral clock divisor 0 = 8 001 Peripheral clock divisor 0 = 1 010 Peripheral clock divisor 0 = 2 011 Peripheral clock divisor 0 = 3 100 Peripheral clock divisor 0 = 4 101 Peripheral clock divisor 0 = 5 110 Peripheral clock divisor 0 = 6 111 Peripheral clock divisor 0 = 7	
8:31		Reserved	

15.3.10 MAL Clock Divisor Register (CPR0_MALD)

The MAL clock divisor register (CPR0_MALD) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 9.2* on page 208 for instructions on changing clocking mode.

Figure 15-11. MAL Clock Divisor Register (CPR0_MALD)			
0:5		Reserved	
6:7	MALDV0	MAL Clock Divisor 0 00 MAL clock divisor 0 = 4 01 MAL clock divisor 0 = 1 10 MAL clock divisor 0 = 2 11 MAL clock divisor 0 = 3	
8:31		Reserved	

15.3.11 Sync PCI Clock Divisor Register (CPR0_SPCID)

The PCI clock divisor register (CPR0_SPCID) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers.

Figure 15-12. Sync PCI Clock Divisor Register (CPR0_SPCID)			
0:5		Reserved	
6:7	SPCID0	PCI Clock Divisor 0 00 PCI clock divisor 0 = 4 01 PCI clock divisor 0 = 1 10 PCI clock divisor 0 = 2 11 PCI clock divisor 0 = 3	
8:31		Reserved	

16. Clock and Power Management

The PPC440EP provides a clock and power management (CPM) controller that reduces power dissipation by stopping clocks in unused or dormant functional units. Use of the CPM controller requires careful programming and special consideration to avoid compromising system and functional unit integrity.

16.1 Overview

The CPM controller supports three different types of sleep interfaces to the functional units:

- In a CPM class 1 interface, the CPM_Sleep_N signal is asserted by the CPM controller when a register bit is set by software. The functional unit is unconditionally put to sleep. There is no other communication with the functional unit.
- In a CPM class 2 interface, the functional unit uses a combination of its internal state and external inputs to determine whether or not it can be put to sleep. If sleeping is permitted, the functional unit asserts the Sleep_Req signal to the CPM controller that responds by asserting CPM_Sleep_N if the enable for that unit is set. The CPM_Sleep_N signal to a class 2 unit is deasserted when the CPM controller enable bit for that unit is reset, or when the unit deasserts its Sleep_Req signal.
- The CPM class 3 interface has a CPM_SleepInit signal that is asserted by the CPM controller to request that a functional unit go to sleep. If the unit can sleep, it asserts the Sleep_Req signal to the CPM controller. The CPM_Sleep_N signal is then asserted by the CPM controller to shut off the class 3 clocks in the functional unit. The functional unit or the CPM controller can end the sleep state. If the CPM controller enable bit for the unit is reset, the CPM controller immediately deasserts CPM_SleepInit and CPM_Sleep_N.

16.2 CPM Registers

Table 16-1 lists the registers used to program the CPM controller.

Table 16-1. CPM Registers

Register	Description	DCR Address	Access	(Bits 0:31)
CPM0_ER	CPM Enable Register	0x00B0	Read/Write	0x000000000
CPM0_FR	CPM Force Register	0x00B1	Read/Write	0x000000000
CPM0_SR	CPM Status Register	0x00B2	Read Only	0xFFFFFFFF

16.2.1 CPM Enable Register (CPM0_ER)

The CPM0_ER bits enable the process of putting a functional unit to sleep. The class of a unit determines how its interface signals are controlled when the bit associated with the unit is set to 1.

Class 1 When an associated CPM0_ER bit is set to 1, the CPM_Sleep_N signal to the class 1 unit is asserted. When the bit is set to 0, CPM_Sleep_N is deasserted.

Class 2 When an associated CPM0_ER bit is set to 1, and the Sleep_Req signal from the class 2 unit is asserted (the unit is requesting sleep state), CPM_Sleep_N to the class 2 unit is asserted. When the bit is set to 0, the CPM_Sleep_N signal is deasserted.

Class 3 When an associated CPM0_ER bit is set to 1, the CPM_SleepInit signal to the class 3 unit is asserted (the CPM controller is requesting permission to put the unit to sleep). When the class 3 unit activating the Sleep_Req in response, (the unit is giving permission to be put to sleep), CPM_Sleep_N signal to the class 3 unit is asserted. When the bit is set to 0, CPM_SleepInit and CPM_Sleep_N are deasserted.

Figure 16-1. CPM Enable Register (CPM0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 1
3	P42OPB	PLB4 to OPB Bridge (USB 2.0 device bus)	Class 2
4	UDMA	DMA2PLB4 DMA for USB2.0 Device	Class 2
5	FPU	PPC440 Floating Point Unit	Class 1
6	CPU	440EP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	P32OPB	PLB3 to OPB Bridge	Class 2
9	OPB2P3	OPB to PLB3 Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11		Reserved	
12	DMC	DDR SDRAM Controller	Class 2
13	PLB4	PLB4 Arbiter	Class 2
14	PLB4x3	PLB4 to PLB3 Bridge Controller	Class 2
15	PLB3x4	PLB3 to PLB4 Bridge Controller	Class 2
16	PLB3	PLB3 Arbiter	Class 2
17	NDFC	NAND Flash Controller	Class 2
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1

27	UART2	Universal Asynchronous Receiver/Transmitter 2	Class 1
28	UART3	Universal Asynchronous Receiver/Transmitter 3	Class 1
29	EMC1	Ethernet 1	Class 1
30:31		Reserved	

16.2.2 CPM Force Register (CPM0_FR)

Setting a CPM0_FR bit forces assertion of the CPM_Sleep_N signal to the functional unit. For a class 1 unit, this is equivalent to setting the CPM0_ER bit associated with the unit. For class 2 or class 3 units, CPM_Sleep_N is asserted regardless of the state of the Sleep_Req signal coming from the unit.

Figure 16-2. CPM Force Register (CPM0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 1
3	P42OPB	PLB4 to OPB Bridge (USB 2.0 device bus)	Class 2
4	UDMA	DMA2PLB4 DMA for USB2.0 Device	Class 2
5	FPU	PPC440 Floating Point Unit	Class 1
6	CPU	440EP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	P32OPB	PLB3 to OPB Bridge	Class 2
9	OPB2P3	OPB to PLB3 Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11		Reserved	
12	DMC	DDR SDRAM Controller	Class 2
13	PLB4	PLB4 Arbiter	Class 2
14	PLB4x3	PLB4 to PLB3 Bridge Controller	Class 2
15	PLB3x4	PLB3 to PLB4 Bridge Controller	Class 2
16	PLB3	PLB3 Arbiter	Class 2
17	NDFC	NAND Flash Controller	Class 2
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1

27	UART2	Universal Asynchronous Receiver/Transmitter 2	Class 1
28	UART3	Universal Asynchronous Receiver/Transmitter 3	Class 1
29	EMC1	Ethernet 1	Class 1
30:31		Reserved	

16.2.3 CPM Status Register (CPM0_SR)

The read-only CPM0_SR shows the current state of all CPM_Sleep_N signals.

The following figure describes CPM0_SR bit assignment and CPM class for each PPC440EP functional unit.

Figure 16-3. CPM Force Register (CPM0_SR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 1
3	P42OPB	PLB4 to OPB Bridge (USB 2.0 device bus)	Class 2
4	UDMA	DMA2PLB4 DMA for USB2.0 Device	Class 2
5	FPU	PPC440 Floating Point Unit	Class 1
6	CPU	440EP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	P32OPB	PLB3 to OPB Bridge	Class 2
9	OPB2P3	OPB to PLB3 Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11		Reserved	
12	DMC	DDR SDRAM Controller	Class 2
13	PLB4	PLB4 Arbiter	Class 2
14	PLB4x3	PLB4 to PLB3 Bridge Controller	Class 2
15	PLB3x4	PLB3 to PLB4 Bridge Controller	Class 2
16	PLB3	PLB3 Arbiter	Class 2
17	NDFC	NAND Flash Controller	Class 2
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1

27	UART2	Universal Asynchronous Receiver/Transmitter 2	Class 1
28	UART3	Universal Asynchronous Receiver/Transmitter 3	Class 1
29	EMC1	Ethernet 1	Class 1
30:31		Reserved	

17. Debug Facilities

The debug facilities of the PPC440EP include support for several debug modes for debugging during hardware and software development, as well as debug events that allow developers to control the debug process. Debug registers control these debug modes and debug events. The debug registers may be accessed either through software running on the processor or through the JTAG debug port of the PPC440EP. Access to the debug facilities through the JTAG debug port is typically provided by a debug tool such as the RISCWatch™ development tool. A trace port, which enables the tracing of code running in real time, is also provided.

The JTAG interface is a part of the PPC440 processor. Refer to the *PPC440 Processor User's Manual* for details.

Part IV. PPC440EP Peripheral Functions and Interfaces

18. DDR SDRAM Controller

The double data rate (DDR) SDRAM memory controller supports x8 or greater DDR SDRAMs, consists of a PLB slave interface and a DCR interface, and provides a 32-bit interface to SDRAM memory with optional Error Checking and Correction (ECC). The DDR SDRAM can be relocated in the memory address range, but relocation is generally not needed.

The controller supports page mode operation with bank interleaving always active and can maintain up to eight open pages. The controller supports up to four 256 MB logical banks in limited configurations, providing memory up to 1 GB. Global memory timings, address and bank sizes, and memory addressing modes are programmable. System power can be reduced by placing the DDR SDRAM controller in sleep and/or self-refresh mode.

For 32-bit interface, words are stored in the following order: 1, 3, 0, 2. They are retrieved in the same order.

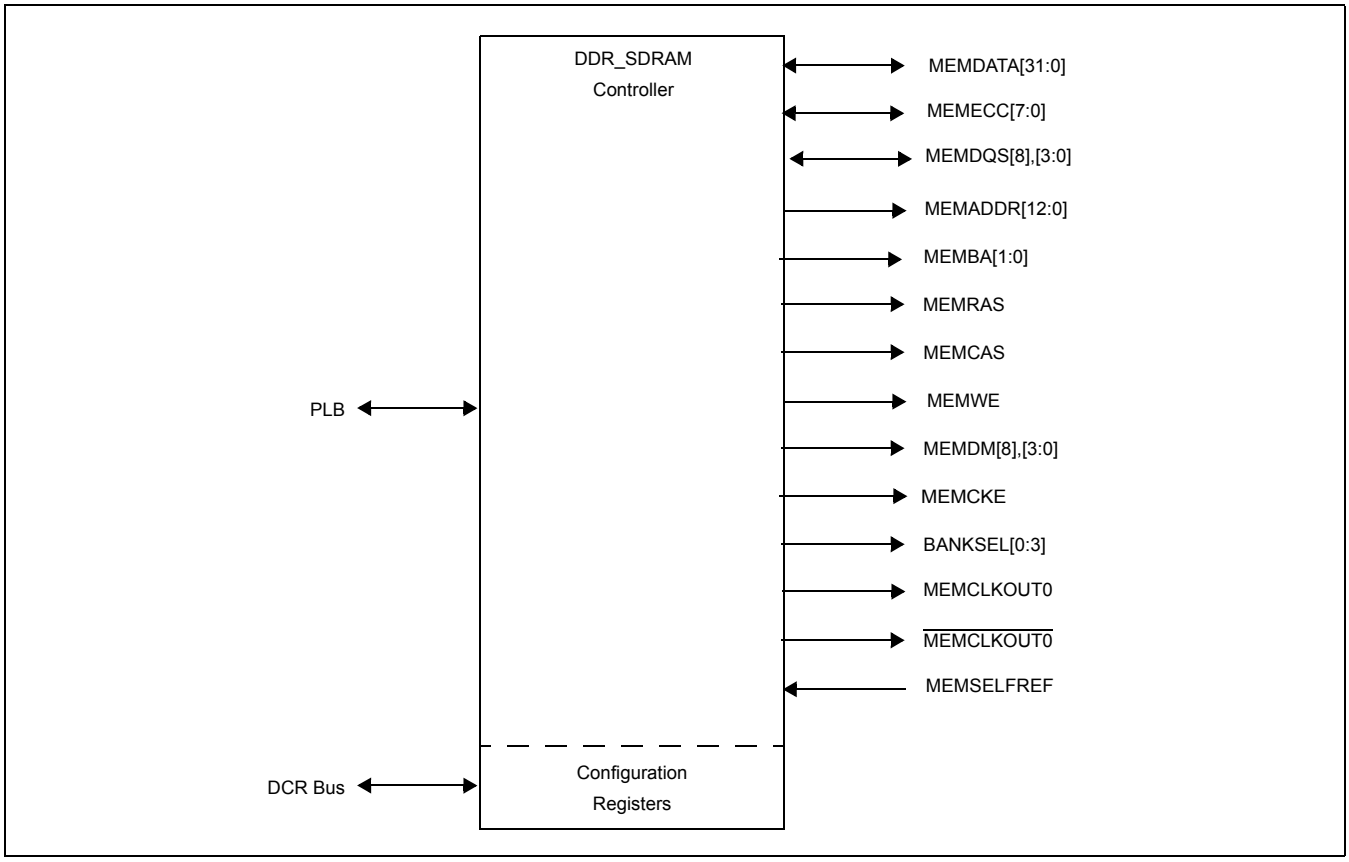
18.1 Interface Signals

In many systems the memory controller can directly interface to DDR SDRAM; however, since each application is different, a detailed timing analysis should always be performed.

The PPC440EP uses PowerPC bit ordering (most significant bit =0); however, the DDR SDRAM address signals (MemAddr12:0 and BA1:0) and external data bus (Memdata[63:0]) use industry standard bit ordering bit 0 is the least significant bit. When a data word is written out to the DDR memory interface the internal most significant bit is 0 and appears as the least significant bit on the DDR interface. The remaining bits of the word follow in order. The main effect of this is that when data internal to the PPC440EP is written out to the DDR interface the bits appear reversed. For example, an internal register value of 0xF753 corresponds to a value of 0xCAEF on the external memory data.

Figure 18-1 illustrates the DDR SDRAM signal I/Os.

Figure 18-1. DDR SDRAM Controller Signals



The usage and signal state after a chip or system reset for each of the DDR SDRAM signals is shown in *Table 18-1*.

Table 18-1. DDR SDRAM Signal Usage and State During/Following Reset

Signal	During Reset State	Following Reset State	Usage
MEMCLKOUT0	n/a	n/a	Differential clock to SDRAMs
$\overline{\text{MEMCLKOUT0}}$	n/a	n/a	Differential clock to SDRAMs
MEMCKE	0s	0s	Clock enable.
MEMBA[1:0]	0s	0s	Bank address. Used to select an internal SDRAM bank in dual- and quad-bank SDRAM devices.
MEMADDR[12:0]	0s	0s	Memory address. See *SDRAM Commands and Operations* on page 347 for details.
$\overline{\text{MEMRAS}}$	1	1	Row Address Strobe.
$\overline{\text{MEMCAS}}$	1	1	Column Address Strobe.
$\overline{\text{MEMWE}}$	1	1	Write Enable.
MEMDATA[31:0]	Unknown	High -Z	Data input/output. MemData0 is the least significant bit.
MEMDQS[8],[3:0]	Unknown	High -Z	Byte lane strobe.
MEMECC[7:0]	Unknown	High -Z	ECC check bits.

Table 18-1. DDR SDRAM Signal Usage and State During/Following Reset (continued)

Signal	During Reset State	Following Reset State	Usage
MEMDM[8],[3:0]	1s	1s	Byte lane mask.
BANKSEL[0:3]	1s	1s	Select up to four external DDR SDRAM banks
MEMSELFREF	n/a	n/a	Hardware self refresh signal

18.2 DDR SDRAM Registers

All DDR SDRAM configuration registers are accessed with the move-to-DCR (**mtdcr**) and move-from-DCR (**mfdcr**) instructions using indirect addressing. In indirect addressing, the **mtdcr** and **mfdcr** instructions access a given DDR SDRAM register via its address offset which is contained in the SDRAM0_CFGADDR register. The data for the specified register is moved to or moved from the SDRAM0_CFGDATA register as described below. The SDRAM0 configuration registers are listed in *Table 18-3*. The offsets for all indirectly addressed DCR registers appear in *Table 4-3* on page 139.

Table 18-2. DDR SDRAM Controller DCR Addresses

Mnemonic	Register	Address	Access
SDRAM0_CFGADDR	DDR SDRAM Controller Address Register	0x0010	R/W
SDRAM0_CFGDATA	DDR SDRAM Controller Data Register	0x0011	R/W

Figure 18-2. SDRAM Configuration Address Register (SDRAM0_CFGADDR)

0:23			
24:31	DCRA	8-bit SDRAM Register Offset Value	This value can range from 0x00 to 0x7F. Its contents are used as the indirect DCR address for accessing a SDRAM register.

Figure 18-3. SDRAM Configuration Data Register (SDRAM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contains the value of the SDRAM register as indicated by the SDRAM0_CFGADDR register.

The following PowerPC code illustrates how to access a DDR SDRAM register by writing the SDRAM0_CFGDATA register and then reading back the written value.

```
li      r3,register_offset      ! address offset of DDR_SDRAM register
lis     r4,<config upper>       ! upper half of configuration data
ori     r4,r4,<config lower>    ! lower half of configuration data
mtdcr   SDRAM0_CFGADDR,r3       ! set offset address
mtdcr   SDRAM0_CFGDATA,r4       ! write configuration data
mfdcr   r5, SDRAM0_CFGDATA      ! read back configuration data
```

Note: Reserved fields in the DDR SDRAM controller configuration registers must not change when the register is written. To modify bit fields within a register, read the register, use a mask to clear or set the target bits, or in the new field value, and then write the result back. A subsequent read of the target register is recommended to ensure that the targeted bit fields were updated properly.

Table 18-3. DDR SDRAM Registers

Mnemonic	Register	Offset	Access	Page
SDRAM0_BESR0	Bus Error Status Register 0	0x0000	R/W	317
SDRAM0_BESR1	Bus Error Status Register 1	0x0008	R/W	318
SDRAM0_BEAR	Bus Error Address Register	0x0010	R	319
SDRAM0_MIRQ	Master Write Interrupt	0x0011	R/W	319
SDRAM0_SLI0	PLB Slave Interface Options	0x0018	R/W	320
SDRAM0_CFG0	DDR SDRAM Controller Options 0	0x0020	R/W	321
SDRAM0_CFG1	DDR SDRAM Controller Options 1	0x0021	R/W	322
SDRAM0_DEVOPT	DDR SDRAM Device Options	0x0022	R/W	323
SDRAM0_MCSTS	DDR SDRAM Memory Controller Status	0x0024	Read only	323
SDRAM0_RTR	Refresh Timer Register	0x0030	R/W	324
SDRAM0_PMIT	Power Management Idle Timer	0x0034	R/W	324
SDRAM0_UABBA	PLB UABus Base Address	0x0038	R/W	324
SDRAM0_B0CR	DDR SDRAM Bank 0 Configuration	0x0040	R/W	325
SDRAM0_B1CR	DDR SDRAM Bank 1 Configuration	0x0044	R/W	325
SDRAM0_B2CR	DDR SDRAM Bank 2 Configuration	0x0048	R/W	325
SDRAM0_B3CR	DDR SDRAM Bank 3 Configuration	0x004C	R/W	325
SDRAM0_TR0	DDR SDRAM Timing Register 0	0x0080	R/W	326
SDRAM0_TR1	DDR SDRAM Timing Register 1	0x0081	R/W	328
SDRAM0_CLKTR	DDR SDRAM Clock Timing Register	0x0082	R/W	334
SDRAM0_WDDCTR	Write Data, DQS, DM Clock Timing Register	0x0083	R/W	337
SDRAM0_DLYCAL	Delay Line Calibration Register	0x0084	R/W	339
SDRAM0_ECCESR	ECC Error Status	0x0098	R/W	340
SDRAM0_CID	Controller ID Register	0x00A4	R	341
SDRAM0_RID	Revision ID Register	0x00A8	R	341

18.2.1 Device Configuration

Table 18-4 lists registers that apply globally to the memory subsystem and are used to configure its operation. The table also shows the effects of reading and writing to each register with SDRAM0_CFG0[DCEN] asserted/deasserted.

Note: Reserved fields in the memory configuration registers must not change when the register is written. Changing reserved fields can cause DDR SDRAM controller malfunction.

Table 18-4. DCRs with Dependencies on SDRAM0_CFG0[DCEN]

Mnemonic	Register	Offset	CFG0[DCEN]	Transaction	Result
SDRAM0_BESR0	Bus Error Status Register 0	0x0000	X (note)	Write	Clear
		0x0004	X	Write	Set
SDRAM0_BESR1	Bus Error Status Register 1	0x0008	X	Write	Clear
		0x000C	X	Write	Set
SDRAM0_BEAR	Bus Error Address Register	0x0010	X	Write	No Change
SDRAM0_MIRQ	Bus Master Interrupt	0x0011	X	Write	Clear
		0x0012	X	Write	Set
SDRAM0_SLI0	DDR SDRAM Slave Interface Options	0x0018	X	Write	Update
SDRAM0_CFG0	DDR SDRAM Options 0	0x0020	X	Write	Update
SDRAM0_CFG1	DDR SDRAM Options 1	0x0021	X	Write	Update
SDRAM0_DEVOPT	DDR SDRAM Device Options	0x0022	0	Write	Update
			1	Write	No change
SDRAM0_MCSTS	Memory Controller Status	0x0024	X	Write	No Change
SDRAM0_RTR	Refresh Timer Register	0x0030	0	Write	Update
			1	Write	No Change
SDRAM0_PMIT	Power Management Idle Timer	0x0034	0	Write	Update
			1	Write	No Change
SDRAM0_UABBA	PLB UABus Base Address	0x0038	0	Write	Update
			1	Write	No Change
SDRAM0_B0CR	DDR SDRAM Bank 0 Configuration	0x0040	0	Write	Update
			1	Write	No Change
SDRAM0_B1CR	DDR SDRAM Bank 1 Configuration	0x0044	0	Write	Update
			1	Write	No Change
SDRAM0_B2CR	DDR SDRAM Bank 2 Configuration	0x0048	0	Write	Update
			1	Write	No Change
SDRAM0_B3CR	DDR SDRAM Bank 3 Configuration	0x004C	0	Write	Update
			1	Write	No Change
SDRAM0_TR0	SDRAM Timing Register 0	0x0080	0	Write	Update
			1	Write	No Change
SDRAM0_TR1	SDRAM Timing Register 1	0x0081	X	Write (note)	Update
SDRAM0_CLKTR	DDR Clock Timing Register	0x0082	X	Write[2]	Update
SDRAM0_WDDCTR	Write Data/DM/DQS Clock Timing Register	0x00x83	X	Write[2]	Update

Table 18-4. DCRs with Dependencies on SDRAM0_CFG0[DCEN] (continued)

Mnemonic	Register	Offset	CFG0[DCEN]	Transaction	Result
SDRAM0_DLYCAL	Delay Line Calibration Register	0x0084	X	Write	Update
SDRAM0_ECCESR	ECC Error Status	0x0098	X	Write	No Change
any	any	any	X	Read	Read Data

X indicates don't care.
Modification of SDRAM0_TR1, SDRAM0_CLKTR, and SDRAM0_WDDCTR after enabling SDRAM0_CFG0[0] is allowed by the logic; however, software must guarantee that the effects of doing so does not affect the operation of the DDR SDRAM controller. The DDR SDRAM should be idle and have no pending requests. Also, when changing SDRAM0_CLKTR, the downstream effects (such as minimum lock time, and re-establishing lock) on the system/chip level PLLs must be accounted for before attempting to access the memory. Modification of SDRAM0_CLKTR after setting SDRAM0_CFG0[0] is not recommended.

18.2.1.1 Initial Configuration Following Power-On-Reset

Note: Configuration of all SDRAM-specific registers must be performed prior to enabling the DDR SDRAM controller as described in *Initialization* on page 342.

All registers must be configured prior to setting SDRAM0_CFG0[DCEN] = 1. Any DCR write to the above SDRAM registers while SDRAM0_CFG0[DCEN] = 1 will complete on the DCR bus; however, no update to target register(s) will be performed.

Write access to SDRAM0_CFG0 is allowed, independent of the SDRAM0_CFG0[DCEN] value. Software must ensure that the DDR SDRAM controller is idle when updating/changing SDRAM0_CFG0. This is required to guarantee that the register update will not affect the operation of any in-progress or preceding PLB-to-memory accesses.

Write access to the DDR SDRAM controller error status registers (SDRAM0_BESR0, SDRAM0_BESR1, SDRAM0_BEAR, SDRAM0_MIRQ, and SDRAM0_ECCESR) are independent of the SDRAM0_CFG0 value. DCR writes to the error status registers are given the highest priority in the event that a DCR error status register write occurs coincident with an error status register update.

DCR reads will complete normally with the contents of the targeted register returned independent of the SDRAM0_CFG0[0] value.

18.2.1.2 Re-Configuration Following Initial Configuration

Re-configuration of the SDRAM-specific registers may be performed at any time following the initial configuration. Before re-configuring, all pending and queued requests targeting the DDR SDRAM controller must be allowed to complete. The following details the specific sequence required:

1. Guarantee all pending and queued memory requests are complete.
2. Disable DDR SDRAM controller - write SDRAM0_CFG0[DCEN]=0.
3. Update SDRAM specific timing registers.
4. Enable DDR SDRAM controller - Write SDRAM0_CFG0[DCEN]=1.

Once enabled, the DDR SDRAM controller will re-initialize the SDRAM devices by performing the necessary auto refresh/MRS/auto refresh sequence as described in *Initialization* on page 342.

18.2.2 PPC440EP PLB3 and PLB4 Masters

For PLB3 and PLB4 Master assignments, refer to *PLB Master Priority Assignment* on page 64.

18.2.3 Bus Error Syndrome Register 0 (SDRAM0_BESR0)

The SDRAM0_BESR0 tracks errors encountered during PLB Master[0:3] accesses to the DDR SDRAM. Bits in SDRAM0_BESR0 are cleared by writing a 32-bit value to SDRAM0_BESR0 with a 1 in any bit position that is to be cleared and 0 in all other bit positions.

Figure 18-4. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	M0ET	Master 0 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
3	M0RW	Master 0 Read/Write Status 0 Write Error 1 Read Error	
4	M0FL	Master 0 Field Lock 0 BESR0 Unlocked 1 BESR0 Locked	
5	M0AL	Master 0 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
6:8	M1ET	Master 1 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
9	M1RW	Master 1 Read/Write Status 0 Reserved 1 Read Error	
10	M1FL	Master 1 Field Lock 0 BESR1 Unlocked 1 BESR1 Locked	
11	M1AL	Master 1 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
12:14	M2ET	Master 2 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
15	M2RW	Master 2 Read/Write Status 0 Write Error 1 Reserved	
16	M2FL	Master 2 Field Lock 0 BESR2 Unlocked 1 BESR2 Locked	
17	M2AL	Master 2 Address Lock 0 BEAR Not Locked 1 BEAR Locked	

18:20	M3ET	Master 3 Error Type 000 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
21	M3RW	Master 3 Read/Write Status 0 Write Error 1 Read Error	
22	M3FL	Master 3 Field Lock 0 BESR3 Unlocked 1 BESR3 Locked	
23	M3AL	Master Address Lock 0 BEAR Not Locked 1 BEAR Locked	
24:31		Reserved	

18.2.4 Bus Error Syndrome Register 1 (SDRAM0_BESR1)

The SDRAM0_BESR1 tracks errors encountered during PLB Master [5:7] accesses to system memory (see *Table 18-5*). This register can be accessed at offset 0x08 for the purpose of reading or clearing error status bits. Offset 0x0C is available for test purposes only and allows error bits to be set. A write access to 0x08 should provide a 32-bit mask, where each 1 in the mask will clear the corresponding bit in the error status register. A write access to 0x0C should provide a 32-bit mask, where each 1 in the mask will set the corresponding bit in the error status register.

Figure 18-5. Bus Error Syndrome Register 1 (SDRAM0_BESR1)

0:2	M4ET	Master 4 Error Type 000 No Error 001 Reserved 010 ECC Error 100 Reserved	
3	M4RW	Master 4 Read/Write 0 Write Error 1 Read Error	
4	M4FL	Master 4 Field Lock 0 BESR4 Unlocked 1 BESR4 Locked	
5	M4AL	Master 4 Address Lock 0 BEAR Not Locked 1 BEAR Locked by	
6:8	M5ET	Master 5 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
9	M5RW	Master 5 Read/Write 0 Write Error 1 Read Error	
10	M5FL	Master 5 Field Lock 0 BESR5 Unlocked 1 BESR5 Locked	

11	M5AL	Master 5 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
12:14	M6ET	Master 6 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
15	M6RW	Master 6 Read/Write 0 Write Error 1 Read Error	
16	M6FL	Master 6 Field Lock 0 BESR6 Unlocked 1 BESR6 Locked	
17	M6AL	Master 6 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
18:20	M7ET	Master 7 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	
21	M7RW	Master 7 Read/Write 0 Write Error 1 Read Error	
22	M7FL	Master 7 Field Lock 0 BESR7 Unlocked 1 BESR7 Locked	
23	M7AL	Master 7 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
24:31		Reserved	

18.2.5 Master Bus Error Address Register (SDRAM0_BEAR)

The SDRAM Bus Error Address Register (SDRAM0_BEAR) is a 32-bit register containing the address of the access where an uncorrectable ECC error occurred. If the master that initiated the transfer requested error locking, and the SDRAM0_BEAR is not already locked, the contents of SDRAM0_BEAR are locked until the lock bit in one of the SDRAM Bus Error Registers (SDRAM0_BESR0 or SDRAM0_BESR1) is cleared. The content of the SDRAM0_BEAR is accessed indirectly through the SDRAM0_CFGADDR and SDRAM0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

Figure 18-6. Bus Error Address Register (SDRAM0_BEAR)

0:31	BEAR	Address of Bus Error	

18.2.6 Master Write Interrupt (SDRAM0_MIRQ)

The SDRAM0_MIRQ register tracks errors encountered during Master [0:3 and 5:7] write accesses to system memory (see *Table 18-5*). This register can be accessed at offset 0x11 for the purpose of reading or clearing error status bits. A write access to 0x11 provides a 32-bit mask, where each 1 in the mask clears the corresponding bit

in the error status register. Offset 0x12 is available for test purposes only and allows error bits to be set. A write access to 0x12 provides a 32-bit mask, where each 1 in the mask sets the corresponding bit in the error status register.

This register is set when a posted write error occurs for the associated PLB master. The corresponding DDR SDRAM output will assert and remain asserted for the associated master's transfer until the corresponding register bit is cleared by a DCR write to offset 0x11.

Figure 18-7. Master Write Interrupt (SDRAM0_MIRQ)

0	M0IR	Master 0 Write Interrupt 0 No Error 1 Write Error	
1	M1IR	Master 1 Write Interrupt 0 No Error 1 Write Error	
2	M2IR	Master 2 Write Interrupt 0 No Error 1 Write Error	
3	M3IR	Master 3 Write Interrupt 0 No Error 1 Write Error	
4	M4IR	Master 4 Write Interrupt 0 No Error 1 Write Error	
5	M5IR	Master 5 Write Interrupt 0 No Error 1 Write Error	
6	M6IR	Master 6 Write Interrupt 0 No Error 1 Write Error	
7	M7IR	Master 7 Write Interrupt 0 No Error 1 Write Error	
8:31		Reserved	

18.2.7 PLB Slave Interface Options (SDRAM0_SLIO)

This register enables the configuration of the PLB Slave-specific interface options and determines how the interface responds on the PLB, based on the status of the address queue and buffer queues.

See *Section 18.5 DDR Slave Interface Options* for a detailed description of the behavior of each of these options.

Figure 18-8. PLB Slave Interface Options (SDRAM0_SLIO)			
0	RDRE	PLB Slave Read Rearbitrate Enable 0 PLB slave read rearbitrate disabled 1 PLB slave read rearbitrate enabled	
1	WRRE	PLB Slave Write Rearbitrate Enable 0 PLB slave write rearbitrate disabled 1 PLB slave write rearbitrate enabled	
2	RARW	PLB Read Around Write Enable 0 PLB read around write disabled 1 PLB read around write enabled	
3:31		Reserved	

18.2.8 Memory Controller Options 0 (SDRAM0_CFG0)

This register enables the configuration of the DDR SDRAM controller specific options. See the associated DDR SDRAM sections of this document for specific details and resultant behavior.

All application-specific options selected using SDRAM0_CFG0 should be configured prior to enabling the DDR SDRAM controller.

Figure 18-9. Memory Controller Options 0 (SDRAM0_CFG0)			
0	DCEN	DDR SDRAM Controller Enable 0 DDR SDRAM disabled 1 DDR SDRAM enabled	When SDRAM0_CFG0[DCEN=1], DDR SDRAM is initialized using the power-on sequence and subsequently available for access. All DDR SDRAM controller configuration registers should be initialized and valid when SDRAM0_CFG0[DCEN=1].
1		Reserved	
2:3	MCHK	Memory Data Error Checking 00 None 01 Reserved 10 ECC generation only (no checking or correction) 11 ECC checking and correction	
4	RDEN	Registered DIMM Enable 0 Disable 1 Enable	
5	PMU	Page Management Unit 0 Page management unit enabled 1 Page management unit disabled	When SDRAM0_CFG[PMU=0], up to 8 open pages are maintained for subsequent accesses. When SDRAM0_CFG[PMU=1], the accessed page is opened for a given access, remains open for the duration of the access (allowing page hits within PLB sequential bursts) and is precharged (using read/write with autoprecharge command) upon completion of the access.
6	DMWD	DDR SDRAM Width 0 32-bit 1 64-bit	**Note:** This bit should be set to 0.
7		Reserved	

8:9	UIOS	Unused I/O State 00 Active (High-Z on Read, Driven on Write) - switching DQS 01 Active (High-Z on Read, Driven on Write) -static values 10 Static (Always Driven) - static values 11 High-Z	This field can be used to control the state of the unused I/O for various configurations through the corresponding I/O driver data and enable signals. Unused I/O are defined as the I/O associated with a function that is not enabled. Example: CB[0:7], DM[8], and DQS[8] would be unused I/O when ECC is not enabled. Likewise, DATA[32:63], DM[4:7], and DQS[4:7] would be unused I/O in 32-bit mode. All unused I/O receivers will be gated off.
10	PDP	Page Deallocation Policy 0 Pseudo Least Recently Allocated 1 Least Recently Used	This field selects the page deallocation policy when page mode is enabled. Page deallocation occurs when the PMU has 8 open pages and an access does not target a page or bank address associated with one of those open pages. A PMU entry must be deallocated or replaced (and the corresponding page closed). With PMU_DP=0, the PMU entry that has been open longest will be deallocated. With PMU_DP=1, the page that was least recently used (least recently accessed) will be deallocated. **Note:** This distinction is only relevant for memory subsystems employing more than two physical banks (chip selects) of memory.
11:31		Reserved	

SDRAM0_CFG0[UIOS] enables the user to configure the behavior of the unused I/O, if any, at the chip level. Based on the selected UIOS setting, the DDR SDRAM output connections to the chip level I/O (Z, RG, A, and TS) inputs are controlled to provide the described behavior.

The one potential unused I/O grouping is: CB[0:7], DM[8], and DQS[8] for ECC disabled.

The behavior of this I/O grouping, based on the selected UIOS setting, is as follows:

Unused IO	UIOS			
	00	01	10	11
	Write/ Read	Write/ Read	Write/ Read	Write/ Read
DM[8]	1 / 1	1 / 1	1/1	1/1
DQS[8]	T / Hi-Z[1]	0 / Hi-Z	0/0	Hi-Z/Hi-Z
ECC[0:7]	0s / Hi-Z	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z

18.2.9 Memory Controller Options 1 (SDRAM0_CFG1)

SDRAM0_CFG1 enables the configuration of the DDR SDRAM controller power management and self-refresh features. See the associated DDR SDRAM sections of this document for specific details and resultant behavior.

Figure 18-10. Memory Controller Options 1 (SDRAM0_CFG1)			
0	SRE	Self-Refresh Entry 0 Exit 1 Enter	This bit may be set by software (using DCR write). Once set, the SDRAM is maintained in self-refresh mode until this bit is reset by software. The DDR_SDRAM controller responds to this bit independent of the state of SDRAM0_CFG0[DCEN].
1	PMEN	Power Management Enable 0 Power management disabled 1 Power management enabled	See *Power Management* on page 360 for specific details.
2:31		Reserved	

18.2.10 DDR SDRAM Device Options (SDRAM0_DEVOPT)

SDRAM0_DEVOPT allows for the configuration of the defined DDR SDRAM device-specific options. Users should consult the vendor-specific DDR SDRAM device specification for a detailed description of these options.

Figure 18-11. DDR_SDRAM Device Options (SDRAM0_DEVOPT)			
0	DLL	DDR_SDRAM Device DLL Disable 0 Device DLL enabled 1 Device DLL disabled	Enable DLL for normal operation. The DLL setting controls the state of address bit during the initialization of the Extended Mode Register.
1	DS	DDR_SDRAM Device I/O Drive Strength 0 Drive strength is normal 1 Drive strength is weak	Clear DS for normal operation. The DS setting controls the state of address bit MemAddr1 during the initialization of the Extended Mode Register.
2:31		Reserved	

18.2.11 Memory Controller Status (SDRAM0_MCSTS)

SDRAM0_MCSTS provides real-time status to the system software on the operation of the DDR SDRAM controller. The information returned indicates the DDR SDRAM status at the time of the DCR read.

Figure 18-12. Memory Controller Status (SDRAM0_MCSTS)			
0	MRSC	MRS Command Complete 0 MRS Not Complete 1 MRS Complete	This bit will be set to 1 once the DDR SDRAM controller has successfully completed the Mode Register Set Command, which results from setting DC_EN in SDRAM0_CFG0. Clearing DC_EN will clear this bit in the following cycle.
1	SRMS	Self-Refresh Mode Status 0 SDRAM is not in Self-Refresh Mode 1 SDRAM is in Self-Refresh Mode	This bit will be set upon the successful completion of Self-Refresh Mode entry, that is, the SDRAM device has been placed in Self-Refresh Mode. This bit will be cleared when Self-Refresh Mode has been exited. This bit applies to the software-initiated Self-Refresh Mode using SDRAM0_CFG0[SRE].

2	CIS	Core Idle Status 0 Busy 1 Idle	Indicates that there are no current or pending queued read/write accesses.
3:31		Reserved	

18.2.12 Refresh Timer Register (SDRAM0_RTR)

The 16-bit field of RTR determines the memory refresh rate for the DDR SDRAM. The internal counter runs at the controller clock frequency; thus, at 100/133 MHz, a value of 0x05F0/0x07E0 produces a refresh interval of 15.20µs (1520 times 10ns equals 15.20µs/2016 times 7.5ns equals 15.12µs). This register must be programmed to accommodate the DDR SDRAM device-specific refresh interval for the target operating frequency.

Note: Bit 0 is the Most Significant Bit and bit 15 is the Least Significant Bit.

Figure 18-13. Refresh Timing Register (SDRAM0_RTR)

0:1	0	Non-programmable Bits 0:1 of the Refresh Interval	Hard coded to zero.
2:12	RINT	Programmable Bits 2:12 of the Refresh Interval	
13:15	0	Non-programmable Bits 13:15 of the Refresh Interval	Hard coded to zero.
16:31		Reserved	

18.2.13 Power Management Idle Timer (SDRAM0_PMIT)

The 10-bit field (only five bits are user programmable) of the SDRAM0_PMIT determines the number of controller clock cycles that the DDR SDRAM controller must be idle before making a sleep request to the Clock and Power Management controller (CPM), when power management is enabled using SDRAM0_CFG1[PMEN]. The reset value initializes this timer to 32 clock cycles.

Note 1: The minimum granularity of the idle timer is 32 clock cycles.

Note 2: Bit 0 is the most significant bit and bit 4 is the least significant bit.

Figure 18-14. Power Management Idle Timer (SDRAM0_PMIT)

0:4	PMC	Power Management Count	Programmable
5:31		Reserved	

18.2.14 PLB UABus Base Address (SDRAM0_UABBA)

The UABBA must be configured in conjunction with the SDRAM0_BnCR registers to define the base address of each memory bank.

To maintain default system memory map, the SDRAM0_UABBA[28:31] should stay at zero. Contact PowePC support if there is a need to change it.

Figure 18-15. PLB UABus Base Address (SDRAM0_UABBA)			
0:27		Reserved	
28:31	UBBA	PLB UABus Base Address	Defines the 4 GB aligned region in which the physical memory is located.

18.2.15 Memory 0–3 Configuration (SDRAM0_B0CR:SDRAM0_B3CR)

The DDR SDRAM memory controller provides support for four logical banks. The SDRAM0_ BnCR registers configure and enable memory in each respective logical bank. Only logical banks are enabled, SDRAM0_BnCR[BE]=1, are initialized when the DDR SDRAM controller is enabled, SDRAM0_CFG[DCE]=1. Since the SDRAM0_BnCR registers cannot be modified when SDRAM0_CFG[DCE]=1, adding or removing memory banks requires that the DDR SDRAM controller be disabled and then reinitialized.

Note: The base address must be aligned on a boundary that matches the size of the region as defined by the SDSZ field. For example, a 16-MB region must begin on an address that is divisible by 16 MB.

Figure 18-16. Memory 0-3 Configuration (SDRAM0_B0CR:SDRAM0_B3CR)			
0:8	SDBA	Base Address	
9:11		Reserved	
12:14	SDSZ	Size 000 Reserved 001 8 MB 010 16 MB 011 32 MB 100 64 MB 101 128 MB 110 256 MB 111 reserved	
15		Reserved	
16:18	SDAM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 1xx Reserved	See *Table 18-5 DDR SDRAM Addressing Modes* for details.
19:30		Reserved	
31	SDBE	Memory Bank Enable	

Table 18-5. DDR SDRAM Addressing Modes

Addressing Mode	DDR SDRAM Memory Organization[1]
Mode 1	11 x 8 - 4 Bank 12 x 8 - 4 Bank 13 x 8 - 4 Bank
Mode 2	11 x 9 - 4 Bank 12 x 9 - 4 Bank 13 x 9 - 4 Bank
Mode 3	11 x10 - 4 Bank 12 x10 - 4 Bank 13 x10 - 4 Bank
Mode 4	11 x11 - 4 Bank 12 x11 - 4 Bank 13 x11 - 4 Bank
Note: Row x Column - Number of physical banks contained in the DDR SDRAM module.	

18.2.16 SDRAM Timing Register 0 (SDRAM0_TR0)

This register must be programmed to configure the DDR SDRAM device-specific timing parameters, expressed in units of clock cycles, as defined by the selected DDR SDRAM device specification.

Settings below the minimums defined by the device specifications and the minimums, in terms of clock cycles, calculated from the formulas provided herein are not supported.

Figure 18-17. SDRAM Timing Register 0 (SDRAM0_TR0)

0	SDWR	DDR SDRAM Write Recovery 0 2 CLK 1 3 CLK	T_{wr}
1	SDWD	DDR SDRAM Write to Write delay when crossing a chip select boundary 0 0 CLK 1 1 CLK	This field configures the delay between back to back write cycles that cross chip select boundaries. A setting of 0 enables seamless writes when crossing chip select boundaries, while a setting of 1 will insert a single clock cycle delay between back-to-back writes when crossing a chip select boundary.
2:6		Reserved	
7:8	SDCL	DDR SDRAM CAS_ Latency 00 Reserved 01 2 CLK 10 2.5 CLK 11 3 CLK	T_{aa} This field indicates the DDR SDRAM device CAS latency (independent of SDRAM0_CFG0[R_DIMM_EN]) and is used directly during the SDRAM device mode set command to configure the DDR SDRAM. See the section on "Registered DIMM Support" for the supported configurations and CAS_ Latency settings. **Note:** Setting SD_CL to 2.5 CLK generally requires that SDRAM0_TR1[RDCD] be set to 1, SDRAM0_TR1[RDSS] be set to T2 Sample, and SDRAM0_TR1[RDSL] be set to Stage 3.

9:11		Reserved	
12:13	SDPA	DDR SDRAM CBR Precharge Command to next Activate Command minimum 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rp}
14:15	SDCP	DDR SDRAM Read/Write Command to Precharge Command 00 2 CLK 01 3 CLK 10 4 CLK 11 Reserved	T_{ras} - T_{rcd} The minimum value is calculated by first determining the values of T_{ras} and T_{rcd} in units of clock cycles (divide the respective device-specific AC timing specifications by the clock cycle time and round up to the nearest whole clock). Then apply the formula provided. SDCP settings that violate this rule (that is, settings less than the minimum calculated above for the device-specific timing parameters), independent of the programmed value of SDRD, are not supported.
16:17	SDLD	DDR SDRAM Command Leadoff 00 1 CLK 01 2 CLK 10 Reserved 11 Reserved	
18:23		Reserved	
24	TWTR	DDR SDRAM Write-To-Read Command 0 0 Selects Twtr of 1 clock 1 1 Selects Twtr of 2 clocks	Defaults to 0. Defines Write-To-Read.
25:26	RFTA	DDR SDRAM Refresh-To-Activate command adder Add the specified number of clocks to the value programmed in SDRAM0_TR1[SDRA] 00 0 01 1 10 2 11 Reserved	Defines an adder to the value selected in SDRAM0_TR1[SDRA]. It should set to a non-zero value only if SDRAM_TR1[SDRA] = 0b111. **Example:** Setting SDRAM0_TR1[SDRA] = 0b111 selects 13 clock cycles. Additionally, setting SDRAM0_TR1[RFTA] = 0b01 provides a total of 14 clocks.
27:29	SDRA	DDR SDRAM CBR Refresh Command to next Activate Command minimum 000 6 CLK 001 7 CLK 010 8 CLK 011 9 CLK 100 10 CLK 101 11 CLK 110 12 CLK 111 13 CLK	T_{rfc}
30:31	SDRD	DDR SDRAM RAS to CAS Delay 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rcd}

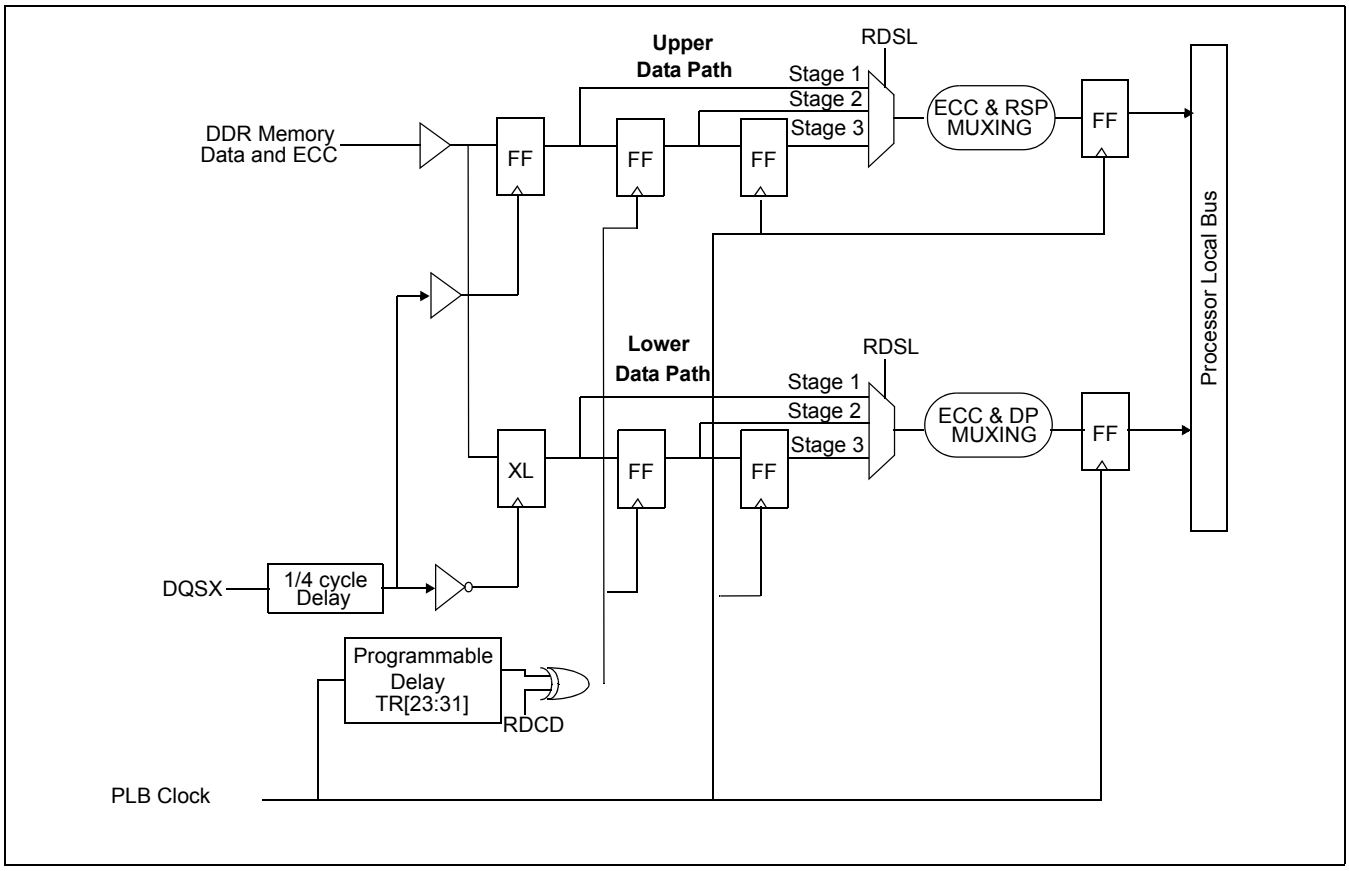
18.2.17 SDRAM Timing Register 1 (SDRAM0_TR1)

SDRAM0_TR1 controls the various aspects of the DDR SDRAM read data path.

DDR memory is source synchronous in that a read data strobe/clock (DQS) is propagated with the DDR read data. Read data is initially sampled in the DDR DQS clock domain and must be synchronized with the controller internal clock domain for transfer on the PLB Bus. Because the read Data and DQS are propagated from the DDR memory to the DDR SDRAM controller and derived from the DDR Memory clock, which is generated and propagated from the controller to the DDR memory, there is the potential for a varying (across different implementations) range of read Data/DQS arrival times relative to the fixed internal controller clock domains. Note that the read Data/DQS arrival is dependent on the DDR memory clock, and the memory clock is affected by the configuration of SDRAM0_CLKTR. The effects of such configuration must be understood and accounted for in determining the appropriate SDRAM0_TR1 settings.

As such, the DDR SDRAM controller read data path is structured to enable, in conjunction with SDRAM0_TR1, configurable support for a wide range of DDR read Data and read DQS arrival times with respect to the controller internal clock domains. The discussion that follows refers to determinations that should be made based on a complete and thorough memory subsystem timing budget analysis. The variations and options discussed are assumed to be static, once determined, for the specific application. Dynamically varying SDRAM0_TR1 during in-progress read accesses is not supported and doing so may yield unpredictable behavior and/or DDR SDRAM malfunction.

Figure 18-18. DDR SDRAM Controller Read Data Path Structure



The DDR SDRAM read data path structure is composed of three stages of registers for both the high order (rising edge of DQS sample) data and ECC check bits and the low order (falling edge of DQS sample) data and ECC check bits. The logic elements are referred to as the "upper" data path and the "lower" data path in the following sections. Stages 1, 2, and 3 are placed physically adjacent to each other and to the IO cell for each associated bit of the read data bus.

Stage 1 consists of a flip-flop (FF) for the upper data path and a transparent latch (XL) for the lower data path. The upper data is captured into the FF with the rising edge of read DQS, while the lower data flushes through the XL when the read DQS is high and captured and held with the falling edge of read DQS.

Stage 2 consists of a FF for both the upper and lower data path. Stage 2 has programmable delay capability in one coarse delay of 1/2 cycle (SDRAM0_TR1[20]) as well as finer delay capability using the Read Clock Delay Tuning Bits (SDRAM0_TR1[23:31]). The coarse half cycle delay is provided primarily for CAS Latency 2.5 support and enables read Data to be captured into Stage 2 while compensating for the 1/2 cycle latency adder and allowing proper synchronization with the internal DDR SDRAM clock domain. The delay effect of the Read Clock Delay Tuning Bits (SDRAM0_TR1[23:31]) is dependent on many factors, including delay line physical placement, ASIC process variation, and voltage/temperature variations. The Delay Line Calibration Register (SDRAM0_DLYCAL) may be used to calibrate the tuning bits.

Stage 3 consists of a FF for both the upper and the lower data path. Stage 3 is clocked by the main DDR SDRAM PLB clock, which also clocks the DDR SDRAM internal registers, read and write data buffers, as well as the PLB interface logic. There is no programmable delay capability for this clock.

The PLB Read Sample Point register is the final stage of relevance in the read data path structure. It is at this stage that the final relative clock domain synchronization occurs and that the DDR memory read data is sampled and driven back to the PLB read data bus in response to a given PLB Master read request.

18.2.17.1 SDRAM0_TR1 [RDSL]/[RDCD]/[RDCT]

Within SDRAM0_TR1, it is the SDRAM0_TR1[RDSL] setting that determines which of the three Stages will be the source of the data steered to the input of the Read Sample Point register. The critical path within the memory subsystem timing budget for a DDR application will be the lower read data; that is, the read data launched and propagated with the falling edge of read DQS, captured and synchronized to the internal clock domain(s).

In order to use Stage 1, lower read Data/DQS must arrive early enough so that the data can flush through the XL downstream and meet the setup and hold time requirements at the PLB Read Sample Point register. Note that the setup/hold time requirements at the PLB Read Sample Point register will differ for ECC enabled vs. ECC disabled.

If the lower read Data/DQS arrival is such that the downstream setup time requirements cannot be met at the PLB Read Sample Point register along the Stage 1 path, Stage 2 may be used. This is provided that the lower read Data arrival is such that the setup/hold time requirements at Stage 2 can be met. In the event that the lower read Data/DQS arrival is such that the read Data does not meet the setup time requirements at Stage 2, the Stage 2 sample clock may be delayed so that the setup/hold time requirements can be met to enable the Data to be captured.

Sample Stage 3 is provided as a final synchronization option for circumstances (read Data/DQS arrival) where the Stage 2 sample clock is delayed to the point where the register to register timing path between Stage 2 and the downstream PLBS Read Sample Point register can no longer be met. Such a circumstance may, for example, be running a CAS Latency 2.5 device with the Stage 2 read sample clock delay by 1/2 cycle or more (through the combination of SDRAM0_TR1[RDCD] and SDRAM0_TR1[RDCT] settings) with ECC enabled at a high frequency. If necessary, data can be captured directly into Stage 3 and then passed to the Read Sample Point register in the following clock cycle.

18.2.17.2 SDRAM0_TR1 [RDSS]

The SDRAM0_TR1[RDSS] setting determines the clock cycle, relative to the DDR read command, in which the internal logic should sample the read Data at the PLB Read Sample Point register. The relative cycles for the defined sample points are CAS Latency dependent. Both unbuffered (R_DIMM_EN=0) and registered (R_DIMM_EN=1) interfaces (such as Registered DIMMs) are depicted, as the configured sample point (and the arrival of the DDR read Data) differs based on these two types of memory interfaces. RDSS must be configured, in conjunction with SDRAM0_TR1[RDSL], to select the source and sample point for the incoming read Data.

18.2.17.3 CAS Latency 2 Devices

The DDR memory devices (described in *Figure 18-19*) launch read data nominally coincident with the second rising edge of its Memory Clock following the cycle in which the read command was sampled by the device.

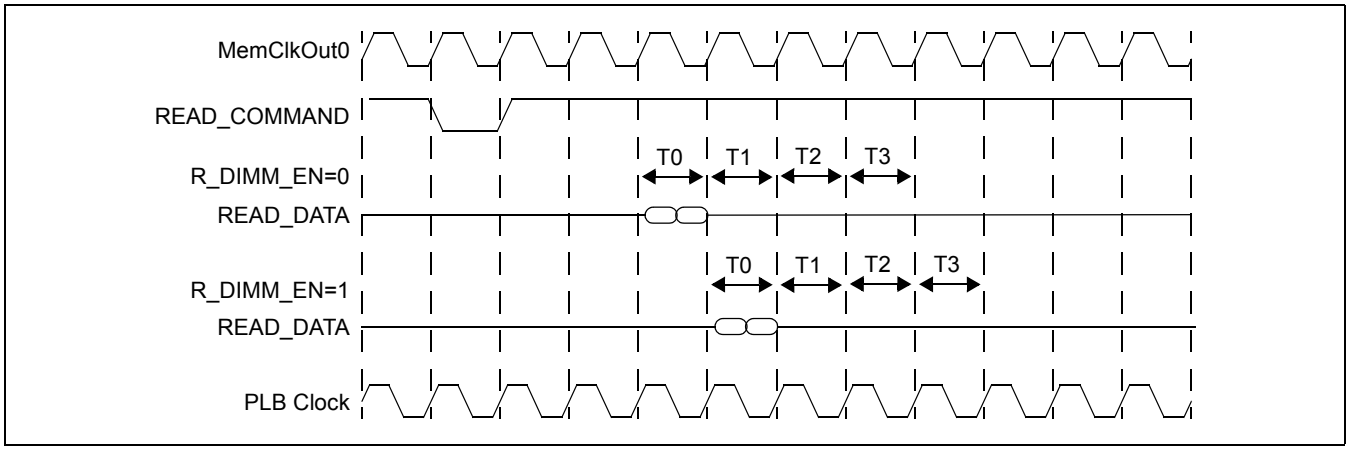
A T0 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T0. This is the end of the first cycle in which data, both upper and lower, was launched from the DDR device. A T0 Sample should only be used in conjunction with Read Sample Stage 1 and requires that the timing budget support the use of Stage 1. A T0 Sample should never be used if SDRAM0_TR1[RDSL] is configured for Stages 2 or 3.

A T1 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T1. This is the end of the second cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T1 Sample will generally be used in conjunction with Read Sample Stage 2 and requires that the timing budget support the use of Stage 2.

A T2 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T2. This is the end of the third cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T2 Sample will generally be used in conjunction with Read Sample Stage 3 and requires that the timing budget support the use of Stage 3. It is expected that CAS Latency 2 applications would never need to use a T2 Sample or Stage 3.

A T3 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T3. This is the end of the fourth cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. It is expected that CAS Latency 2 applications would never need to use a T3 Sample and, as such, should not be configured to do so.

Figure 18-19. Read Sample Cycle: CL = 2



18.2.17.4 CAS Latency 2.5 Devices

For CAS Latency 2.5 devices (described in *Figure 18-20*), the DDR device launches read data nominally coincident with the third *falling* edge of it's Memory Clock following the cycle in which the read command was sampled by the device. Note that a half cycle delay (SDRAM0_TR1[RDCD]=1) on MRDP_CLK enables read Data capture into Stage 2 to compensate for the 1/2 latency adder (associated with CL=2.5 devices) and the resultant mismatch with the internal read sample clock. The half cycle delay also allows proper synchronization with the internal DDR SDRAM clock domain.

A T0 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T0. This is the end of the first *half* cycle from which data was launched from the DDR device. A T0 Sample should only be used in conjunction with Read Sample Stage 1 and requires that the timing budget support the use of Stage 1. A T0 Sample should never be used if SDRAM0_TR1[RDSL] is configured for Stages 2 or 3. It is expected that the use of T0 Sample for CL=2.5 applications would be unlikely given the half cycle latency adder of the devices. That said, it may be possible, through SDRAM0_CLKTR configuration, to recover the half cycle latency adder provided that the memory subsystem timing budget supports such configuration, thereby enabling a T0 Sample. In such a configuration, the appropriate setting for SDRAM0_TR1[RDCD] must be determined.

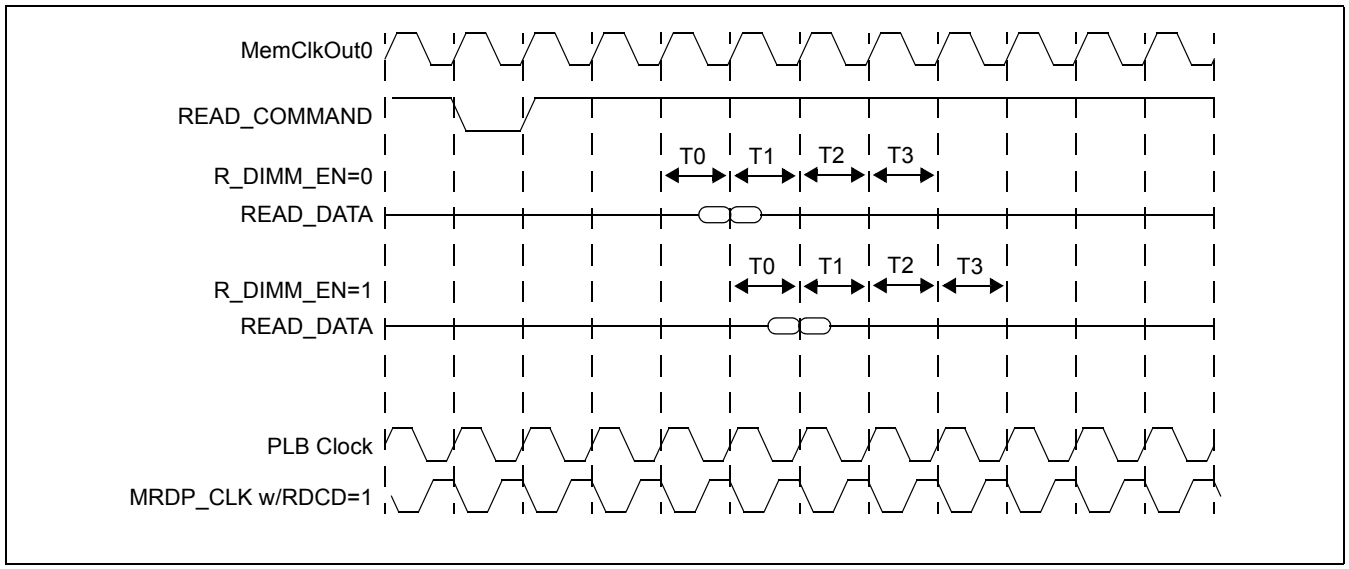
A T1 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T1. This is the end of the third *half* cycle, relative to the *half* cycle from which data was launched from the DDR device. A T1 Sample will generally be used in conjunction with Read Sample Stage 2 and requires that the timing budget support the use of Stage 2.

A T2 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T2. This is the end of the fifth *half* cycle, relative to the *half* cycle from which data was launched from the DDR device. A T2 Sample will generally be used in conjunction with a Read Sample Stage 3 and requires that the timing budget support the use of Stage 3 if configured as such. Depending of the read Data/DQS arrival, if needed, it is possible to configure a T2 Sample in conjunction with Read Sample Stage 2. Such a configuration requires that the timing budget supports/requires the use of Stage 2 and the appropriate setting SDRAM0_TR1[RDCD] must be determined.

A T3 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T3. This is the end of the seventh *half* cycle, relative to the *half* cycle from which data was launched from the DDR device. It is expected that CAS Latency 2.5 applications would never use a T3

Sample. Depending of the read Data/DQS arrival, if needed, it is possible to configure a T3 Sample in conjunction with Read Sample Stage 3. Such a configuration requires that the timing budget support/require the use of Stage 3 and the appropriate setting SDRAM0_TR1[RDCD] must be determined.

Figure 18-20. Read Sample Cycle: CL = 2.5



18.2.17.5 CAS Latency 3 Devices

For CAS Latency 3 devices (described in *Figure 18-21*), the DDR device launches read data nominally coincident with the third rising edge of its Memory Clock following the cycle in which the read command was sampled by the device.

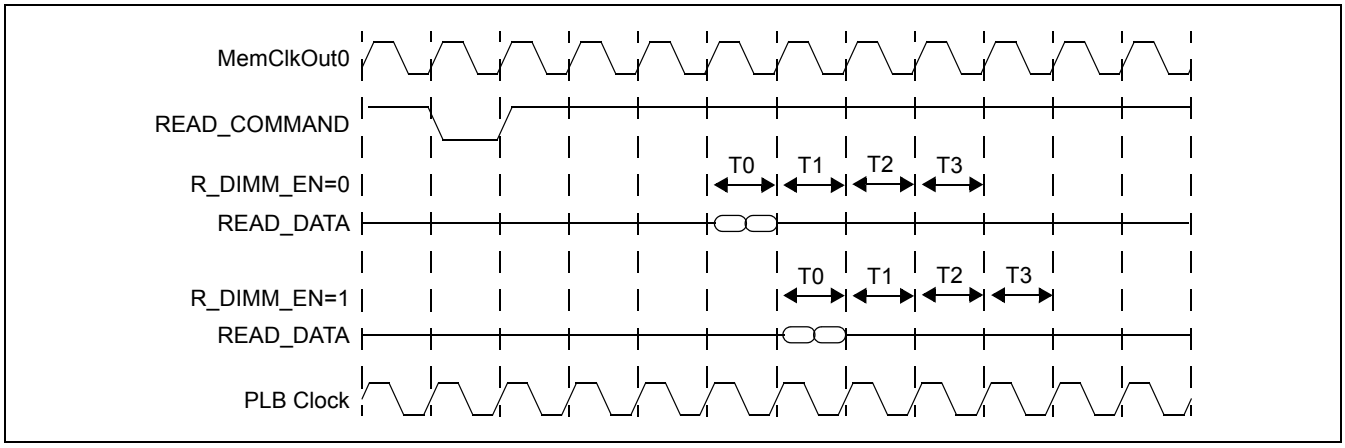
A T0 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T0. This is the end of the first cycle in which data, both upper and lower, was launched from the DDR device. A T0 Sample should only be used in conjunction with Read Sample Stage 1 and requires that the timing budget support the use of Stage 1. A T0 Sample should never be used if SDRAM0_TR1[RDSL] is configured for Stages 2 or 3.

A T1 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T1. This is the end of the second cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T1 Sample will generally be used in conjunction with Read Sample Stage 2 and requires that the timing budget support the use of Stage 2.

A T2 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T2. This is the end of the third cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T2 Sample will generally be used in conjunction with Read Sample Stage 3 and requires that the timing budget support the use of Stage 3. It is expected that CAS Latency 3 applications would never need to use a T2 Sample or Stage 3.

A T3 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T3. This is the end of the fourth cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. It is expected that CAS Latency 3 applications would never use a T3 Sample and, as such, should not be configured to do so.

Figure 18-21. Read Sample Cycle: CL = 3



18.2.17.6 SDRAM0_TR1 Recommendations/Considerations

Within the logic, the SDRAM0_TR1[RDSS] configuration setting is independent of the SDRAM0_TR1[RDSL] setting. As such, the two can be configured independently and in such a way as to prevent successful capture of the DDR read data. Care must be taken to understand the memory subsystem clocking, timing budget, and delays before modifying SDRAM0_TR1.

In general, it is recommended that the SDRAM0_TR1[RDSS] setting equal the SDRAM0_TR1[RDSL] setting, unless the memory subsystem timing budget dictates otherwise. It is also recommended that the default SDRAM0_TR1 settings be used for CAS Latency 2 and 3 devices and the SDRAM0_TR1[RDCD] be set to 1 for CAS Latency 2.5 devices and SDRAM0_TR1[RDCT] be configured if appropriate or necessary.

The combination of Read Sample Stage 1 (SDRAM0_TR1[RDSL]=00) and a T0 Sample Cycle (SDRAM0_TR1[RDSS]=00) constitutes the "low latency" path and should only be used if the memory subsystem timing budget has sufficient margin to do so at the target operating frequency for the specific application. In general, the low latency path is more suited for non-ECC applications running at lower frequencies.

It is expected that ECC enabled applications would not use the low latency path. That said, it is up to the DDR SDRAM controller user to make the determination for the specific application and memory subsystem timing budget.

Figure 18-22. SDRAM Timing Register 1 (SDRAM0_TR1)

0:1	RDSS	Read Sample Cycle Select 00 T0 Sample 01 T1 Sample 10 T2 Sample 11 T3 Sample	**Note:** Set RDSS to T2 Sample for CAS latency 2.5 devices.
2:7		Reserved	

8:9	RDSL	Read Sample Stage Select 00 Stage 1 01 Stage 2 10 Stage 3 11 Reserved	**Note:** Set RDSL to Stage 3 for CAS latency 2.5 devices.
10:19		Reserved	
20	RDCD	Read Clock Delay - Stage 2 0 0 clock delay 1 1/2 clock delay	**Note:** Set RDCD to 1 for CAS latency 2.5 devices.
21:22		Reserved	
23:31	RDCT	Read Clock Delay Tuning Bits	Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 30:31, select increments of 1/4 delay element. Bits 23:29, select increments of 1 full delay element. This field should never be programmed to exceed a delay equivalent to 1/2 the Read Clock frequency.

18.2.18 DDR SDRAM Clock Timing Register (SDRAM0_CLKTR)

18.2.18.1 CLKTR Phase Advance and Delay

The DDR SDRAM controller provides the capability to change the timing and phase relationship of the controller-generated DDR Memory clock. This feature allows the user to effectively adjust the DDR memory interface timings, should it be necessary.

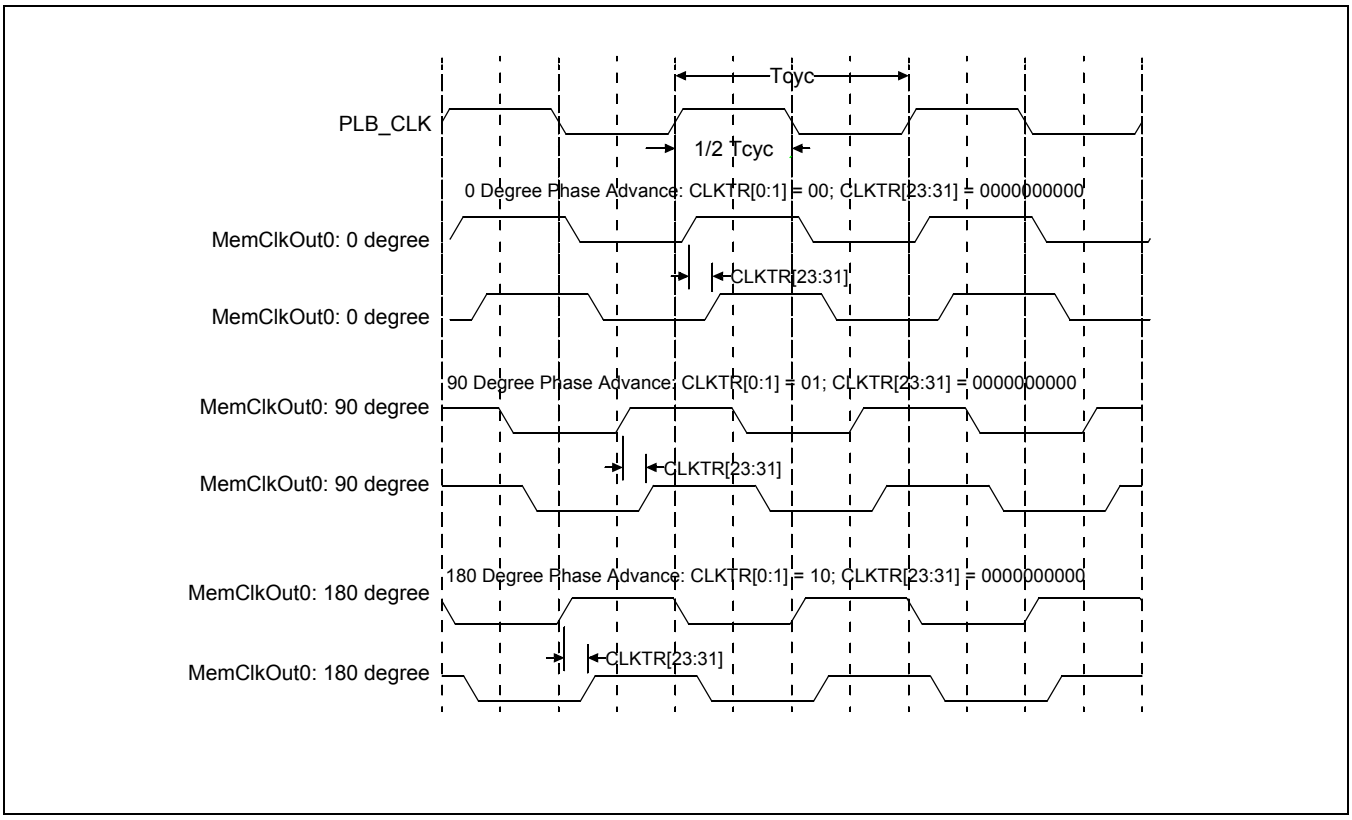
Note: Any modification to the SDRAM0_CLKTR register affects the MemClkOut0 signal. As such, this register should only be modified before initialization, and system software must guarantee adequate time for the "ripple" effects (such as allowing down stream PLLs to re-lock) to stabilize. It is the responsibility of the DDR SDRAM controller user to guarantee that the effect(s) of such modifications on the memory interface timing and timing budget are valid and do not create timing relationships that violate the DDR device specifications, the memory subsystem specific timing budget, or conflict with the overall DDR interface clocking methodology selected.

Note: A complete and thorough understanding of DDR Memory interface protocol and timing, as well as the target memory subsystem timing budget, is an absolute must when considering use of the capabilities provided by the SDRAM0_CLKTR.

The delay effect of the DDR Clock Delay adjusting bits (SDRAM0_CLKTR[23:31]) is dependent on may factors, including delay line physical placement, ASIC process variation, and voltage/temperature variations. The Delay Line Calibration Register (SDRAM0_DLYCAL) may be used to calibrate the adjusting bits.

Figure 18-23 depicts the configurable phase relationships available with SDRAM0_CLKTR. Note that all depicted relationships are with respect to the indicated DDR SDRAM outputs and are independent of any effects of in line path elements such as on-chip or off-chip PLLs, IO driver delays, board wiring, etc. As such, the effects of any in line path elements must be considered when determining the timing relationship of the resultant signals at the DDR memory devices.

Figure 18-23. CLKTR Phase Advance and Delay



The MemClkOut0 signal may be advanced in two coarse increments of 90 and 180 degrees as configured via SDRAM0_CLKTR[0:1]. The indicated phase advance is effectively with respect to the PLB clock signal, which is used to clock the DDR SDRAM controller memory command interface and controller internal registers including the PLB read and write data path interfaces.

With SDRAM0_CLKTR[0:1] = 00, the MemClkOut0 signal is generated in phase, and slightly delayed, with respect to the PLB clock signal. The DDR Clock Delay Tuning bits (SDRAM0_CLKTR[23:31]) provide the ability to add delay, in fine increments, to further adjust the MemClkOut0 signal.

With SDRAM0_CLKTR[0:1] = 01, the MemClkOut0 signal is generated phase advanced by 90 degrees (1/4 Tcyc), and slightly delayed, with respect to the PLB clock signal. The DDR Clock Delay Tuning bits (SDRAM0_CLKTR[23:31]) provide the ability to add delay, in fine increments, to further adjust the MemClkOut0 signal.

With SDRAM0_CLKTR[0:1] = 10, the MemClkOut0 signal is generated phase advanced by 180 degrees (1/2 Tcyc), and slightly delayed, with respect to the PLB clock signal. The DDR Clock Delay Tuning bits (SDRAM0_CLKTR[23:31]) provide the ability to add delay, in fine increments, to further adjust the MemClkOut0 signal.

18.2.18.2 SDRAM0_CLKTR Phase Advance and Delay Considerations

As indicated previously, a complete and thorough understanding of the DDR device interface protocol and timing parameters is recommended. The following information is provided to highlight some of the issues and effects to be evaluated when considering adjusting the DDR Memory clock.

1. Command Interface setup and hold time. Advancing (or not advancing) the DDR memory clock will affect the command interface setup and hold times guaranteed at the device inputs.
2. DDR device Write DQSS minimum/maximum window. Advancing (or not advancing) the DDR memory clock will affect the timing relationship of write DM, DQS, and DATA with respect to the DDR memory clock. The SDRAM0_WDDCTR register provides compatible configuration options to allow this relationship to be adjusted to compensate for SDRAM0_CLKTR modifications. See *Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)* on page 337 for more details.
3. DDR device Read Access time. DDR Read Data and DQS are driven nominally phase aligned with the rising edge of the DDR Memory clock. Advancing the DDR memory clock, with respect to the internal DDR SDRAM controller read data sample clock, advances the read data access time. This is only true if the DDR SDRAM controller read data sample clock is derived from within the ASIC clock generation logic. In general, this is not true if the read data sample clock is derived from an external feedback read data clock derived from the external memory clock.
4. For the majority of DDR SDRAM systems, SDRAM0_CLKTR[0:1] should be set to 90 degrees phase advance.

Figure 18-24. DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR)

0:1	CLKP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved	In most applications, CLKP should be set to 90 degrees phase advance.
2:22		Reserved	
23:31	DCDT	DDR Clock Delay Tuning Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.	This field should never be programmed to exceed a delay equivalent to 1/4 the MemClkOut0 frequency.

18.2.18.3 DDR Delay Line Register (SDR0_DDRDL0)

DDR delay line register (SDR0_DDRDL0) is the PPC440EP system DCR register which is accessed indirectly through the SDR0_CFGADDR and SDR0_CFGDATA registers using the **mtdcr** and **mfdcr** instructions. This system reserved register is auto configured for a memory clock range of 100 to 166 MHz and is not intended for user manipulation. The following figure describes the SDR0_DDRDL0 register bits.

Reset value = 0x00000012

Figure 18-25. DDR Delay Line Register (SDR0_DDRDL0)

0:25		Reserved	
26:31	TUNE	DDR Delay Line TUNE	

18.2.19 Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

Figure 18-26 describes the SDRAM0_WDDCTR register bits described in detail in the next subsections.

Figure 18-26. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

0:1	WRCP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved	
2:22		Reserved	
23:31	DCD	DDR Write Data/DM/DQS Clock Delay Tuning Bits Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.	

18.2.19.1 WDDCTR Phase Advance and Delay

The DDR SDRAM controller provides the capability to change the timing and phase relationship of the controller generated DDR Memory write data, write data mask, and write data strobe signals. The afore mentioned group of signals will be collectively referred to as the "write data and control group" in the following paragraphs. This feature allows the user to effectively adjust the DDR memory interface timings for Tdqss, should it be necessary. The DDR device Tdqss window is defined with respect to the DDR Memory clock at the device. As such, the SDRAM0_WDDCTR register, in conjunction with the SDRAM0_CLKTR register, can be used to configure the relationship of the "write data and control group outputs" with respect to the MemClkOut0 signal.

It is the responsibility of the chip user to guarantee that the effect(s) of such modifications on the memory interface timing and timing budget are valid and do not create timing relationships that violate the DDR device specifications, the memory subsystem specific timing budget, or conflict with the overall DDR interface clocking methodology selected.

A complete and thorough understanding of DDR Memory interface protocol and timing, as well as the target memory subsystem timing budget is an absolute must when considering use of the capabilities provided by SDRAM0_WDDCTR.

18.2.19.2 WDDCTR/CLKTR Relationships

The following figures depict the range of timing relationships for the "write data and control group" outputs that may be configured through SDRAM0_WDDCTR. The relationships are depicted with respect to the MemClkOut0 signal and the corresponding setting for SDRAM0_CLKTR. The effects of a non-zero value setting for SDRAM0_CLKTR[23:31] or SDRAM0_WDDCTR[23:31] are not explicitly depicted; however, the effects of such settings must be considered when configuring the relative relationships depicted. Also, all depicted relationships are with respect to the indicated chip outputs and are independent of any effects of in line path elements such as on-chip or off-chip PLLs, IO driver delays, board wiring, etc. As such, the effects of any in line path elements must be considered when determining the timing relationship of the resultant signals at the DDR memory devices.

The "write data and control group" signals may be advanced in two coarse increments of 90 and 180 degrees as configured via SDRAM0_WDDCTR[0:1]. The indicated phase advance is effective with respect to the PLB clock, which is used to clock the DDR SDRAM controller's memory command interface and controller internal registers, including the PLB read and write data path interfaces.

Figure 18-27 depicts the default configuration setting with no phase advance on SDRAM0_CLKTR or SDRAM0_WDDCTR. This effectively configures the DDR SDRAM controller to launch the "write data and control group" coincident with the $T_{dqss(min)}$ timing reference defined by the DDR device specification.

Figure 18-27. CLKTR 0 Degree Phase Advance and WDDCTR 0 Degree Phase Advance

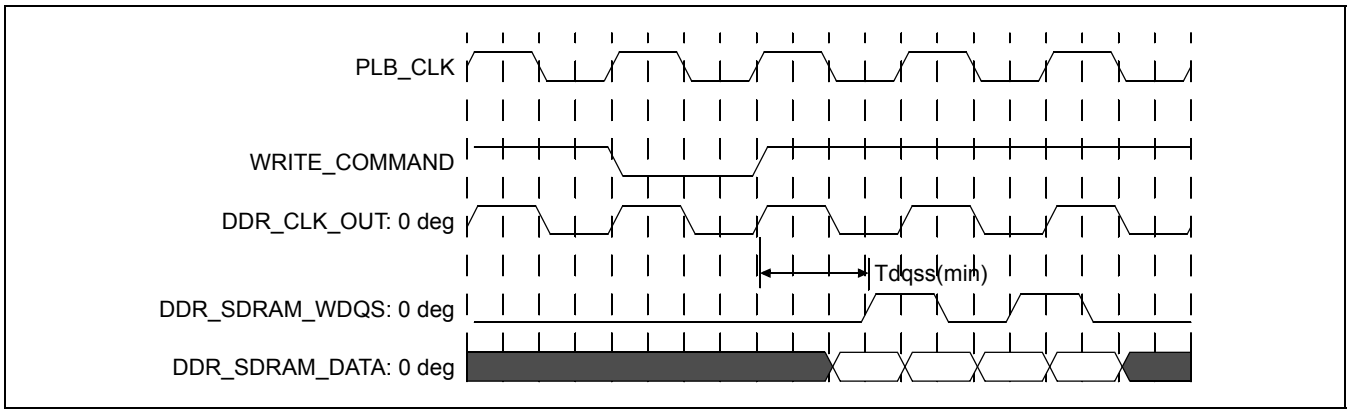
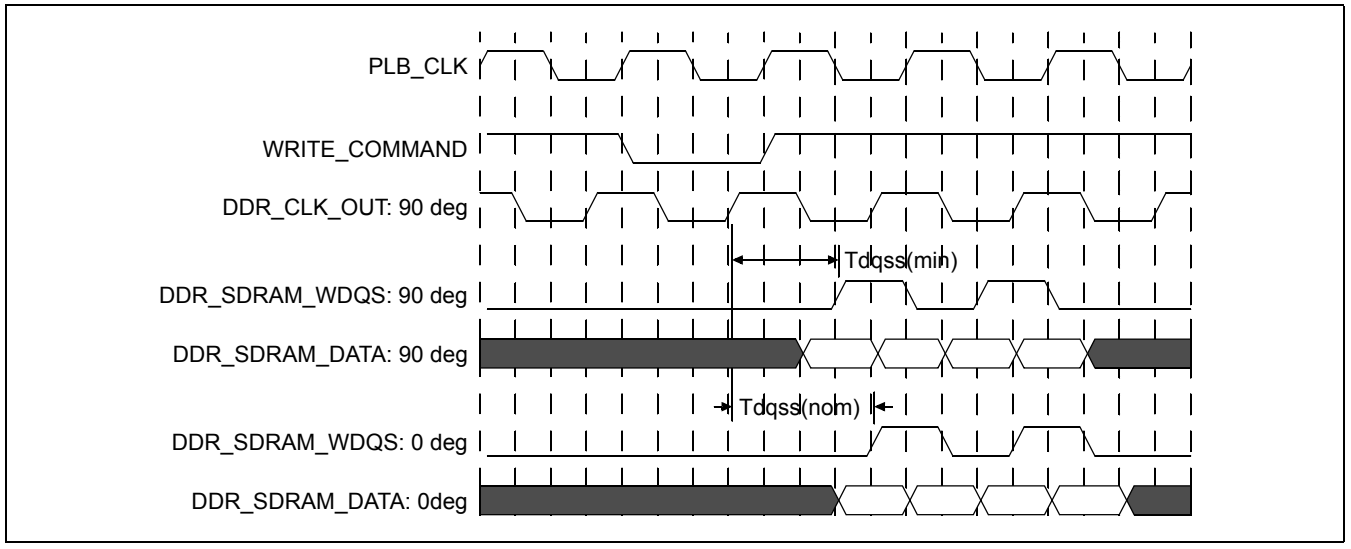


Figure 18-28 depicts a 90 degree phase advance on SDRAM0_CLKTR. With SDRAM0_WDDCTR configured for a 90 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(min)}$ timing reference defined by the DDR device specification. With SDRAM0_WDDCTR configured for a 0 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(nom)}$ timing reference defined by the DDR device specification.

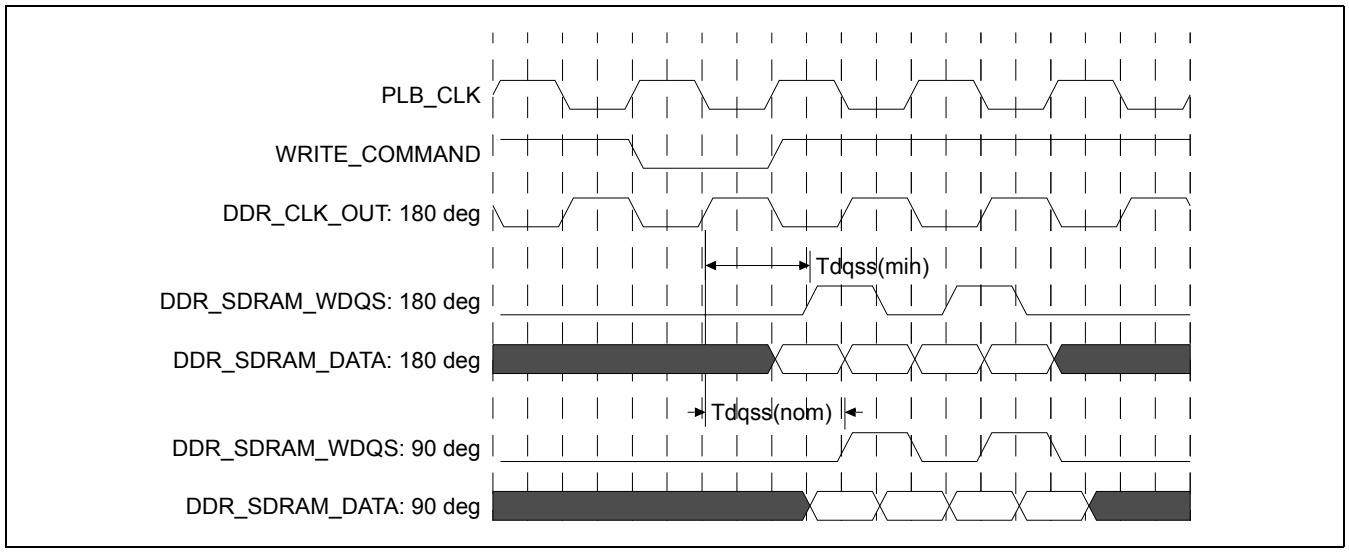
Figure 18-28. CLKTR 90 Degree Phase Advance and WDDCTR 90/0 Degree Phase Advance



Note: The effect (setup/hold) on the DDR command interface (WRITE_COMMAND) of the SDRAM0_CLKTR 90 degree phase advance.

Figure 18-29 depicts a 180 degree phase advance on SDRAM0_CLKTR. With SDRAM0_WDDCTR configured for a 180 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(min)}$ timing reference defined by the DDR device specification. With WDDCTR configured for a 90 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(nom)}$ timing reference defined by the DDR device specification.

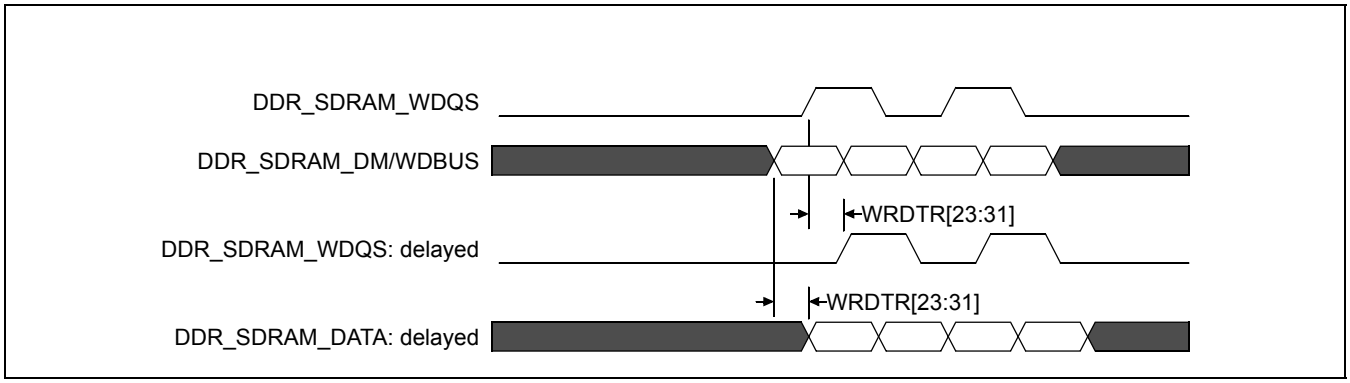
Figure 18-29. CLKTR 180 Degree Phase Advance and WDDCTR 180/90 Degree Phase Advance



Note: The effect (setup/hold) on the DDR command interface of the SDRAM0_CLKTR 180 degree phase advance.

As illustrated in *Figure 18-30*, the DDR Write Data/DM/DQS Clock Delay Tuning Bits (SDRAM0_WDDCTR[23:31]) provide the ability to further adjust the "write data and control group" by adding delay in fine increments. The delay effect of the DDR Write Data/DM/DQS Clock Delay Tuning Bits (SDRAM0_WDDCTR[23:31]) is dependent on may factors including delay line physical placement, ASIC process variation, and voltage/temperature variations. The Delay Line Calibration Register (SDRAM0_DLYCAL) may be used to calibrate the tuning bits.

Figure 18-30. SDRAM0_WDDCTR Delay



18.2.20 Delay Line Calibration Register (SDRAM0_DLYCAL)

SDRAM0_DLYCAL register provides software with a hardware-based method of calibrating the various programmable delay lines available for use in the DDR SDRAM controller. These delay lines are not associated with the SDR0_DDRDL delay PLL.

Figure 18-31. Delay Line Calibration Register (SDRAM0_DLYCAL)			
0	DLCR	Delay Line Calibration Request 0 Delay Line Calibration not requested 1 Delay Line Calibration requested	Setting DLCR bit initiates the delay line calibration. During the re-initialization process, the DLCR bit is cleared. The delay line calibration occurs automatically following SysReset or upon request by setting the DLCR bit. The results of the calibration are provided to assist in the setting of SDRAM0_CLKTR, SDRAM0_WRDTR, and SDRAM0_SDTR2.
1:3	DLCS	Delay Line Calibration Status 000 Delay Line Calibration not run 001 Delay Line Calibration in progress 010 Delay Line Calibration complete 100 Delay Line Calibration error	This two bit field indicates the real-time status of the calibration process at the time of the register read. Once complete (DLCS = 010), the DLY_VAL field is valid. A delay line calibration status of "error" indicates that the calibration process completed without successfully determining the value of DLCV. Sufficient delay stages have been included in the calibration delay line such that this completion status should never occur within the supported operating frequency range.
4:21		Reserved	
22:29	DLCV	Delay Line Calibration Value	This 8-bit binary encoded value indicates the number of delay elements in a single 2x clock cycle or a single 1x half clock cycle. Bit 22 is Most Significant Bit; bit 29 is Least Significant Bit.
30:31		Reserved	

18.2.21 ECC Error Status Register (SDRAM0_ECCESR)

The ECC Error Status Register (SDRAM0_ECCESR) tracks ECC related errors encountered during SDRAM memory PLB master accesses to system memory. Bits in SDRAM0_ECCESR are cleared by writing a 32-bit value to SDRAM0_ECCESR with a 1 in any bit position that is to be cleared and 0 in all other bit positions.

Figure 18-32. ECC Error Status Register (SDRAM0_ECCESR)			
0:15	BnCE	Byte Lane n Corrected Error (where n=0-15) 0 No Error 1 Error Occurred in Byte Lane n	
16:17	CKBE	Error Detected in check bits 64-bit Mode 00 No Error 01 Reserved 10 Error in check bits 11 Reserved 32-bit Mode 00 No Error 01 Error in Lower check bits 10 Error in Upper check bits 11 Error in Upper and Lower check bits	
18	CE	Correctable Error	
19	UE	Uncorrectable Error	

20:23	BKnE	Bank n Error (where n=0-3). The possible values for each bit are: 0 No Error 1 Error Occurred in Bank n	This 4-bit field indicates the bank in which an error occurred; bit 20 (BK0E) = bank 0, and so on.
24:31		Reserved	

18.2.22 Controller ID Register (SDRAM0_CID)

The Controller ID Register is a "Read Only" register implemented to identify the DDR SDRAM controller and its history.

Figure 18-33. Controller ID Register (SDRAM0_CID)

0:11	CID	Core ID identifies the IBM specific core number associated with the DDR SDRAM.	
12:15	CDC	Core Derivative Code B Base Library Core D Derivative of Base Library Core	
16:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) Netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding Netlist modification.	

18.2.23 Revision ID Register (SDRAM0_RID)

The Revision ID Register is a "Read Only" register implemented to identify the DDR SDRAM controller revision level and revision history.

Figure 18-34. Revision ID Register (SDRAM0_RID)

0:7	CDID	Controller Derivative ID nn = controller derivative ID number nn = 00 indicates the Base Library Controller (CDC = B) nn = A non-zero values indicates the specific controller derivative ID number and is tracked by development (CDC = D).	
8:23	RCID	Controller Revision Control ID cccc indicates the controller SCCS revision control ID associated with the specific version of the controller.	
24:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) Netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding Netlist modification.	

The Controller ID Register and the Revision ID Register, in combination, provide important tracking information about the specific DDR SDRAM version implemented.

18.3 Initialization

The following two steps are needed to begin proper initialization.

1. After reset is deactivated, pause the amount of time indicated in the DDR SDRAM memory device specification. No requests to memory controller can occur during the pause period. Configure DDR SDRAM configuration registers (can occur during pause period). If modifying SDRAM0_CLKTR, sufficient time must be allowed for any internal and/or external in-line PLLs to stabilize and lock before proceeding to Step 2.

Note: Step 2 can only occur after SDRAM configuration is completed in step 1.

2. Set SDRAM0_CFG0[DCEN] = 1 to enable the SDRAM controller.

The following steps are then performed by the DDR SDRAM controller to complete initialization:

1. Asserts CKE High.
2. Issues precharge command to all banks.
3. Waits minimum number of clock cycles as defined by SD_PTA.
4. Issues Extend Mode Register Set command (EMRS) to Enable DDR SDRAM DLL.
5. Issues Mode Register Set command (MRS1) to load mode register and to reset DDR SDRAM DLL.
6. Issues precharge command to all banks.
7. Performs two auto refresh cycles (each separated by SD_RFTA clock cycles).
8. Issues Mode Register Set (MRS2) command to enable normal operation.
9. Waits for a minimum count of 200 cycles from step 4 to expire.
10. Makes DDR SDRAM available for access.

18.4 DDR to Memory Address Decode

The DDR SDRAM supports the extended addressing capability of the PLB and can be located within any 4-GB aligned region of the 64-GB address space available given the 36-bit PLB address. The range of addresses to which the DDR SDRAM responds within the programmed 4-GB aligned region can be further limited to the specific range of addresses for which physical memory is installed.

Note: Moving the memory map may cause overlap with other mapped cores.

The specific range of addresses to which the PLB slave will respond is defined and configured by the combination of the PLB UABus Base Address Register (SDRAM0_UUABA) and the Memory Configuration Registers (SDRAM0_B0CR:B3CR). For example, the base address for memory bank 0 is defined by SDRAM0_UUABA[28:31] and SDRAM0_B0CR[0:7]; likewise, the base address for memory bank 1 is defined by SDRAM0_UUABA[28:31] and SDRAM0_B0CR[0:7]. The PLB slave interface will respond to any PLB access that targets a defined base address and lies within the size of the physical memory region defined for the enabled memory bank provided that the access type is supported.

18.5 DDR Slave Interface Options

The DDR SDRAM PLB slave interface has several configurable options to allow users to change the way in which the DDR SDRAM responds to read and write requests for various conditions such as buffer full, address queue full, and read/write priority. These options are explained in the following sections and configured via the PLB Slave Interface Options (SDRAM0_SLIO) register.

18.5.1 Read Rearbitration

The DDR SDRAM will immediately accept any pending read request provided that an address queue entry is available. If the address queue is full, the default behavior of the DDR SDRAM is to WAIT the request until an entry in the address queue is available. The WAIT signal conveys to the PLB arbiter that the DDR SDRAM is unable to accept the current read request and that the PLB arbiter should maintain the current request assertion until the DDR SDRAM is able to accept the read request. This default behavior can be changed by setting SDRAM0_SLIO[0] (RDRE) to enable PLB slave read rearbitration. When RD_RE is set, any read request received when the address queue is full (PLB slave not able to accept immediately) will trigger a REARBITRATE signal and allow the PLB arbiter to re-arbitrate the PLB.

18.5.2 Write Rearbitration

The DDR SDRAM will immediately accept any pending write request provided that an address queue entry is available. If the address queue is full, the default behavior of the DDR SDRAM is to WAIT the request until an entry in the address queue is available. The WAIT signal conveys to the PLB arbiter that the DDR SDRAM is not able to accept the current write request and that the PLB arbiter should maintain the current request assertion until the DDR SDRAM is able to accept the write request. This default behavior can be changed by setting SDRAM0_SLIO[1] (WRRE) to enable PLB slave write rearbitration. When WR_RE is set, any write request received when the address queue is full or if there is less than 32 bytes of free space in the write buffer will trigger a REARBITRATE signal and allow the PLB arbiter to re-arbitrate the PLB Bus.

18.5.3 Read Around Write

The default behavior of the DDR SDRAM is to complete all read and write requests in the order in which each request was accepted.

Setting SDRAM0_SLIO[3] (RARW) enables the "read around write" feature which enables queued, non-burst read requests to bypass previously posted writes as long as the target address range of the read does not conflict with the target address range of the posted writes that precede the read request. In this mode, all writes will be completed in order with respect to all other writes and all reads will be completed in order with respect to all other reads. Also, to insure that the write pipe does not stall when in this mode, the read around write operations will be interleaved with the previously posted write cycles. For example, if there are 3 write requests queued followed by 3 non-burst, non-conflicting reads, the sequence to the memory would be write-read-write-read-write-read. Burst read requests are not allowed to bypass or read around any previously posted write cycle.

The minimum granularity of the address check for read around write candidates is on a 32-byte aligned boundary. When queued, ending address of each write is calculated based on the size of the transfer. Subsequent read request addresses are compared to the write starting address (bits [0:26]) and the write ending address (bits [0:26]). Non-conflicting (based on address compare), queued reads are marked for bypass and do so at the earliest available opportunity.

18.6 Basic DDR SDRAM Memory Requirements

The DDR SDRAM controller supports 100 and 133 MHz, quad-bank DDR SDRAMs with a x32 or x64-bit wide bus (x40/x72 supported if ECC) with **one DQS per byte** and uses fully programmable timing parameters. The DDR SDRAM controller supports the *Double Data Rate (DDR)SDRAM Specification*, JESD79, JEDEC Standard.

18.7 Page Management

The DDR SDRAM paging policy is configurable in that the Page Management Unit (PMU) may be enabled or disabled and the page deallocation policy set to least recently used or least recently allocated.

18.7.1 PMU Enabled

The DDR SDRAM controller supports page mode operation with bank interleaving and maintains up to eight open pages in the memory subsystem. Subsequent accesses that target an open page result in a page hit.

The DDR SDRAM controller page management unit (PMU) tracks memory accesses (activate, read/write, precharge, and refresh) and maintains a directory of up to eight open pages. All PMU entries are allocated and deallocated based on current and pending accesses. Allocated entries are checked against the address of the pending access, and a page hit is signaled to the DDR SDRAM control logic when a match occurs. All PMU directory entries are deallocated when the DDR SDRAM control logic issues a precharge all command to the DDR SDRAM in preparation for an auto refresh cycle.

The PMU supports activating and accessing a ninth page for a non-conflicting address in the memory subsystem before closing one of the eight open pages. A non-conflicting address is any address whose chip select and bank address are not currently allocated in the PMU. Access to this ninth page (activate and read or write command) occurs followed by a precharge command to one of the other eight open pages. If access to the ninth page conflicts with an open page, the DDR SDRAM controller issues a precharge command to the bank before performing the access.

In the event that all 8 PMU entries are allocated and an access occurs that does not target one of the open allocated pages in memory (page miss), the PMU entry must be deallocated and the PMU directory updated with the page tag information for the new access. The PMU entry deallocated depends on the configured page deallocation policy.

If configured for Pseudo Least Recently Allocated (PMU_DP = 0), PMU entries are allocated in a circular queue with each new page accessed being allocated in the next queue entry. If an access conflicts with an allocated entry, the PMU directory entry of the conflicting page closed by the corresponding precharge command is updated with the new page address. Subsequent accesses (page hits) to previously allocated entries do not affect the position of an allocated entry in the queue. Once all PMU entries are allocated (8 open pages), an access to a non-conflicting address will trigger a deallocation and replacement of the next PMU entry indexed by the queue pointer.

If configured for Least Recently Used (or Least Recently Allocated) (PMU_DP = 1), the PMU directory is maintained as a stack that is updated with each access such that the page most recently accessed is on top of the stack (entry 1) and the page least recently accessed is the highest numbered allocated entry in the stack (entry 8 in the case of a fully allocated PMU directory). Each new page allocated is pushed onto the stack at entry 1. Accesses that target a conflicting address result in that entry being deallocated and the corresponding page in memory precharged, with the new page address being pushed onto the stack. For example, if an access occurs that conflicts with PMU directory entry 5, the page associated with entry 5 is deallocated, entries 1 through 4 are shifted to locations 2 through 5 respectively, and the new page is allocated in entry 1 (entries 6 through 8 are unaffected), maintaining the LRU ordering. Accesses that page hit, result in the corresponding PMU entry being moved to the top of the stack and all prior entries being pushed to that point into the stack, again maintaining the

LRU ordering. Once all PMU entries are allocated (8 open pages), an access to a non-conflicting address will trigger a deallocation of entry 8 (least recently accessed page), with entries 1 through 7 being shifted to entries 2 through 8, respectively, and the new page being allocated at the top of the stack (entry 1).

Open pages can be spread across the system memory array on multiple chip selects or be contained in a single chip select, depending on the memory access sequences and the memory subsystem implementation. For a single bank (single chip select) memory subsystem, the number of open pages is limited to 4, which is the number of internal banks associated with the SDRAM device in that bank. In this case, the maximum of four open pages is a limitation of the memory subsystem implementation, not the DDR SDRAM controller.

The SDRAM page size for page hits varies, depending on programmed addressing mode which is derived from the DDR SDRAM organization. *Table 18-6* and *Table 18-7* details the relationship of the address mode to the page size:

Table 18-6. 32-Bit SDRAM Page Size

Address Mode	Page Size
1	1 KB
2	2 KB
3	4 KB
4	8 KB

Table 18-7. 64-Bit SDRAM Page Size

Address Mode	Page Size
1	2 KB
2	4 KB
3	8 KB
4	16 KB

18.7.2 PMU Disabled

The DDR SDRAM controller also supports disabling the PMU for applications with random memory access sequences or for any application that will not benefit from the multiple open pages maintained when the PMU is enabled. With the PMU enabled, random and non-page local accesses may alter the PMU directory, yielding the worst case latency associated with a page miss with conflict. This necessitates a precharge-activate-read/write sequence. This additional precharge overhead can be eliminated by disabling the PMU.

With the PMU disabled, no pages are kept open in the memory subsystem. Each individual PLB request will result in the DDR SDRAM controller performing an activate command, followed by N read/write commands (where N=1 for non-Burst PLB accesses and N>=1 for Burst PLB accesses), where the final (or only) access is a read/write with auto-precharge command.

Note: Allowing page hits for the duration of the sequential accesses of a PLB Burst enables this mode to sustain the bandwidth without incurring the activate penalty for each individual access of the burst while satisfying a singular PLB burst read/write request.

In this mode, the memory latency for any PLB access will be the "Page Idle" latency as no pages are maintained open for subsequent accesses; thus, there can never be a page hit, nor can there be a page miss with conflict for any PLB access.

18.7.3 Physical Address to Memory Address Mapping

The DDR SDRAM controller supports various quad-bank DDR SDRAM device densities including 64,128, 256, and 512 Mb in x8, x16, and x32 organizations, and it uses one DQS signal per byte. The addressing configuration associated with the specific device(s) organization and density determines the corresponding addressing mode(s), independent of the packaging of those devices (DIMM, SODIMM, or Planar).

Table 18-8 summarizes the supported addressing modes for DDR SDRAM densities and configurations required.

Row and Column addresses are generated based on the organization of the SDRAM banks as defined in the memory timing registers (SDRAM0_B0CR:B3CR). *Table 18-8* and *Table 18-9* show the mapping of the physical address to memory address for all memory accesses based on the configured mode for the associated bank; the DDR SDRAM device organization is row × column (internal banks).

Table 18-8. 32-Bit DDR SDRAM Addressing Modes

Mode	SDRAM Config		BA1	BA0	MA 12	MA 11	MA10 /AP	MA 9	MA 8	MA 7	MA 6	MA 5	MA 4	MA 3	MA 2	MA 1	MA 0
1	Nx8 (4)	Row	20	21	7	8	9	10	11	12	13	14	15	16	17	18	19
		Col	20	21			AP			22	23	24	25	26	27	28	29
2	Nx9 (4)	Row	19	20	6	7	8	9	10	11	12	13	14	15	16	17	18
		Col	19	20			AP		21	22	23	24	25	26	27	28	29
3	Nx10 (4)	Row	18	19	5	6	7	8	9	10	11	12	13	14	15	16	17
		Col	18	19			AP	20	21	22	23	24	25	26	27	28	29
4	Nx11 (4)	Row	17	18	4	5	6	7	8	9	10	11	12	13	14	15	16
		Col	17	18		19*	AP	20	21	22	23	24	25	26	27	28	29

Table 18-9. 64-Bit DDR SDRAM Addressing Modes

Mode	SDRAM Config		BA1	BA0	MA 12	MA 11	MA10 /AP***	MA 9	MA 8	MA 7	MA 6	MA 5	MA 4	MA 3	MA 2	MA 1	MA 0
1	Nx8 (4)	Row	19	20	6	7	8	9	10	11	12	13	14	15	16	17	18
		Col	19	20	**		AP			21	22	23	24	25	26	27	28
2	Nx9 (4)	Row	18	19	5	6	7	8	9	10	11	12	13	14	15	16	17
		Col	18	19			AP		20	21	22	23	24	25	26	27	28
3	Nx10 (4)	Row	17	18	4	5	6	7	8	9	10	11	12	13	14	15	16
		Col	17	18			AP	19	20	21	22	23	24	25	26	27	28
4	Nx11 (4)	Row	16	17	3	4	5	6	7	8	9	10	11	12	13	14	15
		Col	16	17		18*	AP	19	20	21	22	23	24	25	26	27	28

N = 11, 12, or 13

*Column address bit 10 sent out on MA11 for 13 x 11 (4) parts.

**Blank entries are "don't care" for an addressing mode.

***AP is driven based on the specific type of read or write access occurring. With page mode enabled, AP=0 for all read and write commands. With page mode disabled, AP=0 for reads/writes that are part of a sequential PLB burst access and AP=1 for the last read/write of any access.

18.8 SDRAM Commands and Operations

The DDR SDRAM controller supports the following commands:

- Read
- Write
- Activate
- Single-Bank Precharge
- Read/Write with Auto-Precharge
- Precharge All Banks
- Auto CAS Before RAS (CBR) Refresh
- Self-Refresh (Hardware and Software Initiated)

18.8.1 DDR SDRAM Timing Parameters

Programmable memory timing support provides for flexibility at the system level. As a result, memory timing parameters can be adjusted to support faster SDRAM technologies as they become available, and maximize the performance of the SDRAM memory subsystem.

Note: All SDRAM timing diagrams illustrate cycle-based programmable timing parameters only. AC specific timing information should not be inferred from the timing diagrams.

Table 18-10 summarizes the SDRAM memory timing parameters.

Table 18-10. SDRAM Memory Timing Parameters

Name	Function	Description
SD_RCD	Activate to Read/Write Command	Minimum number of clock cycles from an Activate Command to a Read or Write Command. Corresponds to DRAM RAS to CAS assertion delay.
SD_RFTA	Refresh to Activate	Minimum number of clock cycles from a CBR Refresh Command to the next Activate Command.
SD_CTP	Command to Precharge	Sets the minimum number of clock cycles from a Read or Write Command to the Precharge Command.
SD_PTA	Precharge to Activate	Minimum number of clock cycles required to wait following a Precharge Command to issue the next Activate Command.
SD_RRD	Bank A Activate to Bank B Activate	Module Bank-to-Bank Activate Delay. Hard coded to 2-clock cycles. Pertains to internal banks only. This parameter is fixed and not programmable.
SD_SRX	Self Refresh Exit Delay	Minimum number of clock cycle until access to DDR SDRAM allowed following self refresh exit. 200+ clocks to allow for DLL to lock. This parameter is fixed and not programmable.
SD_CASL	$\overline{\text{CAS}}$ Latency	$\overline{\text{CAS}}$ access latency.
SD_LDF	Command Leadoff Delay	Number of clock cycles from address/command assertion to chip select assertion.

The following timing diagrams illustrate the relationship of the programmable timings for activate, read, write, and precharge commands:

Figure 18-35. Activate - Read - Precharge - Activate

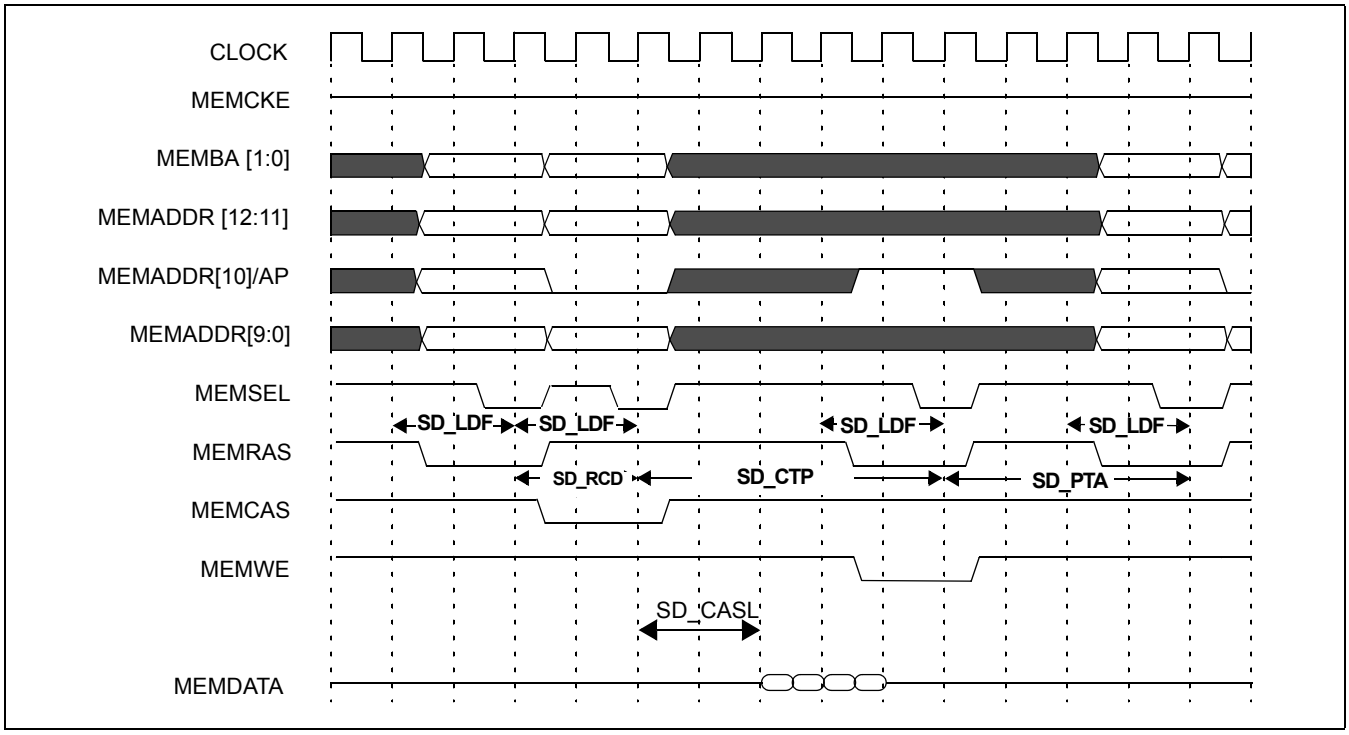


Figure 18-36. Activate - Write - Precharge - Activate

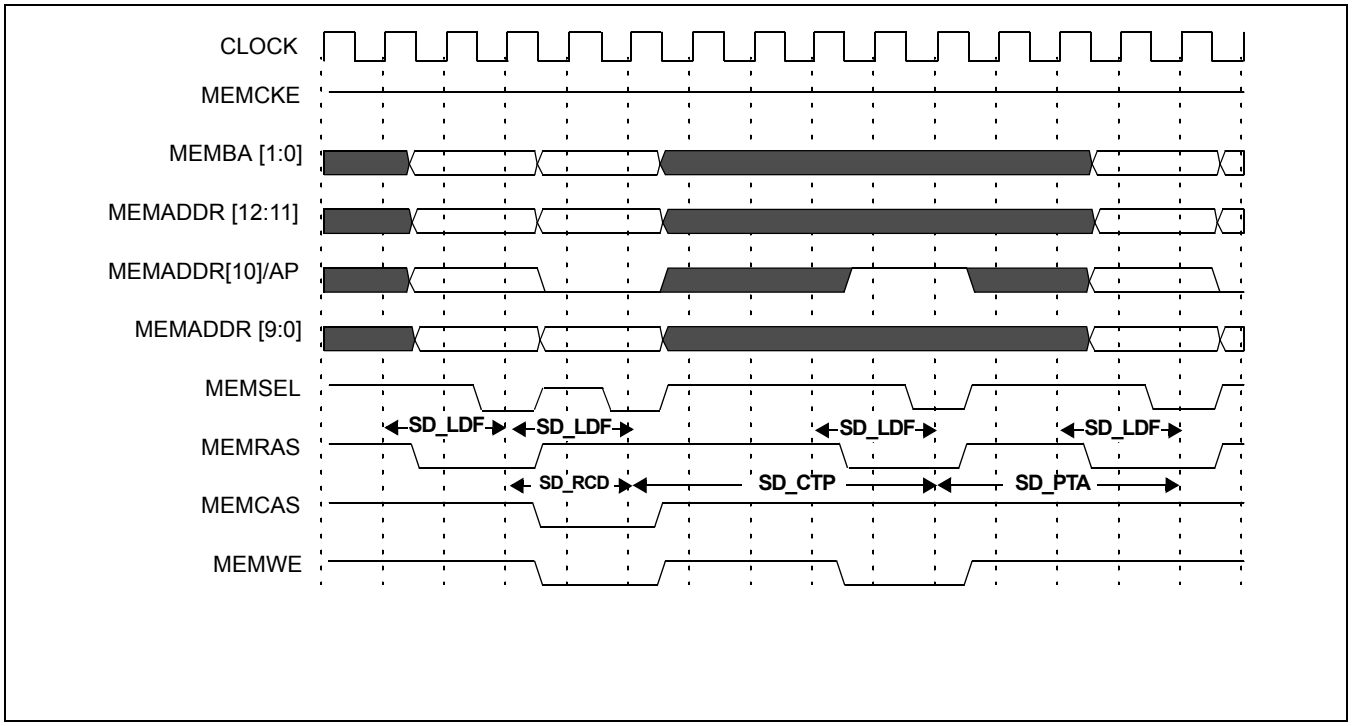
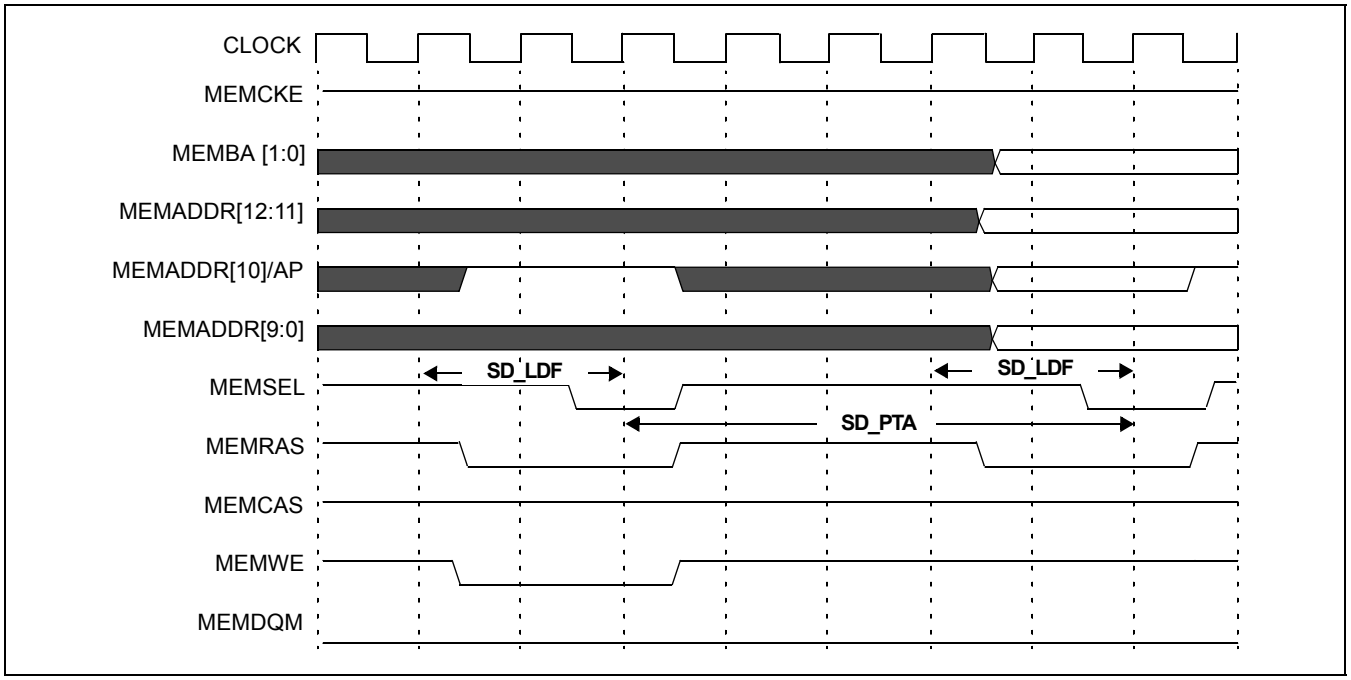


Figure 18-37. Precharge All - Activate



18.8.2 Precharge Command

The precharge command instructs the SDRAM to precharge some or all banks and is generated by the DDR SDRAM controller. The memory address bus contains the command associated with the precharge cycle, which is formatted as shown in *Table 18-11*.

Table 18-11. Precharge Command

Command	BA1	BA0	MA 12	MA 11	MA10 / AP	MA 9	MA 8	MA 7	MA 6	MA 5	MA 4	MA 3	MA 2	MA 1	MA 0
Precharge All Banks					b'1'										
Single Bank Precharge for Chip Select N	BA1 _N	BA0 _N			b'0'										

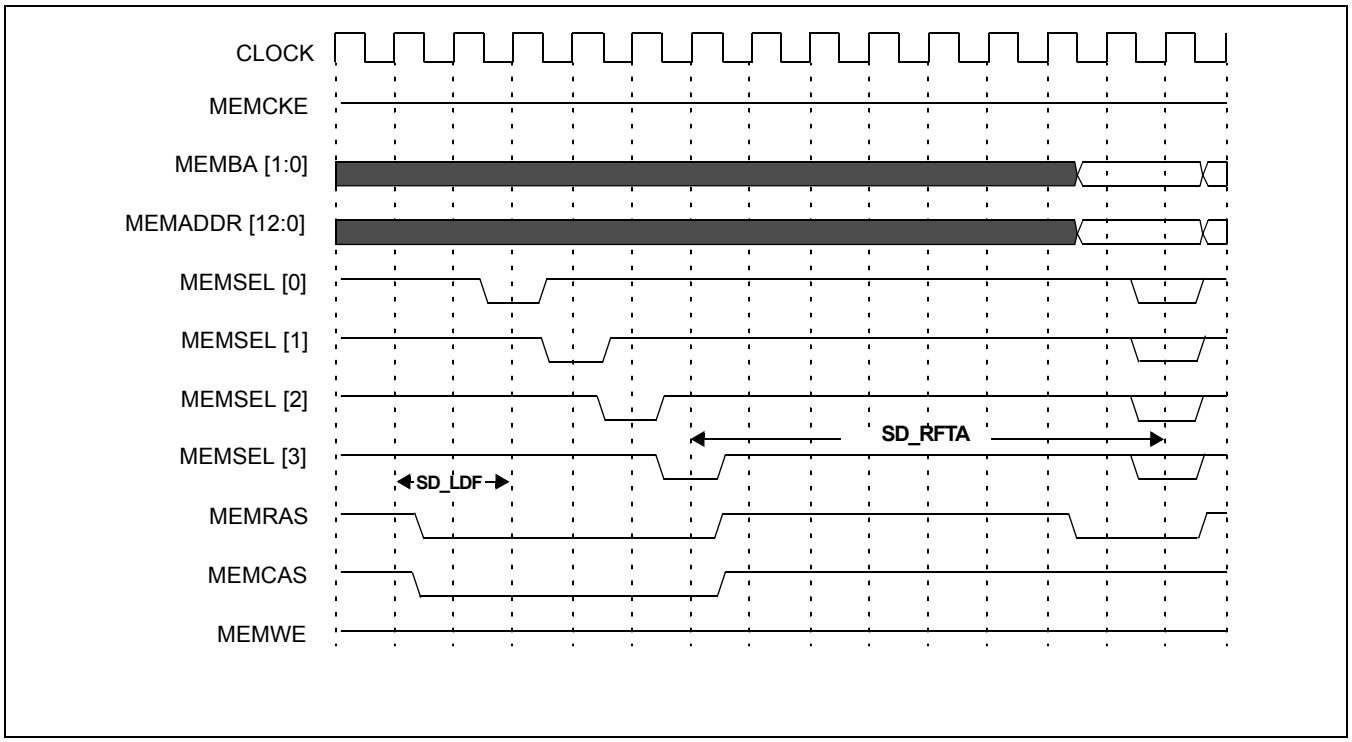
Note: Shaded entries in *Table 18-11* are "don't care" and are driven with a stable value for every clock cycle.

18.8.3 Refresh

Refresh of odd memory banks is staggered from the refresh of even memory banks. Only banks enabled in the memory bank configuration register are initialized following reset and refreshed during normal operation. Once the memory controller is enabled and the initialization sequence has completed, the refresh mechanism starts automatically with refreshing of the memory continuing independent of SDRAM0_CFG0[DCEN].

Refresh requests are generated internally when the refresh timer expires. The refresh interval is programmable using the Refresh Timer Register (SDRAM0_RTR). During refresh, all SDRAM accesses are delayed until the current or pending refresh cycle completes.

Figure 18-38. Auto (CBR) Refresh



18.8.4 Self-Refresh Operation

The DDR SDRAM controller supports both hardware-initiated and software-initiated self-refresh. In either case, the self-refresh exit trigger mechanism is controlled by resetting (once set) SDRAM0_CFG1[SRE].

Note: Only the SDRAM memory is affected by the self-refresh operation.

Although not required, it is recommended that all pending and previously queued requests targeting the DDR SDRAM controller be allowed to complete before entering software-initiated self-refresh. Any pending or previously queued requests which have not completed prior to entering software-initiated self-refresh will stall (including the PLB handshake if in progress) until self-refresh is exited by clearing SDRAM0_CFG1[SRE] or by asserting RESET.

18.8.4.1 Software-Initiated Self-Refresh Operation

The SDRAM Controller supports self-refresh operation for applications desiring lower power.

Entry

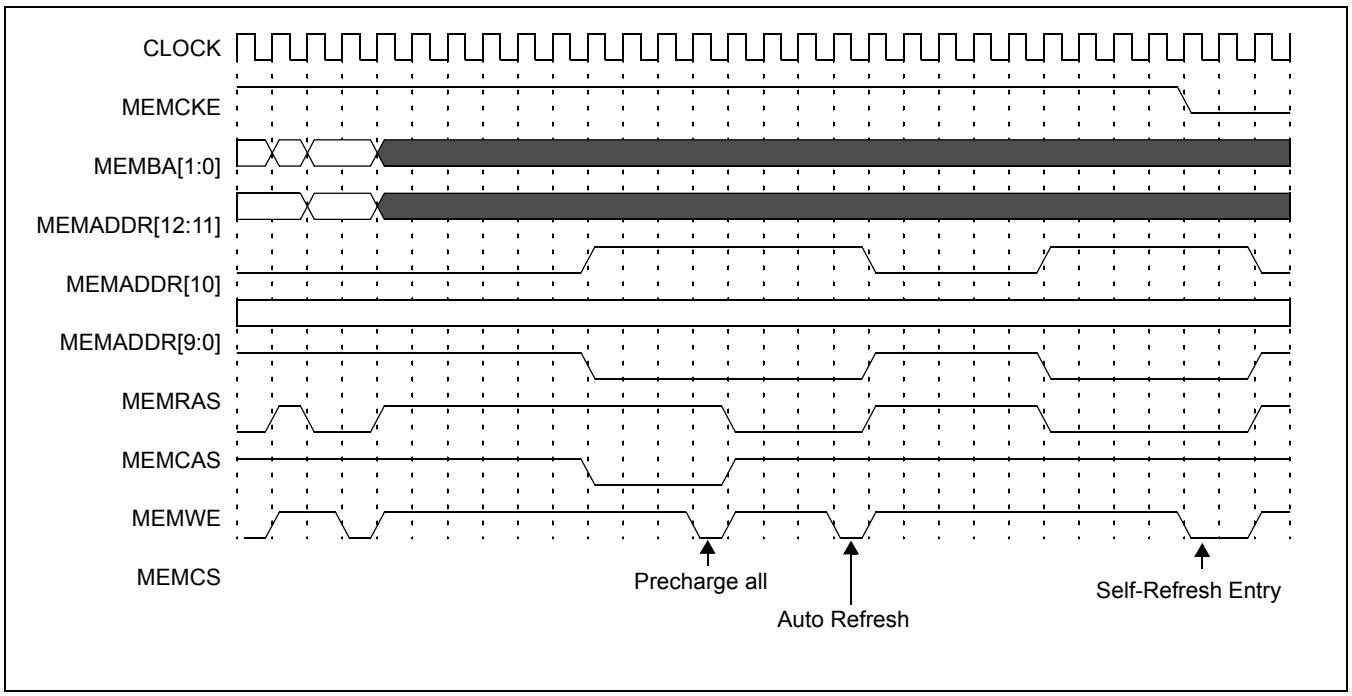
Self-refresh entry by software is initiated by setting SDRAM0_CFG1[SRE]. When set, the DDR SDRAM controller will perform the following:

1. Complete the current request.
2. Perform precharge all to close all open pages.
3. Perform an auto-refresh cycle.

4. Enter self-refresh mode and set SDRAM0_MCSTS[SRMS].

The DDR SDRAM controller will maintain the SDRAM in self-refresh mode, independent of any pending memory access request, until SDRAM0_CFG1[SRE] is cleared.

Figure 18-39. Self-Refresh Entry

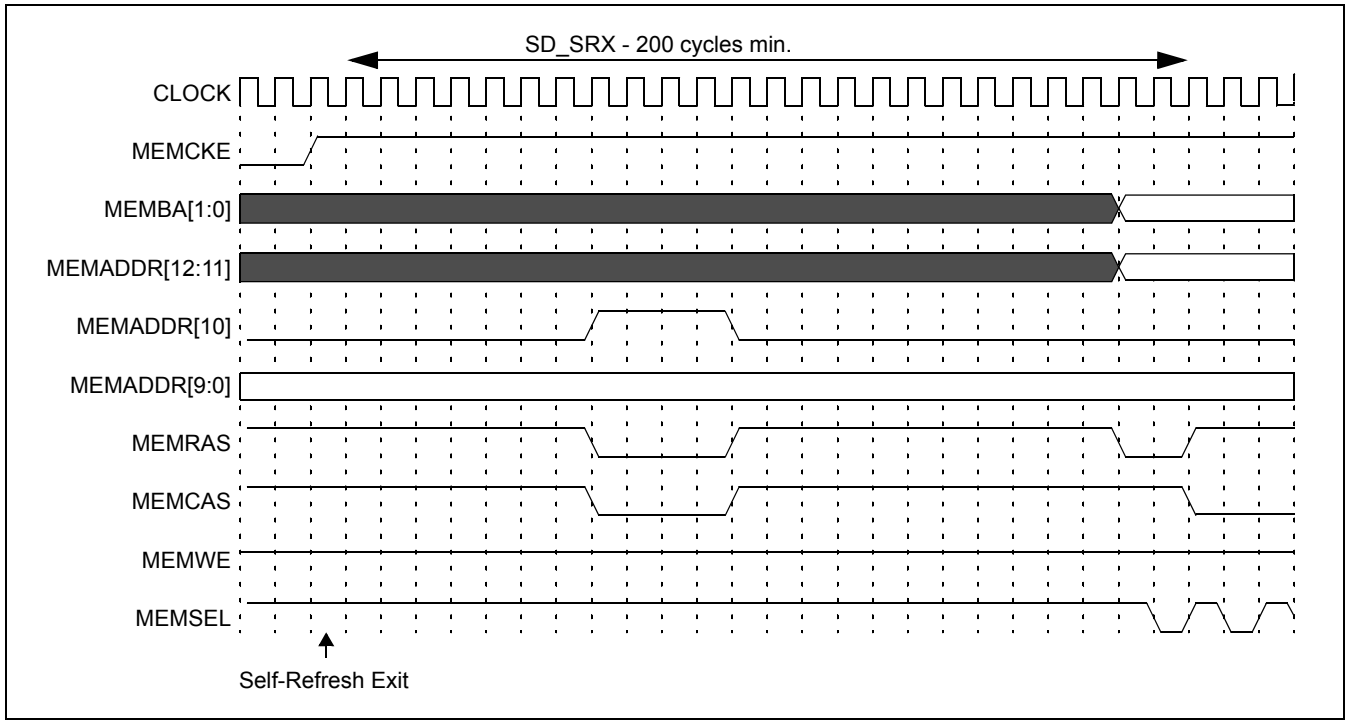


Exit

Once SDRAM0_CFG1[SRE] is cleared, the SDRAM controller will perform the following:

1. Exit self-refresh mode.
2. Clear SDRAM0_MCSTS[SRMS].
3. Ready to service any memory request (after a minimum of 200 cycles following CKE assertion).

Figure 18-40. Self-Refresh Exit



18.8.4.2 Hardware-Initiated Refresh Operation

Entry

Self-refresh entry can be hardware-initiated by asserting the MEMSELFREF input. When CHIP_RESET is inactive, sampling the asserted MEMSELFREF input causes the SDRAM controller to perform the following:

1. Completes the current request (if any). Data for the current or any previously queued access may not be returned or stored properly. Only valid data is written to memory. The assertion of MEMSELFREF during a PLB fixed length burst write-to-memory may result in a partial write.
2. Performs precharge all to close all open pages.
3. Performs an auto-refresh cycle.
4. Enters self-refresh mode.
5. Sets SDRAM_MCSTS[SRMS]

Independent of any pending memory access requests, the SDRAM controller maintains the SDRAM in self-refresh mode until MCOPT2[SRE] is cleared. Once SRDRAM0_CFGI[SRE] is cleared, the SDRAM controller performs the following:

Exit

1. Exit self-refresh mode.
2. Clear MCSTAT[SRMS].
3. Ready to service any memory request (after a minimum of 200 cycles following CKE assertion).

Limitations

The hardware-initiated self-refresh sequence can be initiated during normal operation or immediately following the de-assertion of CHIP_RESET. If CHIP_RESET is asserted during this sequence, the SDRAM controller, along with MCSTS[SRMS] is reset. If asserted MEMSELFREF is sampled during RESET, the SDRAM controller re-initiates the above sequence upon RESET de-assertion. The SDRAM controller does not have to be enabled to respond to MEMSELFREF. During RESET, MEMCKE is driven to a 0. An external system agent must take control of the MEMCKE signal and maintain it low during the reset of the SDRAM Controller to ensure that the SDRAM device is maintained in self-refresh mode. CHIP_RESET and MEMSELFREF may de-assert simultaneously. The SDRAM controller will "save state" and remain in self-refresh mode until SDRAM0_CFGI[SRE] is cleared.

The following table details the specific behavior of the SDRAM controller in response to the CHIP_RESET and MEMSELFREF inputs.

Table 18-12. SDRAM Controller Reset/Self-Refresh Response

Note	Current State				Next State					
	[SRE]		RESET Input	SELFRO Input	SDRAM State		Memory Controller Output			
	Enter	Exit			q(t)	q(t+1)	CKE	Control	Address	Data
1, 6	x	x	1	x	x	idle	0	nop	valid	valid
3, 8, 7	x	x	0	1	~self	self	0	nop	valid	valid
	0	x	0	0	~self	~self	1	x	valid	valid
1, 2, 7, 8	x	x	1	x	~self	self	0	nop	valid	valid
8, 10	x	0	0	0	self	self	0	nop	valid	valid
4, 5	0	1	0	0	self	idle	1	nop	valid	valid
8	x	x	0	1	self	self	0	nop	valid	valid
3, 8	1	0	0	0	~self	self	0	nop	valid	valid
12	1	1	0	0	x	x	x	x	x	x

RESET input is CHIP_RESET; SELFRQ input is MEMSELFREF and [SRE] is SDRAMO.CFGI.

1. This state change is an asynchronous event.
2. If MEMSELFREF is asserted and the SDRAM is not in the idle state, bus contention on the data bus may occur during reset and thereafter until MEMSELFREF is de-asserted. This only occurs if there is a loss of power to the SDRAM device so that the previously entered self-refresh mode state was lost.
3. The SDRAM controller completes the current PLB read/write request (ignoring any other requests) issues a precharge all command (all open pages closed), performs 1 auto refresh cycle, and places the memory in self-refresh mode.
4. Upon self-refresh exit, an auto refresh command is issued to resynchronize the internal refresh logic.
5. The auto refresh timer (RTR) must be programmed before CFGI[SRE] is cleared when exiting a hardware initiated self-refresh sequence during which the MCDDR core was reset.
6. After the transition to the idle state, the SDRAM controller must be fully initialized before setting CFGO[DCEN] to enable the SDRAM controller. See the Initialization Sequence sub-section of the SDRAM section for details of the SDRAM response to setting CFGO[DCEN].
7. To guarantee self-refresh entry, MEMSELFREF must remain asserted until MEMCKE is de-asserted or for xx clocks if CHIP_RESET is asserted.
8. While in the self-refresh state, all SDRAM controller memory access requests are ignored at the PLB interface until CFGI[SRE] is cleared. CFGO[DCEN] must be set, following configuration setup, to enable read/write access to memory.
9. The SDRAM controller can issue the Mode Register Set (MRS) command at any time provided that the requirements defined in the DCR section of this document are followed. The IOP can query the status of the MRS command request via MCSTAT[MRSC].

10. CHIP_RESET and MEMSELFREF de-assert simultaneously. The SDRAM controller will "save state" and remain in self-refresh mode until CFGI[SRE] is cleared.
11. A "1" in the "Enter" column indicates the setting of CFGI[SRE]. A "1" in the "Exit" column indicates the resetting (to "0" once set) of CFGI[SRE].
12. This state is invalid.

18.8.5 Mode Register Write Commands

The mode register write commands are issued during initialization to configure the DDR SDRAM operating mode. A total of three mode register write commands are issued. The first command (EMRS) targets the extend mode register and is used to configure the supported DDR SDRAM device-specific options. The supported DDR SDRAM device-specific options are selected using configuration the DDR SDRAM Device Options (SDRAM0_DEVOPT) register. SDRAM0_DEVOPT[0] configures the DLL option and SDRAM0_DEVOPT[1] configures the device drive strength. The second command (MRS1) targets the base mode register and configures the device access parameters and resets the device DLL. The third command (MRS2) targets the base mode register and enables normal operation.

The base mode set command vector consists of the following fields:

- BA[1:0]: Select base mode register
- MA[12:7]: Operating mode
- MA[6:4]: CAS Latency - configured by SDRAM0_TR0[SDCL] register bits for a $\overline{\text{CAS}}$ latency of 2, 2.5, or 3.
- MA[3]: Burst type - hard coded to 0 to select sequential wrap addressing.
- MA[2:0]: Burst length - hard coded to 0b010 to configure mode set for a burst of 4.

Table 18-12 Mode Set Command Vectors on page 354 illustrates the various mode set command vectors.

Table 18-12. Mode Set Command Vectors

CMD	CL	BA1	BA0	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
EMRS		0	1	0	0	0	0	0	0	0	0	0	0	DS	DLL
MRS1	2	0	0	0	0	0	1	0	0	1	0	0	0	1	0
	2.5	0	0	0	0	0	1	0	1	1	0	0	0	1	0
	3	0	0	0	0	0	1	0	0	1	1	0	0	1	0
MRS2	2	0	0	0	0	0	0	0	0	1	0	0	0	1	0
	2.5	0	0	0	0	0	0	0	1	1	0	0	0	1	0
	3	0	0	0	0	0	0	0	0	1	1	0	0	1	0

After the SDRAM operating mode has been configured, normal memory accesses can proceed. The mode set command vector is placed on the external memory address bus in the corresponding mode register write command window.

18.9 Registered DIMM Support

Registered interface operation is supported and enabled using SDRAM0_CFG0[RDEN]. The CAS latency setting should be programmed based solely on the DDR SDRAM device CAS latency. Thus, a DDR SDRAM device CAS latency of 2 clocks (SDRAM0_TR0[SDCL] = 2'b01) corresponds to a DIMM CAS latency of 3 clocks, and a DDR SDRAM device CAS latency of 3 clocks (SDRAM0_TR0[SDCL] = 2'b11) corresponds to a DIMM CAS latency of 4 clocks.

18.10 Error Checking and Correction (ECC)

The DDR SDRAM controller supports ECC by enabling SDRAM0_CFG0[2:3]. The ECC block provides check bit generation on PLB-to-memory writes and checking/correction on PLB-to-memory reads. The ECC uses a dual 32-bit or 64-bit single-bit-error-correction/double-bit-error-detection (SEC/DED) code depending on the memory interface width. In 64-bit mode, the DDR SDRAM controller implements a 64-bit data/8-bit check interface on the memory. In 32-bit mode, the DDR SDRAM controller implements a dual 32-bit data/8-bit check interface on the memory. In dual 32-bit mode, the ECC (generation, check and correction) is performed on each 32 bit access to the memory allowing for SEC/DED coverage within each 32-bit word.

18.10.1 Read-Modify-Writes

Any single beat PLB write cycle to memory with byte enables that are not divisible by an ECC word (4 Byte and aligned for 32-bit mode and 8 Byte and aligned for 64-bit mode) will trigger the ECC block to generate a read cycle to fetch the appropriate quad word, modify that quad word, generate the ECC bits and write the quad word back out to memory.

Specifically, in 32-bit mode, PLB single beat write cycles with the following byte enables (16 bits each) will not trigger a read-modify-write:

0000_0000_0000_1111, 0000_0000_1111_0000, 0000_0000_1111_1111,
0000_1111_0000_0000, 0000_1111_0000_1111, 0000_1111_1111_0000,
0000_1111_1111_1111, 1111_0000_0000_0000, 1111_0000_0000_1111,
1111_0000_1111_0000, 1111_0000_1111_1111, 1111_1111_0000_0000,
1111_1111_0000_1111, 1111_1111_1111_0000, 1111_1111_1111_1111

Specifically, in 64-bit mode, PLB single beat writes cycles with the following byte enables (16 bits each) will not trigger a read-modify-write:

0000_0000_1111_1111, 1111_1111_0000_0000, 1111_1111_1111_1111

The ECC features are described in *Table 18-13*.

Table 18-13. ECC Features

Feature	Explanation
Standard SEC/DED coverage	The ECC module corrects all single bit errors and detects all double bit errors when reading from memory.
Aligned nibble error detect	The ECC module detects any and all errors which may exist in an aligned four bit nibble.
Checking and Correction Disable	ECC error correction may be disabled. Mixed use of ECC and non-ECC banks is not supported.

18.10.2 ECC Timing

The effect of ECC on read and write accesses is shown in *Table 18-14*:

Table 18-14. Effect of ECC on Timing

PLB Transaction	Added Latency	Comment
ECC enabled: Read	0/1 Clock	As defined at 0, Read Data is registered prior to going through ECC tree. Actual added latency is programmable and depends on SDRAM0_TR1T settings, operating frequency, and chip/system level timings. As it is defined, it equals 0.
ECC enabled: Burst or full single beat write	None	
ECC enabled: Partial Writes requiring read-modify-write	Read Latency + 2 clocks	On partial writes, a read-modify-write sequence, including bus turn around, is required to correctly generate the write check bits and store the resultant data.

18.10.3 ECC Configuration Register

The ECC module uses only one configuration register to control its operation (see*Memory Controller Options 0 (SDRAM0_CFG0)* on page 321).

18.10.4 ECC Error Register

After an ECC error has occurred, the ECC Error register must be reset in order to enable it to record future errors. Reset this register with a write of 0xFFFFFFFF.

18.10.5 Error Detection and Error Handling

The DDR SDRAM controller detects, reports, and logs errors and error status for PLB-to-memory accesses and enables "locking" of the SDRAM0_BESR and SDRAM0_BEAR of the master for errors reported on the PLB. When locked, the error registers remain locked until software clears the associated master lock error status bits in SDRAM0_BESR0/SDRAM0_BESR1.

18.10.6 Memory Read and Write Errors

There are two general types of ECC errors, single bit correctable, and multi-bit uncorrectable. ECC-related errors are detected and logged for DDR SDRAM PLB-to-memory accesses. ECC must be globally enabled, using SDRAM0_CFG0, and ECC error checking and correction must be enabled using SDRAM0_CFG0[3].

When ECC is disabled, or if ECC is enabled and ECC error checking and correction is disabled, no memory access error checking occurs. Specific error types, information logged, and error responses are detailed in the following sections.

18.10.7 Uncorrectable ECC Error on Memory Read

An uncorrectable error detected during a PLB-to-memory read causes the data being returned from system memory (unchanged) to be transferred on the PLB with the associated ERROR signal. The address associated with the error is logged in the SDRAM0_BEAR. The error status for the associated master is logged in either the SDRAM0_BESR0 or SDRAM0_BESR1, depending on the PLB master ID. The ECC Error Status Register is updated to indicate that an Uncorrectable Error occurred with the chip select SDRAM0_ECCESR[BKnE]

associated with the logged error. The uncorrectable error interrupt signal will be asserted and remain asserted until the uncorrectable error bit is cleared in the SDRAM0_ECCESR Register. This level-sensed interrupt may be used to signal the Uncorrectable Error event to the system.

18.10.8 Uncorrectable ECC Error on Memory Partial Write

An Uncorrectable Error detected on the read portion of a read-modify-write sequence for a PLB-to-memory partial write results in the data returned from system memory (unchanged) being combined with the "write" data and written back to memory with new ECC check bits. The associated interrupt signal is asserted to indicate the error. This signal will remain asserted until the corresponding bit is cleared in the SDRAM0_MIRQ register. The address associated with the error is logged in the SDRAM0_BEAR. The error status for the associated master is logged in either SDRAM0_BESR0 or SDRAM0_BESR1, depending on the PLB master ID. The ECC Error Status Register is updated to indicate that an Uncorrectable Error occurred with the chip select SDRAM0_ECCESR[BKnE] associated with the logged error. The uncorrectable error interrupt signal will be asserted and remain asserted until the Uncorrectable Error bit is cleared in the SDRAM0_ECCESR Register. This level-sensed interrupt may be used to signal the Uncorrectable Error event to the system.

18.10.9 Correctable ECC Error on Memory Read

A Correctable Error detected on a DDR SDRAM PLB-to-memory read results in the corrected data returned from system memory being transferred to the requesting master device on the PLB. The ECC Error Status Register (SDRAM0_ECCESR) is updated to indicate that a Correctable Error occurred with the byte lane corrected SDRAM0_ECCESR[BnCE], check bit status (if applicable), and the chip select SDRAM0_ECCESR[BKnE] of the logged error. The correctable error interrupt signal will be asserted and remain asserted until the Correctable Error bit is cleared in the SDRAM0_ECCESR. This level-sensed interrupt may be used to signal the Correctable Error event to the system. The address of a single bit error is not saved. The data that contains the error in the memory system is not corrected in the memory system.

18.10.10 Correctable ECC Error on PLB Partial SDRAM Memory Write

A Correctable Error detected on the read portion of a read-modify-write sequence during a PLB-to-memory partial write (less than a double word) results in the corrected data returned from system memory being combined with the "write" data and written back to memory with new ECC check bits. The ECC Error Status Register is updated to indicate that a Correctable Error occurred with the byte lane corrected SDRAM0_ECCESR[BnCE], check bit status (if applicable), and the chip select SDRAM0_ECCESR[BKnE] of the logged error. The correctable error interrupt signal will be asserted and remain asserted until the Correctable Error bit is cleared in the SDRAM0_ECCESR. This level-sensed interrupt may be used to signal the Correctable Error event to the system.

18.10.11 ECC Diagnostics

The occurrence of a correctable single bit error or an uncorrectable multi-bit error can be simulated and the operation of the ECC verified through a series of memory accesses while varying the Memory Data error checking setting (SDRAM0_CFG0[MCHK]). The general sequence is as follows:

1. Initialize all DDR SDRAM related configuration registers.
2. Enable ECC generation only.
3. Enable the SDRAM controller.
4. Initialize (Write) the entire installed memory space to initialize the data and check bits.
5. Allow the DDR SDRAM to become idle.
6. Set the Memory Data error checking to "None."

7. Modify one bit (in the case of a CE) or two bits (in the case of a UE) at the address to be tested.
8. Allow the DDR SDRAM to become idle.
9. Set the Memory Data error checking to "ECC Checking and Correction."
10. Read the corresponding address.

In the case of simulating a Correctable Error, the corrected data should be returned to the system. In the case of simulating an Uncorrectable Error, the original data (with the two changed bits) should be returned to the system. In either case, the described error response corresponding to the error simulated should also occur.

18.10.12 ECC Code Matrix

The 32-bit mode ECC code matrix for word 0 is shown in *Table 18-15* and *Table 18-16*.

Table 18-15. 32-Bit Mode ECC Code Matrix for Word 0

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				X	
X				X				X				X				X	X	X	X	X
	X				X				X				X			X				X
		X				X				X				X			X			
			X				X				X				X			X	X	
				X	X	X	X					X	X	X	X					X
								X	X	X	X	X	X	X	X					
X	X	X	X									X	X	X	X	X	X	X		

Table 18-16. 32-Bit Mode ECC Code Matrix for Word 0 (continued)

21	22	23	24	25	26	27	28	29	30	31	UC0	UC1	UC2	UC3	UC4	UC5	UC6	UC7
		X				X				X	X							
X	X	X	X	X	X	X	X	X	X	X		X						
			X				X						X					
X				X				X						X				
	X	X			X	X			X	X					X			
X	X	X					X	X	X	X						X		
			X	X	X	X	X	X	X	X							X	
		X				X	X	X	X									X

The 32-bit mode ECC code matrix for word 1 is shown in *Table 18-17* and *Table 18-18*.

Table 18-17. 32-Bit Mode ECC Code Matrix for Word 1

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
		X				X				X				X					X
			X				X				X				X	X			
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		
X				X				X				X						X	
	X	X	X		X	X	X		X	X	X		X	X	X				
				X	X	X	X					X	X	X	X	X	X	X	X
								X	X	X	X	X	X	X	X	X	X	X	X
X	X					X	X			X	X	X	X						

Table 18-18. 32-Bit Mode ECC Code Matrix for Word 1 (continued)

52	53	54	55	56	57	58	59	60	61	62	63	LC0	LC1	LC2	LC3	LC4	LC5	LC6	LC7
			X				X		X			X							
X				X						X			X						
	X				X						X			X					
		X				X		X	X	X	X				X				
								X	X	X	X					X			
X	X	X	X														X		
				X	X	X	X	X	X	X	X							X	
X	X	X	X	X	X	X	X		X	X	X								X

The 64-bit mode ECC code matrix is shown in *Table 18-19*. It is derived from the upper (UC 0:7) and lower (LC 0:7) check bits calculated for 32-bit mode.

Table 18-19. 64-Bit Mode ECC Code Matrix

UC0	UC1	UC2	UC3	UC4	UC5	UC6	UC7	LC0	LC1	LC2	LC3	LC4	LC5	LC6	LC7	C0	C1	C2	C3	C4	C5	C6	C7
x								x								x							
	x								x								x						
		x								x								x					
			x								x								x				
				x								x								x				
					x								x								x			
						x								x								x		
							x								x								x	
								x								x								x

18.11 Power Management

The DDR SDRAM controller implements Class II (Macro Paced Sleep) power management as defined by the *Clock and Power Management Interface Specification for Core + ASICs*. As such, the DDR SDRAM controller provides a single sleep request output which indicates that the controller is idle and may be put to sleep.

18.11.1 Sleep Mode Entry

The DDR SDRAM controller power management behavior is configurable using DCR registers SDRAM0_CFG1[PMEN], located at offset 0x21, and SDRAM0_PMIT[PM_C] located at offset 0x34. Setting the PMEN=1 bit enables power management logic to send a request to the Clock Power Management (CPM) unit during sleep once the controller has been idle for the programmed (PM_C) number of controller clock cycles. With PMEN=0, DDR SDRAM controller power management is disabled.

18.11.2 Sleep Mode

In sleep mode, all DDR SDRAM controller clocking is disabled, with the exception of SDRAM refresh logic and power management WAKE logic. SDRAM refresh preserves the contents of the memory and maintains the refresh interval. Power management WAKE logic monitors master device requests on the PLB and DCR bus interfaces and initiates the WAKE-UP sequence when a valid master request PLB or DCR bus cycle is received.

18.11.3 Sleep Mode Exit

Once a valid master request, PLB request, or DCR request targeting the DDR SDRAM is registered, the controller deasserts its sleep request in the following clock cycle. Once sleep mode is exited, the DDR SDRAM begins the requested access in the following clock cycle.

18.12 System Memory Clock Concerns

The DDR memory clock signal found in the PPC440EP has the ability to drive a limited number of loads, two to three. In many board level designs, the DDR memory clock signal must be buffered before sending the signal out to the memory devices. An external differential clock buffer is used to generate individual copies of the differential memory clock. This external differential clock buffer must be a zero-delay (PLL-based) differential clock buffer wire delay and clock buffer-to-memory wire delay.

When using the external clock buffer in general, it is expected that the DDR memory clock is advanced by 90 degrees to help compensate for the on-chip differential clock driver delay. It is also expected that the on-chip generated PLB clock is used to clock the read data path sample stage 2.

19. Peripheral Component Interconnect (PCI) Interface

The peripheral component interconnect (PCI) interface and bridge (referred to asthe PCI bridge in this chapter) provides a means for connecting PCI-compatible devices to the on-chip bus architecture of the PPC440EP chip. The PCI bridge is designed to the *PCI Specification*, Version 2.2. The PCI bridge is bidirectional in that it allows PPC440EP PLB masters to access PCI targets off-chip. It also allows PCI masters to access PLB slave devices such as the SDRAM controller. The PCI bridge contains an arbiter that can optionally be used for host applications.

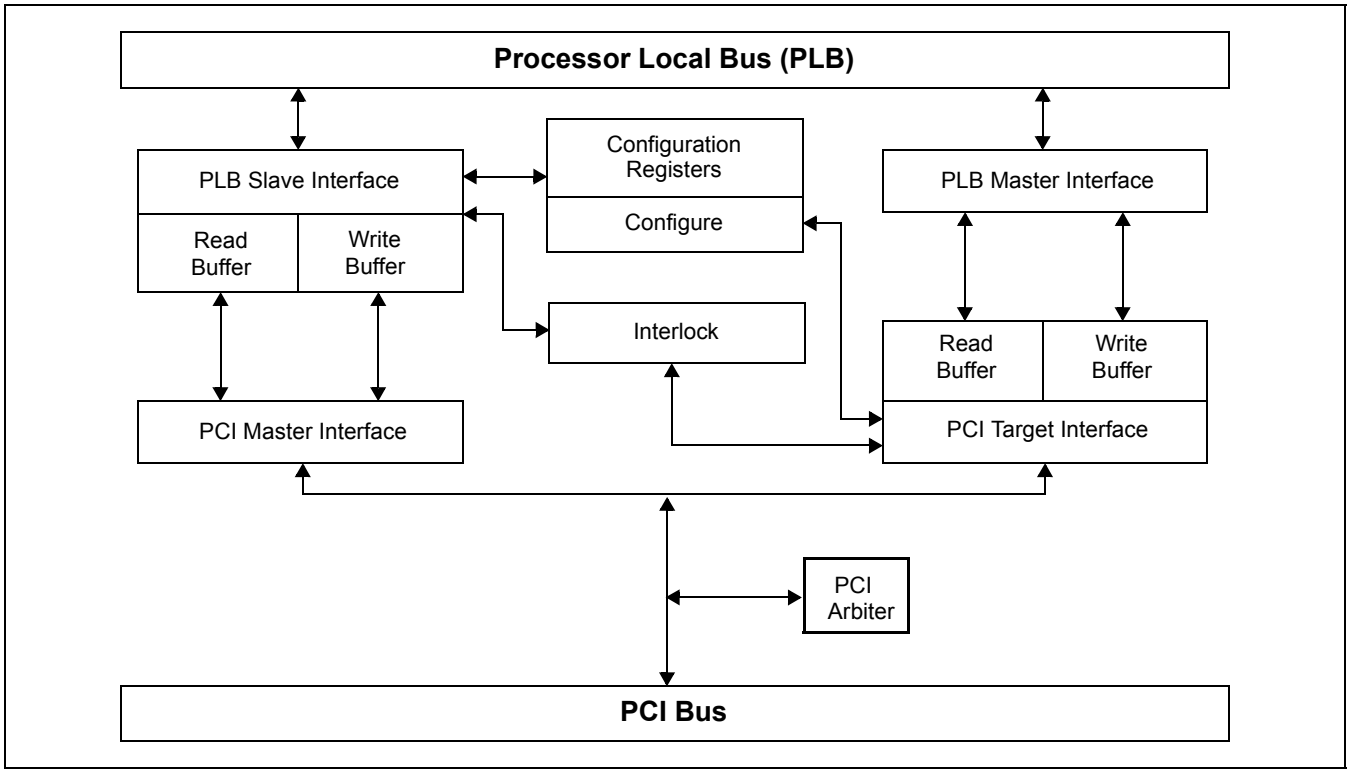
The PCI bridge can be used as the host bridge. The PCI bridge is also configurable by an external PCI agent, allowing it to be used in target adapter applications. The PCI bridge contains address mapping register sets to provide address mapping for both transaction directions (see *Figure 19-1 PCI Bridge Block Diagram*). Agents on the PLB are referred to as masters or slaves. Agents on the PCI are referred to as targets or masters.

19.1 Features

- PCI bus frequency up to 66 MHz (asynchronous)
- Asynchronous clocking between PLB and PCI buses (optional)
- Supports 1:1, 2:1, 3:1, and 4:1 clock ratios from PLB to PCI
- 32-bit PCI Address/Data Bus
- Power Management
- Buffering:
  - PCI target 64-byte write post buffer
  - PCI target 96-byte read prefetch buffer
  - PLB slave 32-byte write post buffer
  - PLB slave 64-byte read prefetch buffer
- Error tracking/status
- PCI arbitration function (optional)
- Supports PCI target-side configuration
- Supports processor access to all PCI address spaces:
  - Single-beat PCI I/O reads and writes
  - PCI memory single-beat and prefetch-burst reads and single-beat writes
  - Single-beat PCI configuration reads and writes (type 0 and type 1)
  - PCI interrupt acknowledge

Figure 19-1 shows the PCI bridge block diagram.

Figure 19-1. PCI Bridge Block Diagram



19.2 Byte Ordering

The PCI bridge configuration register address space must be treated as little endian, as required by *PCI Specification*, Version 2.2. In most cases data memory areas in PCI address space will be configured and used in little endian format. To provide for this, PCI configuration space and memory map regions should be defined as little endian memory space by means of the corresponding entry in the 440EP CPU's MMU.

Byte ordering and management of little endian memory space from a PowerPC CPU point of view is described in detail in *Byte Ordering* in the *PPC440 Processor User's Manual*. PowerPC architecture and CoreConnect bus architecture both use a bit naming convention in which the most significant bit (msb) name incorporates the numeral 0 and the least significant bit (lsb name for a 32-bit vector incorporates the numeral 31. *Table 19-1* shows the correspondence of address bit-naming conventions for PowerPC, CoreConnect PLB, and PCI interface.

Table 19-1. PowerPC, CoreConnect PLB, and PCI Address Bit-Naming Conventions

Functional Unit/Interface	Word Address	Byte Address
PPC440EP Processor Core Address	A0:29	A30:31
CoreConnect — PLB Address Bus	PLB_ABus0:29	PLB_ABus30:31
PCI Address Bus	AD31:2	AD1:0

Table 19-2 shows the correspondence of data bus bit naming conventions and data lane connections for PowerPC, CoreConnect PLB, and PCI interface. Note that within a data lane (column), the data signal naming indicates that, for example, AD31is connected to PLB Write Data24.

Table 19-2. PowerPC, CoreConnect PLB, and PCI Data Bus Bit-Naming Conventions

Functional Unit/ Interface	Most Significant Byte (MSB)	↔	↔	Least Significant Byte (LSB)
Data Byte Value (0x*nn*)	11	22	33	44
Little Endian Byte Address (0b*nn*)	11	10	01	00
PPC440EP Processor Core (Write) Data Bus	Data24:31	Data16:23	Data8:15	Data0:7
CoreConnect — PLB Write Data Bus — Byte Group	PLB Write Data24:31	PLB Write Data16:23	PLB Write Data8:15	PLB Write Data0:7
PLB Byte Enable	PLB_BE3	PLB_BE2	PLB_BE1	PLB_BE0
PCI Byte Enable	C/BE3	C/BE2	C/BE1	C/BE0
PCI Data Bus — Byte Group	AD31:24	AD23:16	AD15:8	AD7:0

1. Logical data work (32-bit word) == 0x11223344
2. 440 CPU performing either:
   - Store word to little endian memory space
   - Store word—byte reversed—to big endian address space

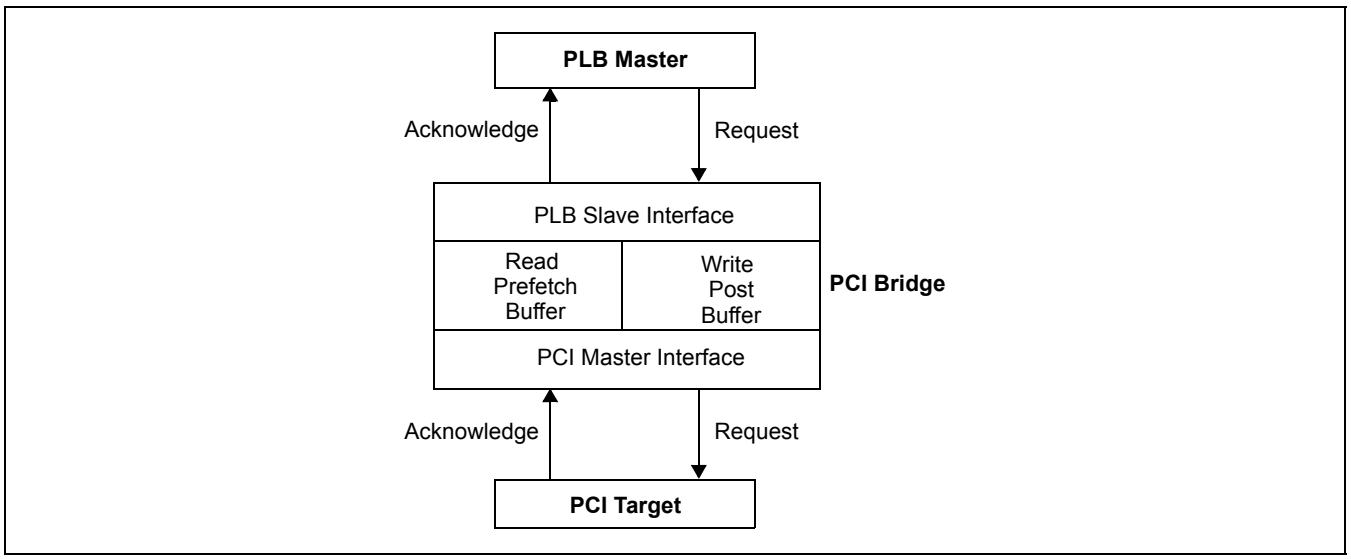
19.3 PCI Bridge Functional Blocks

The following sections describe the PCI bridges and the associated arbiter.

19.3.1 PLB-to-PCI Half-Bridge

As shown in *Figure 19-2*, the 64-bit PLB slave interface and PCI master interface function together as a PLB-to-PCI half-bridge to enable PLB master devices to access PCI target devices. The half-bridge configuration contains a 32-byte write post buffer and a 64-byte read prefetch buffer.

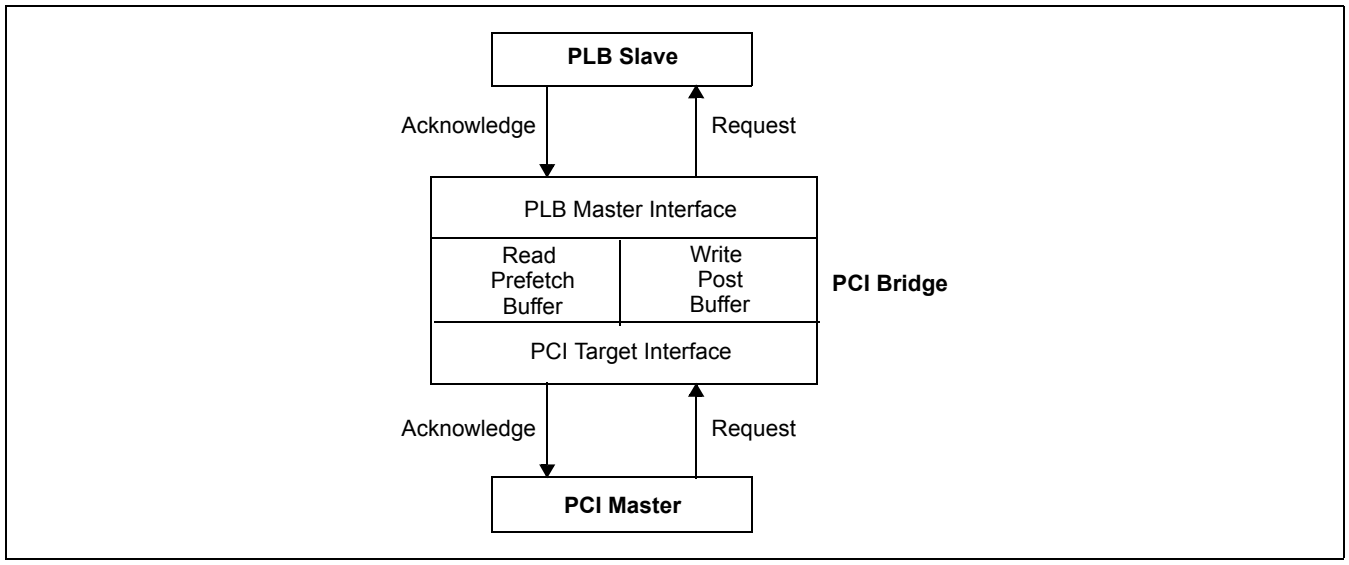
Figure 19-2. PLB-to-PCI Half-Bridge Block Diagram



19.3.2 PCI-to-PLB Half-Bridge

As shown in *Figure 19-3*, the PCI target interface and 64-bit PLB master interface function together as a PCI-to-PLB half-bridge to enable PCI master devices to access PLB slave devices. The half-bridge configuration contains a 64-byte write post buffer and a 96-byte read prefetch buffer.

Figure 19-3. PCI-to-PLB Half-Bridge Block Diagram

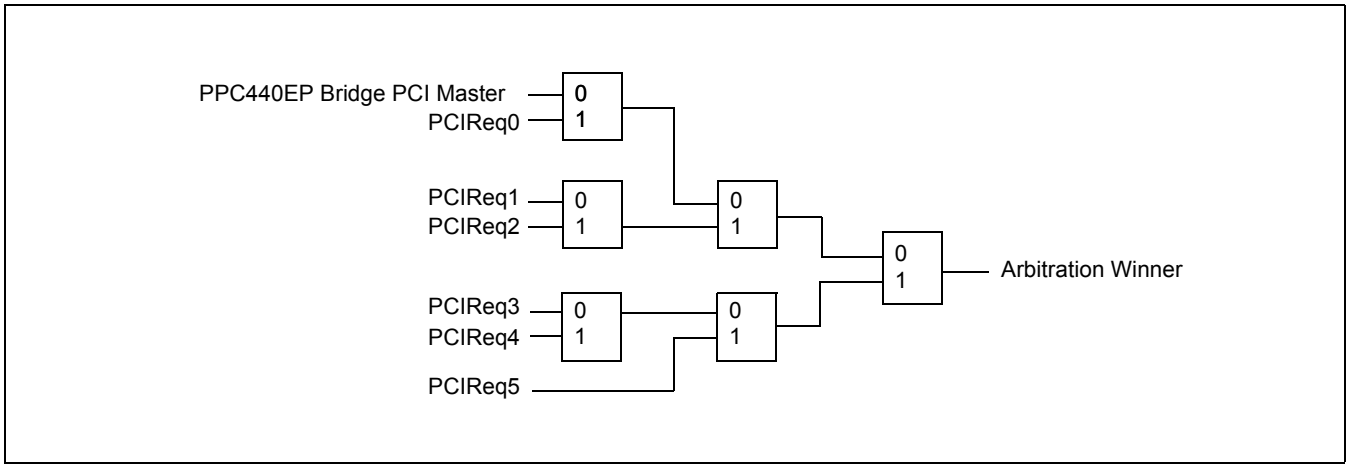


19.3.3 PCI Arbiter

The internal arbiter can be used with up to six external PCI masters ($\overline{Req}/\overline{Gnt}$ pairs) or can be disabled. When the internal arbiter is disabled, there is one $\overline{Req}/\overline{Gnt}$ pair that must be attached to an external arbiter. A strapping configuration pin determines whether the internal arbiter is enabled or not. Priority is round-robin (rotating). Priority switches when a master begins a transfer by asserting $\overline{Frame}$. Each block keeps a priority bit that only switches if its highest priority requestor receives a grant. Assuming that all priority bits are initially cleared and all requests are active, an example rotation would be 440EP 2, 1, 4. *PCI Specification*, Version 2.2 requires that all PCI devices three-state their pins during reset. The 440EP PCI arbiter supports bus parking during normal operation.

Figure 19-4 shows the logical arbitration structure.

Figure 19-4. Arbitration Structure



19.4 PCI Bridge Address Mapping

The following sections describe the address maps supported by the PCI bridge.

19.4.1 PLB-to-PCI Address Mapping

The PCI bridge responds as a slave on the PLB bus in several address ranges. These ranges enable a PLB master to configure the PCI bridge, and to cause the PCI bridge to generate memory, I/O, configuration, interrupt acknowledge, and special cycles to the PCI bus. *Table 19-3* shows the address map from the view of the PLB, that is, as decoded by the PCI bridge as a PLB slave.

Table 19-3. PLB Address Map

PLB Address Range	Description	PCI Address Range
0xE8000000–0xE800FFFF	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 0 to 64KB – 1.	0x00000000–0x0000FFFF
0xE8010000–0xE87FFFFF	Reserved PCI bridge does not respond. (Other bridges use this space for non-contiguous I/O.)	
0xE8800000–0xEBFFFFFF	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 8MB to 64MB – 1.	0x00800000–0x03FFFFFF
0xEC000000–0xEEBFFFFF	Reserved PCI bridge does not respond	
0xEEC00000–0xEECFFFFF	PCIC0_CFGADDR and PCIC0_CFGDATA 0xEEC00000: PCIC0_CFGADDR 0xEEC00004: PCIC0_CFGDATA 0xEEC00008–0xEECFFFFF: Reserved (can mirror PCIC0_CFGADDR and PCIC0_CFGDATA).	
0xEED00000–0xEEDFFFFF	PCI Interrupt Acknowledge and Special Cycle 0xEED00000 read: Interrupt Acknowledge 0xEED00000 write: Special Cycle 0xEED00004–0xEEDFFFFF: Reserved (can mirror Interrupt Acknowledge and Special Cycle).	
0xEEE00000–0xEF3FFFFF	Reserved PCI bridge does not respond.	

Table 19-3. PLB Address Map (continued)

PLB Address Range	Description	PCI Address Range
0xEF400000–0xEF4FFFFF	PCI Bridge Local Configuration Registers 0xEF400000: PCIL0_PMM0LA 0xEF400004: PCIL0_PMM0MA 0xEF400008: PCIL0_PMM0PCILA 0xEF40000C: PCIL0_PMM0PCIHA 0xEF400010: PCIL0_PMM1LA 0xEF400014: PCIL0_PMM1MA 0xEF400018: PCIL0_PMM1PCILA 0xEF40001C: PCIL0_PMM1PCIHA 0xEF400020: PCIL0_PMM2LA 0xEF400024: PCIL0_PMM2MA 0xEF400028: PCIL0_PMM2PCILA 0xEF40002C: PCIL0_PMM2PCIHA 0xEF400030: PCIL0_PTM1MS 0xEF400034: PCIL0_PTM1LA 0xEF400038: PCIL0_PTM2MS 0xEF40003C: PCIL0_PTM2LA 0xF400040–0xEF4FFFFF: Reserved (can mirror PCI local registers)	
0x00000000–0xFFFFFFFF (Note 1)	PCI Memory—Range 0 PMM 0 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	0x0000000000000000– 0xFFFFFFFFFFFFFFFF
0x00000000–0xFFFFFFFF*	PCI Memory—Range 1 PMM 1 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	0x0000000000000000– 0xFFFFFFFFFFFFFFFF
0x00000000–0xFFFFFFFF*	PCI Memory—Range 2 PMM 2 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	0x0000000000000000– 0xFFFFFFFFFFFFFFFF
Note 1: Memory map ranges are fully programmable. The ranges must not overlap with each other or conflict with any other memory mappings.		

Three PCI bridge address ranges, associated with PLB masters in PLB space, are mapped to PCI memory space: PCI master map (PMM) 0, PMM1, and PMM2.

Each PMM is configured using the following registers (*n* is 0, 1, or 2, corresponding with PMM0, PMM1, and PMM2, respectively):

- PMMnLocal Address (PCIL0_PMMnLA)
- PMMnMask/Attribute (PCIL0_PMMnMA)
- PMMnPCI Low Address (PCIL0_PMMnPCILA)
- PMMnPCI High Address (PCIL0_PMMnPCIHA)

The location of each PMM in PLB space is programmable, using the PCIL0_PMMnLA registers. The PLB address range assigned to each PMM should not overlap any other PLB address space range that is used or reserved. Overlapping results in undefined behavior.

The range of PCI memory address space associated with each PMM is also programmable, and is a 64-bit address space to enable address translation between the PCI bus and the PLB. The PCIL0_PMMnPCILA registers contain the low-order word of a PCI address; the PCIL0_PMMnPCIHA registers contain the high-order word of a PCI address. If the high-order word of a PCI address is greater than 0, the PCI bridge generates dual address cycles to the PCI.

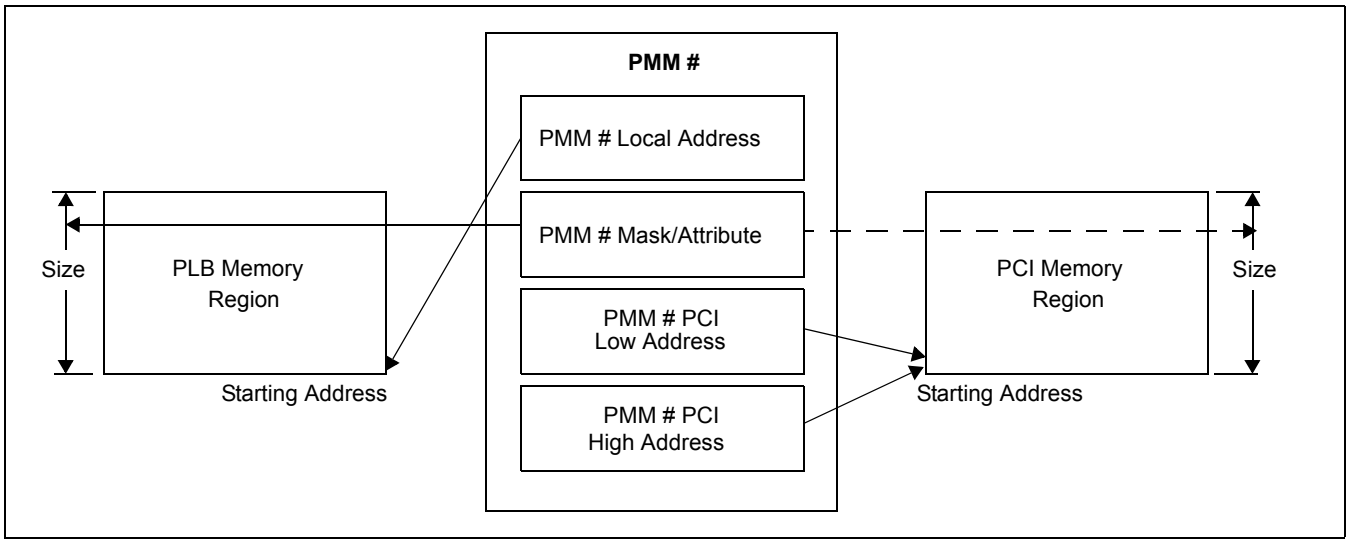
The size of each PMM is programmable, using the mask portion of the PCIL0_PMMnMA registers. The size is a power of 2, ranging from 4KB–4GB. The PLB and PCI address spaces for each PMM are aligned to the size contained in the associated PCIL0_PMMnMA registers.

The attribute portion of the PCIL0_PMMnMA registers specify whether the associated PMM is enabled or disabled, and marked as prefetchable or not prefetchable.

Address ranges and attributes should be initialized before a PMM is enabled.

Figure 19-5 shows the detail of the PMM register sets used to map PLB memory regions to PCI address space.

Figure 19-5. PMM Register Sets Map PLB Address Space to PCI Address Space



19.4.2 PCI-to-PLB Address Mapping

The PCI bridge responds as a PCI target for memory accesses and configuration Type 0 accesses. *Table 19-4* shows the PCI memory address map from the view of PCI, that is, as decoded by the PCI bridge as a PCI target.

Table 19-4. PCI Memory Address Map

PCI Memory Address	Description	PLB Address
0x00000000–0xFFFFFFFF	System Memory or ROM—Range 0 PTM 1 maps a region of PCI memory space to PLB space, which can map to system memory or ROM. Size and location is programmable. The space supports address translation between the PCI and the PLB.	0x00000000–0xFFFFFFFF
0x00000000–0xFFFFFFFF	System Memory or ROM—Range 1 PTM 2 maps a region of PCI memory space to PLB space, which can map to system memory or ROM. Size and location is programmable. The space supports address translation between the PCI and the PLB.	0x00000000–0xFFFFFFFF

19.4.3 PCI Target Map Configuration

Two PCI bridge address ranges in PCI memory space are mapped to PLB space: PCI target map (PTM1) and PTM2 (PTM0 is reserved).

Each PTM is configured using the following registers (n is 1 or 2, corresponding with PTM1 and PTM2, respectively).

- PTMnMemory Size (PCIL0_PTMnMS)
- PTMnLocal Address (PCIL0_PTMnLA)
- PTMnBAR (PCIC0_PTMnBAR)

The size of each PTM is programmable, using the PCIL0_PTMnMS registers. The size is a power of 2, and ranges from 4KB–4GB. The PLB and PCI address spaces for each PTM are aligned to this size.

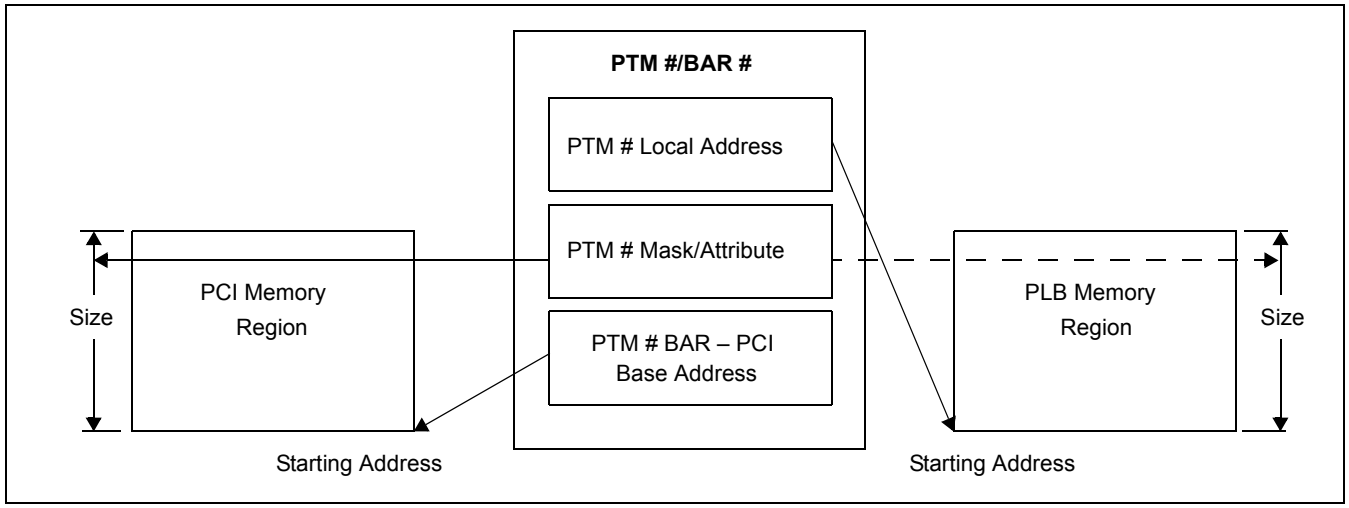
The address range of PLB space accessed through each PTM is also programmable, enabling address translation between the PCI bus and the PLB. The PLB address range is defined in the PCIL0_PTMnLA registers.

The location of each PTM in PCI memory space is programmable, using the PCIC0_PTMnBAR registers.

The PTMs are enabled and disabled using PCIC0_CMD[MA]. PTM address ranges and sizes should be initialized before being enabled. If the PCI bridge is not the host bridge, the local processor must initialize the PTM size before enabling host configuration setting the Host Configuration Enable (HCE) field of the Bridge Options 2 register (PCIC0_BRDGOPT2). This ensures that the host experiences proper behavior from the PCIL0_PTMnBAR registers. Note that PTM1 is always enabled. The PTM1 registers must always be initialized.

Figure 19-6 shows the detail of the PMM/BAR register sets used to map PCI memory regions to PLB address space.

Figure 19-6. PTM Register Sets Map PCI Address Space to PLB Address Space



19.5 PCI Bridge Transaction Handling

The following sections discuss PCI bridge transactions and completion ordering.

19.5.1 PLB-to-PCI Transaction Handling

This section describes how the PCI bridge responds to read and write requests from a PLB master. The PCI bridge decodes and accepts PLB transactions to different address ranges resulting in the generation of memory, I/O, configuration, interrupt acknowledge and special cycles on the PCI bus.

Table 19-5. Transaction Mapping: PLB —> PCI

PLB Transaction PLB Master → Bridge (PLB Slave Interface)	Bridge Mapping and Qualifications	PCI Transaction Bridge (PCI Master Interface) → PCI Target
Single-beat 1 → 8-byte Read	64KB or 56MB PCI I/O address range	I/O Read
Single-beat 1 → 8-byte Write	64KB or 56MB PCI I/O address range	I/O Write
Single-Beat 1 → 8-byte Read	Access to PCIC0_CFGDATA register	Configuration Read (Type 0, 1)
Single-Beat 1 → 8-byte Write	Access to PCIC0_CFGDATA register	Configuration Write (Type 0, 1)
Single-Beat 1 → 8-byte Read	PLB address decodes to PMM0, PMM1, or PMM2, nonprefetchable	Memory Read
Burst Read	PLB address decodes to PMM0, PMM1, or PMM2, nonprefetchable	Memory Read
PLB 4-word and 8-word Line Reads	PLB address decodes to PMM0, PMM1, or PMM2	Memory Read Line
Single-Beat 1 → 4-byte Read	PLB address decodes to PMM0, PMM1, or PMM2, prefetchable	Memory Read Multiple
Burst Read	PLB address decodes to PMM0, PMM1, or PMM2, prefetchable	Memory Read Multiple
Single-Beat 1 → 4-byte Write	PLB address decodes to PMM0, PMM1, or PMM2	Memory Write
Burst Write	PLB address decodes to PMM0, PMM1, or PMM2	Memory Write
Single-Beat 1 → 4-byte Read	Address 0xEED00000	Interrupt Acknowledge
Single-Beat 1 → 4-byte Write	Address 0xEED00000	Special Cycle
—	Not supported	Memory Write and Invalidate
—	Not supported	Memory Write Line

19.5.1.1 PCI Master Commands

The type of cycle generated to the PCI bus depends on the PLB address, the type of PLB transfer, and the data size. The following sections describe the transaction types supported and outlines the translation of commands from one bus to the other.

The terms "single beat" or "1–8-byte," in reference to PLB transfers, refer to the M_size=0000 transaction type.

PCI bridge initiates the following commands as a PCI master:

- I/O Read and I/O Write
  This command is generated in response to PLB 1–8-byte read or write requests that decode to one of the two PCI I/O spaces.

- Configuration Read and Configuration Write (type 0 and type 1)
  This command is generated in response to PLB 1–8-byte read or write requests that decode to the PCIC0_CFGDATA register.
- Memory Read
  This command is generated in response to PLB 1–8-byte reads or byte and half word burst reads that decode to one of the three PMMs when the PMM is marked as nonprefetchable.
- Memory Read Line
  This command is generated in response to PLB 4- and 8-word line reads or word and double word reads of 32 bytes or less that decode to one of the three PMMs.
- Memory Read Multiple
  This command is generated in response to PLB 1–8-byte reads or byte and half word burst reads that decode to one of the three PMMs when the PMM is marked as prefetchable. This command is also generated in response to word and double word burst reads of greater than 32 bytes that decode to one of the three PMMs. For prefetches, the PCI bridge bursts up to a 64 bytes from the PCI.
- Memory Write
  This command is generated in response to PLB 1–8-byte writes or burst writes to one of the three PMMs.
- Interrupt Acknowledge
  This command is generated in response to a PLB 1–8-byte read from address 0xEED00000.
- Special Cycle
  This command is generated in response to a PLB 1–8-byte write to address 0xEED00000.

The Memory Write and Invalidate command is not generated. All PCI memory writes are performed with Memory Write.

The PLB slave responds as a 64-bit device to word and double word bursts. All other commands receive a 32-bit response.

The PCI bridge supports PLB size 1–8-byte encodings. Burst reads of all sizes are also supported. Read line sizes greater than eight words are not supported, and no line writes are supported. The PCI bridge posts all writes which are decoded to PCI memory and PCI I/O space. Posted data is kept in internal write buffers until it can be transferred to the PCI bus. All other writes and all reads are connected, that is, they complete on the PCI bus before completing on the PLB.

19.5.1.2 PLB Slave Read Handling

PLB master read requests are decoded into four types: PCI Memory, I/O, Configuration, and Interrupt Acknowledge. If the request falls within any of these ranges, and is a supported command type, the bridge claims the cycle initially by asserting a PLB wait signal (as opposed to a PLB address acknowledge signal). The bridge must first gain access to the PCI bus before acknowledging a PLB read request. The specific timing of the address acknowledge is dependent upon the type of transfer. All posted writes must be flushed before a read is allowed to complete.

For PLB line reads, the PCI bridge must wait for all read data to be received before acknowledging the PLB request. This is because PCI targets are allowed to disconnect in the middle of a transfer, and the PLB requires line transfers to be atomic. If the system can guarantee that PCI targets do not disconnect these reads, PCIC0_BRDGOPT1[APLRM] can be set to 1. In this mode, line read performance is improved by having the bridge PLB slave assert an address acknowledge signal and begin its data tenure as soon as the first word is received on the PCI bus. If the above guarantee cannot be made, the setting of this bit could hang the bridge.

If the PCI cycle Master Aborts, all beats of read data are returned as 0xFFFFFFFF.

PLB master reads to the PCI bridge configuration registers are allowed to execute regardless of whether any write data is posted in the bridge. The configuration registers are described in *PCI Bridge Configuration Registers* on page 379.

19.5.1.3 Prefetching

When the PCI bridge receives a PLB 1–8-byte or word or double word burst read request that decodes to a PMM marked as nonprefetchable. The PCI bridge runs a single beat read to the PCI. If the PCI cycle is retried, the PLB cycle is rearbitrated.

When the PCI bridge receives a PLB 1–8-byte read request that decodes to a PMM marked as prefetchable, the PCI bridge burst reads up to a 64 bytes from the PCI and saves the data in the PLB slave read prefetch buffer. Less than 64 bytes can be read if the PCI target disconnects, or if the PCI bridge PCI master disconnects due to a master latency time out. Note that PCI bridge prefetching is not affected by memory management page boundaries (PLB_Guarded is ignored). If a subsequent PLB 1–8-byte or byte or half word burst read is contained in the prefetch buffer, the data is returned to the PLB directly from the prefetch buffer, and no cycle is generated on the PCI.

If a PLB read to the PCI bridge occurs while the PCI bridge is prefetching and does not hit in the prefetch buffer, then the PLB read is rearbitrated. After prefetching completes, any PLB read (of any type or address range) to the PCI bridge that does not hit in the prefetch buffer causes the prefetch buffer to be emptied, and a new PCI read to begin. PLB writes, including configuration writes, will invalidate the prefetch buffer.

19.5.1.4 PLB Slave Write Handling

PLB master write requests are decoded into four types: PCI Memory (one of three PMM ranges), PCI I/O, PCI Configuration, or Special Cycles. If the request falls within any of these ranges, and is a supported command type, the bridge claims the cycle by asserting a PLB wait signal. If the write is connected, or translates to a PCI Configuration or Special Cycle, the bridge must gain access to the PCI bus and successfully transfer the data before it may assert a PLB address acknowledge signal. If the address is to PCI I/O or memory, the bridge immediately asserts a PLB address acknowledge signal and posts the write if there is sufficient buffer space.

Internal configuration writes are not allowed to execute if posted write data exists in either the PCI slave write buffer or the PLB slave write buffer. The internal configuration mechanism is described in *PCI Bridge Configuration Registers* on page 379.

PLB Slave Write Post Buffer

The PCI bridge has a 32-byte write post buffer that may contain four separate single-beat PLB write transactions or one burst. New PLB write requests are rearbitrated if there is not enough room in the write post buffer.

The buffers are not snooped, and are always completed on the PCI bus in the same order as they are received on the PLB bus.

Each write buffer entry preserves the master ID and drives the appropriate PLB bus busy signal until the write is deallocated (it completes on the PCI bus). It is recommended that PLB masters do not use PLB bus busy signal. Instead, PLB masters generating cycles to the PCI should use the standard PCI mechanisms for data coherency.

19.5.1.5 Aborted PLB Requests

The PCI bridge aborts PLB reads only.

A PLB master accessing the PCI bridge can abort PLB write cycles only under the following conditions:

- The PCI bridge rearbitrates the cycle.
- The PCI bridge does not see the cycle because the PLB bus is granted to some other master. A CPU/System Memory interface is expected to do this when a CPU cycle is pending to PCI bridge, but a PLB Master requests system memory access requiring snooping.

Note: If a PLB master aborts the write cycle at any other time, the results are undefined and the bus may hang.

19.5.1.6 Retried PCI Reads

The PCI specification requires that a PCI master must repeat any read that is retried. The PCI bridge adheres to this requirement. It is only mentioned here because, under certain conditions with respect to aborted PLB reads, the PCI bridge must execute a PCI read and discard the data.

19.5.2 PCI-to-PLB Transaction Handling

This section describes how PCI bridge handles read and write requests from a PCI master device. PCI bridge responds as a PCI target to PCI memory transactions when the PCI address is in one of the two PTM ranges and PCIC0_CMD[MA] = 1. PCI bridge responds by claiming the PCI cycle and mastering a cycle on the PLB.

PCI bridge is also a PCI target for configuration cycles when its PCIIDSel pin is active. PCI bridge will master abort if a configuration cycle is run to itself.

Table 19-6. Transaction Mapping: PCI → PLB

PCI Transaction PCI Master → Bridge (PCI Target Interface)	Bridge Mapping and Qualifications	PLB Transaction Bridge (PLB Master Interface) → PLB Slave
Single-Beat Memory Read	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	8-byte or double word burst read
Burst Memory Read	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	8-byte or double word burst read
Memory Read Line	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst read
Memory Read Multiple	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst read
Memory Read	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst read
Single-Beat Memory Write	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	1 → 8-byte write
Single-Beat Memory Write and Invalidate	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	1 → 8-byte write
Burst Memory Write	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst write
Burst Memory Write and Invalidate	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst write
—	Not supported	Memory line reads

Table 19-6. Transaction Mapping: PCI → PLB (continued)

PCI Transaction PCI Master → Bridge (PCI Target Interface)	Bridge Mapping and Qualifications	PLB Transaction Bridge (PLB Master Interface) → PLB Slave
—	Not supported	Memory line writes

19.5.2.1 PLB Master Commands

PCI bridge generates PLB transactions based on the type and length of received PCI transactions. The following sections describe the transaction types supported and outline the translation of commands from one bus to the other.

The term "single-beat" refers to the M_size = 0000 transaction type. PCI slave devices are referred to as "targets."

PCI bridge initiates the following PLB master commands:

- 8-Byte Read:
  Generated in response to single beat or burst Memory Read commands from the PCI bus.
- Double word Burst Read:
  Generated in response to Memory Read Line and Memory Read Multiple commands on the PCI bus. PCI bridge can also be programmed to perform double word bursts on behalf of Memory Read.
- 1–8-Byte Write:
  Generated in response to single-beat (1–4 byte) Memory Write commands on the PCI bus; also generated when the PCI master uses non-contiguous byte enables (see *Byte Enable Handling* on page 377).
- Double word Burst Write:
  Generated in response to burst Memory Write and Memory Write and Invalidate commands on the PCI bus.

The PLB treats Memory Write and Memory Write and Invalidate identically (nothing on the PLB distinguishes a Memory Write from a Memory Write and Invalidate.)

PCI bridge does not generate line reads or line writes on the PLB.

19.5.2.2 Handling of Reads from PCI Masters

PCI bridge responds to PCI Memory Read, Memory Read Line, and Memory Read Multiple commands. The PCI bridge initiates all PLB reads as single-beat or double word burst transfers.

Memory Read generates a PLB single-beat double word read. Memory Read Line and Memory Read Multiple commands generate PLB double word bursts. For Memory Read Line, PCI bridge encodes a burst length on the byte enable pins of the PLB that corresponds to the number of double words from the start address to the end of a word boundary and terminates when the encoded number of words has been transferred. This is called a PLB fixed length burst. If the starting address is the last double word on a word boundary (typically, this should not occur), PCI bridge executes a single-beat read. For Memory Read Multiple, PCI bridge sets the byte enables to 0s, indicating a variable length burst.

The PCI target can be programmed to treat Memory Reads as Memory Read Line commands or Memory Read Multiple commands in terms of PLB read behavior.

The PCI bridge guarantees the PCI initial target latency requirement by retrying the PCI cycle if read data is not immediately available in the read buffer. Subsequent latency is programmable using PCIC0_BRDGOPT2[STLD].

The PCI bridge master latency timer can limit the length of read bursts using PCIC0_BRDGOPT1[MLTC]. The timer limits the duration of a burst to the programmed value in units of PLB clock cycles.

Read Buffer

The PCI bridge read buffer stores all read data (including delayed read and prefetched data) when the data is received from the PLB, before it is passed to the PCI. The 96-byte read buffer can store one transaction.

Delayed Reads

A delayed read is queued if a PCI master requests a read while PCI master writes are posted. Posted writes are completed on the PLB before the read is run. PCI bridge continues to post PCI master writes (if buffer space is available) while a delayed read is in progress. Such writes complete on the PLB after the read, even though they complete on the PCI before the read.

A delayed read is also queued if data is not immediately available in the read buffer and a delayed read does not already exist.

When a PCI master returns for a previously requested delayed read, the data is passed out of the read buffer. While the PCI master accepts delayed read data, the PCI bridge can begin to prefetch more read data, if the PCI master posted write buffer is empty. See *Read Prefetching* on page 376 for more details.

Any data remaining in the read buffer after delayed read data has been passed to a PCI master is marked as prefetch data and discarded upon a write in either direction.

PCI bridge can hold one delayed read transaction. PCI bridge retries all other PCI master reads until the delayed read completes on the PCI. The read buffer discards data from a delayed read under only one condition. The PCI discard timer is used to track the amount of time it takes for a PCI master to re-request the read. If the PCI master does not re-request the read in 2^{15} PCI clocks (about one millisecond for a 33 MHz PCI clock), PCI bridge discards the delayed read data. This timer begins counting at the beginning of the initial PCI cycle (delayed read request). If PCI bridge is used in a system on which the PLB target (memory) maximum latency (including PLB arbitration) is a significant portion of the timer duration, the timer can expire despite normal bus operation. One solution to this problem is to disable the PCI Discard Timer.

If a delayed read is burst terminated on the PLB (a rare occurrence), PCI bridge will not repeat the request until the PCI master re-requests and only then if the PCI master requests more data than is already buffered.

Read Prefetching

PCI bridge attempts to prefetch data to maximize burst throughput on PCI read requests. Read prefetching occurs only in response to Memory Read Multiple commands or Memory Read, if the PCI target is programmed to treat them as Memory Read Multiple (Memory Read Line causes prefetching to the next word boundary only). This prefetch buffer is 96 bytes.

If a PCI master reads from the read buffer while a PLB read is in progress, data is passed to PCI as it is being filled from the PLB. If the read buffer goes empty long enough for the PCI subsequent latency timer to expire, the PCI is target disconnected. If the read buffer fills up, the PLB cycle is master terminated. The bridge PLB master will not attempt to reacquire the PLB bus if its posted write buffer is not empty.

Prefetched data is discarded if a write is accepted from either the PLB or the PCI. A PCI master read that misses the prefetch buffer also causes current read data to be discarded and the new request to be serviced.

Byte Enable Handling

PCI byte enables are treated as don't cares for PCI reads. The PCI bridge performs double word burst or single-beat doubleword reads on the PLB, regardless of the byte enables presented by the requesting PCI master.

Note: This rule assumes that all PLB memory is prefetchable and that all PLB memory accesses are nondestructive.

Handling Writes from PCI Masters

PCI bridge responds to Memory Write and Memory Write and Invalidate commands. All PCI master writes are posted. A 64-byte write buffer is used for this purpose. The write buffer accepts up to two separate PCI write transactions. Two single-beat writes, one burst write, or a combination of a single-beat and a burst writes can be posted. If the write buffer is full, new writes are retried until buffer space becomes available.

Note: The maximum of two posted transactions is as viewed from the PCI master. The number of writes performed on the PLB can be more than two, depending on the setting of byte enables of write burst data. See *Byte Enable Handling* on page 377.

The PCI bridge begins a PLB write request as soon as a PCI master write has completed on the PCI bus, or a bursting PCI master has written at least six words of data. The PCI bridge continues to receive data from a bursting PCI master as it transfers data to the PLB. If the write post buffer fills, the PCI master is target disconnected. If the write post buffer empties, the PLB cycle is master terminated.

Writes are executed in the same order they are received.

Byte Enable Handling

The PLB does not support non-contiguous byte enables, whereas the PCI bus does. The PLB supports the use of byte enables only for non-line, non-burst transactions, whereas the PCI bus supports any combination of byte enables for any data phase. Therefore, when a PCI master presents a data phase without all byte enables asserted, the bridge disconnects and treats that data phase as one or two single-beat writes on PLB, depending on whether or not byte enables are non-contiguous.

Masters presenting writes without all byte enables asserted experience degraded performance.

19.5.2.3 Miscellaneous

The PCI target forces single-beat transfers when reserved burst or cache line wrap order is used. The PCI target causes master abort of reserved command encodings, and does not respond to I/O, interrupt acknowledge, or special cycle commands. The PLB master does not abort requests.

19.5.3 Completion Ordering

PCI bridge implements the following completion ordering rules:

1. PCI master writes are accepted if there is room in the PCI write post buffer.
2. New PCI master reads are accepted if there is no delayed read (DRR or DRC) in progress:
   a. If PCI write post buffer is empty and read data is not buffered, then begin a delayed read (enter DRR state).
   b. If PCI write post buffer is not empty, then begin delayed read (enter DRR state).

   Delayed reads are handled as follows:
   a. While in DRR state, retry all PCI master reads. Wait for all PCI master writes if any that were posted before entering DRR state to complete on PLB.
   b. Execute PLB read, enter DRC state.
   c. While in DRC state, retry all PCI master reads if the address does not match. If it does match, pass the read data to the PCI master. If data is passed, exit the DRC state.
3. PLB master writes are accepted if there is room in the PLB write post buffer.
4. PLB master reads are accepted if the PLB write post buffer and the PCI write post buffer are both empty.

19.5.3.1 PCI Producer-Consumer Model

The PCI Producer-Consumer model is followed with one exception: PCI master reads do not flush PLB writes and PCI master writes do not cause PLB prefetched read data to be discarded. If the "flag" is stored in system memory (PLB side), but the "data" is stored in a PCI target, the control software must manually force coherency. This can be done by following two rules:

1. To ensure data written by a PLB master has reached the intended PCI target, the PLB master should execute a read from PCI, to any nondestructive address. This is only necessary if the write is postable.
2. To ensure data read by a PLB master is current (rather than old prefetched data), the PLB master should execute a read from PCI to any other nondestructive address. This is only necessary if the read is prefetchable.

19.5.4 Collision Resolution

The PCI bridge must resolve collisions when a PLB master and a PCI master attempt accesses through the PCI bridge at the same time. *Table 19-7* summarizes collision resolution.

In general, PLB postable writes are always accepted (if buffer space is available), and passed to the PCI when given the chance. PCI master writes are always accepted (if buffer space is available), but cause PLB reads and non-postable writes to be rearbitrated to clear the path to the PLB. PLB reads and non-postable writes proceed as long as there is no PCI master activity, which causes the PLB cycle to be rearbitrated. PCI master reads are always allowed to proceed, and cause PLB reads and non-postable writes to be rearbitrated.

Internal configuration accesses have their own rules. Configuration writes are not allowed to complete while any write data is posted in the PCI bridge, or while the PCI master is prefetching. Otherwise, there are no restrictions. Configuration reads have no restrictions; however, only one internal configuration access (PLB or PCI side) may be serviced at a time.

Table 19-7. Collision Resolution

Access Type	PLB Read from PCI	PLB Postable Write to PCI	PLB Non-postable Write to PCI
PCI Write to PLB	Rearbitrate PLB master (reads flush writes)	No conflict	Rearbitrate PLB master
PCI Read from PLB	Rearbitrate PLB master	No conflict	Rearbitrate PLB master

19.5.5 PCI Control Register (SDR0_PCI0)

The following figure describes the SDR0_PCI0 register bits.

Figure 19-7. PCI Control Register (SDR0_PCI0)

0	PAE	PCI Arbiter Enable 0 PCI arbiter disabled 1 PCI arbiter enabled	Reset value = SDR0_SDSTP1[PAE]
1	PHCE	PCI Host Configuration Enable 0 PCI host configuration disabled 1 PCI host configuration enabled	Reset value = SDR0_SDSTP1[PHCE]
2	PAME	PCI Asynchronous Mode Enable 0 Reserved 1 PCI asynchronous mode enabled	Reset value = SDR0_SDSTP1[PAME] PAME = 0 is not supported.
3:31		Reserved	Reset value = 0

19.6 PCI Bridge Configuration Registers

The PCI bridge has two sets of configuration registers for configuring the bridge, handling errors, and reporting status. Local configuration registers control PLB functions, and are accessed only from the PLB. The PCI configuration registers control PCI functions, and can be accessed from the PLB and the PCI. In addition, the mechanism used to access the bridge configuration registers may also be used to configure other devices on the PCI bus.

19.6.1 PCI Bridge Register Summary

Table 19-8 provides a summary of all of the PCI bridge registers. The registers are discussed in detail in the following sections. See *Reset and Initialization* on page 189 for register reset values.

Table 19-8. Directly Accessed MMIO Registers

Register	Address	Access	Description	Page
PCIL0_PMM0LA	0x0 EF40 0000	R/W	PMM 0 Local Address	381
PCIL0_PMM0MA	0x0 EF40 0004	R/W	PMM 0 Mask/Attribute	384
PCIL0_PMM0PCILA	0x0 EF40 0008	R/W	PMM 0 PCI Low Address	382
PCIL0_PMM0PCIHA	0x0 EF40 000C	R/W	PMM 0 PCI High Address	382
PCIL0_PMM1LA	0x0 EF40 0010	R/W	PMM 1 Local Address	382
PCIL0_PMM1MA	0x0 EF40 0014	R/W	PMM 1 Mask/Attribute	381
PCIL0_PMM1PCILA	0x0 EF40 0018	R/W	PMM 1 PCI Low Address	383
PCIL0_PMM1PCIHA	0x0 EF40 001C	R/W	PMM 1 PCI High Address	383
PCIL0_PMM2LA	0x0 EF40 0020	R/W	PMM 2 Local Address	383
PCIL0_PMM2MA	0x0 EF40 0024	R/W	PMM 2 Mask/Attribute	384
PCIL0_PMM2PCILA	0x0 EF40 0028	R/W	PMM 2 PCI Low Address	384
PCIL0_PMM2PCIHA	0x0 EF40 002C	R/W	PMM 2 PCI High Address	384
PCIL0_PTM1MS	0x0 EF40 0030	R/W	PTM 1 Memory Size/Attribute	385
PCIL0_PTM1LA	0x0 EF40 0034	R/W	PTM 1 Local Address	385
PCIL0_PTM2MS	0x0 EF40 0038	R/W	PTM 2 Memory Size/Attribute	385
PCIL0_PTM2LA	0x0 EF40 003C	R/W	PTM 2 Local Address	386

Table 19-9. PCI Configuration Address and Data Registers

Register	Address	Access	Description
PCIC0_CFGADDR	0x0 EEC0 0000	R/W	PCI Configuration Address Register
PCIC0_CFGDATA	0x0 EEC0 0004	R/W	PCI Configuration Data Register

Table 19-10. PCI Configuration Register Offsets

Register	Offset	Access		Description
		PLB	PCI	
PCIC0_VENDID	0x0 8000 0000	R/W	R	PCI Vendor ID
PCIC0_DEVID	0x0 8000 0002	R/W	R	PCI Device ID
PCIC0_CMD	0x0 8000 0004	R/W	R/W	PCI Command Register
PCIC0_STATUS	0x0 8000 0006	R/W	R/W	PCI Status Register
PCIC0_REVID	0x0 8000 0008	R/W	R/W	PCI Revision ID
PCIC0_CLS	0x0 8000 0009	R/W	R	PCI Class Register
PCIC0_CACHELS	0x0 8000 000C	R	R	PCI Cache Line Size
PCIC0_LATTIM	0x0 8000 000D	R/W	R/W	PCI Latency Timer
PCIC0_HDTYPE	0x0 8000 000E	R	R	PCI Header Type
PCIC0_BIST	0x0 8000 000F	R	R	PCI Built In Self Test Control
PCIC0_PTM1BAR	0x0 8000 0014	R/W	R/W	PCI PTM 1 BAR
PCIC0_PTM2BAR	0x0 8000 0018	R/W	R/W	PCI PTM 2 BAR
PCIC0_SBSYSVID	0x0 8000 002C	R/W	R	PCI Subsystem Vendor ID
PCIC0_SBSYSID	0x0 8000 002E	R/W	R	PCI Subsystem ID
PCIC0_CAP	0x0 8000 0034	R	R	PCI Capabilities Pointer
PCIC0_INTLN	0x0 8000 003C	R/W	R/W	PCI Interrupt Line
PCIC0_INTPN	0x0 8000 003D	R	R	PCI Interrupt Pin
PCIC0_MINGNT	0x0 8000 003E	R	R	PCI Minimum Grant
PCIC0_MAXLTNCY	0x0 8000 003F	R	R	PCI Maximum Latency
PCIC0_ICS	0x0 8000 0044	R/W	R/W	PCI Interrupt Control/Status
PCIC0_ERREN	0x0 8000 0048	R/W	R/W	Error Enable
PCIC0_ERRSTS	0x0 8000 0049	R/W	R/W	Error Status
PCIC0_BRDGOPT1	0x0 8000 004A	R/W	R/W	PCI Bridge Options 1
PCIC0_PLBBESR0	0x0 8000 004C	R/W	R/W	PLB Slave Error Syndrome 0
PCIC0_PLBBESR1	0x0 8000 0050	R/W	R/W	PLB Slave Error Syndrome 1
PCIC0_PLBBEAR	0x0 8000 0054	R/W	R/W	PLB Slave Error Address Register
PCIC0_CAPID	0x0 8000 0058	R	R	Capability Identifier
PCIC0_NEXTIPTR	0x0 8000 0059	R	R	Next Item Pointer
PCIC0_PMC	0x0 8000 005A	R	R	Power Management Capabilities
PCIC0_PMCSR	0x0 8000 005C	R/W	R/W	Power Management Control Status
PCIC0_PMCSRBSE	0x0 8000 005E	R	R	PMCSR PCI-to-PCI Bridge Support Extensions
PCIC0_DATA	0x0 8000 005F	R	R	Data

Table 19-10. PCI Configuration Register Offsets (continued)

Register	Offset	Access		Description
		PLB	PCI	
PCIC0_BRDGOPT2	0x0 8000 0060	R/W	R/W	PCI Bridge Options 2
PCIC0_PMSCRR	0x0 8000 0064	R/W	R/W	Power Management State Change Request Register

19.6.2 PCI Bridge Local Configuration Registers

The PCI bridge local configuration registers have fixed addresses in PLB space and must be accessed using single beat PLB read or write cycles of the same size as shown in the register descriptions.

Failure to access all bytes of a particular register could produce unexpected results. Reading of reserved bit locations produces unpredictable values. Software must use appropriate masks to extract the desired bits. Writes must preserve the values of reserved bit positions by first reading the register, merging the new value, and writing the result.

19.6.2.1 PMM 0 Local Address Register (PCIL0_PMM0LA)

PCIL0_PMM0LA defines the PLB starting address of range 0 in PLB space that is mapped to PCI memory. Only bits that are 1 in the PCIL0_PMM0MA are used to determine the starting address; all other bits are don't cares. Only bits 31:12 are writable. Bits 11:0 are always 0.

Figure 19-8. PMM 0 Local Address Register (PCIL0_PMM0LA)

31:12	WLA	Writable PLB Local Address	
11:0		PLB Local Address	Always 0

19.6.2.2 PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)

PCIL0_PMM0MA controls the size and attributes of the PLB space mapped to PCI memory for range 0.

Figure 19-9. PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)

31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM0LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as all ones.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable 1 Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.

0	ENA	PLB to PCI Memory Mapping Enable 1 Memory mapping is enabled.	Note that PCIL0_PMM0LA, PCIL0_PMM0PCIHA, and PCIL0_PMM0PCILA must be initialized before enabling.

19.6.2.3 PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)

PCIL0_PMM0PCILA defines the low-order 32 bits of the PCI address generated in response to a PLB access to range 0. Only bits that are 1 in PCIL0_PMM0MA are passed to the PCI address. The other (least significant) bits of the PCI address are passed through from the PLB address. Only bits 31:12 are writable. Bits 11:0 are always 0.

Figure 19-10. PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)

31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

19.6.2.4 PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)

PCIL0_PMM0PCIHA defines the high-order 32 bits of the PCI address generated in response to a PLB access to range 0. If PCIL0_PMM0PCIHA is greater than 0, the PCI bridge generates a PCI dual address cycle using the value in PCIL0_PMM0PCIHA as the high-order 32 bits of the PCI address.

Figure 19-11. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA)

31:0		PCI High Address	

19.6.2.5 PMM 1 Local Address Register (PCIL0_PMM1LA)

PCIL0_PMM1LA defines the PLB starting address of range 1 in PLB space that is mapped to PCI memory. See *PMM 0 Local Address Register (PCIL0_PMM0LA)* on page 381.

Figure 19-12. PMM 1 Local Address Register (PCIL0_PMM1LA)

31:0		PLB Local Address	

19.6.2.6 PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)

PCIL0_PMM1MA defines the size and attributes of range 1 in PLB space that is mapped to PCI memory. See *PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)* on page 381.

Figure 19-13. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)			
31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM1LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as 0x11111.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable 1 Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable 1 Memory mapping is enabled.	Note that PCIL0_PMM1LA, PCIL0_PMM1PCIHA, and PCIL0_PMM1PCILA must be initialized before enabling.

19.6.2.7 PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)

PCIL0_PMM1PCILA defines the low-order 32 bits of the PCI address generated in response to a PLB access to range 1. See *PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)* on page 382.

Figure 19-14. PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)			
31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

19.6.2.8 PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)

PCIL0_PMM1PCIHA defines the high-order 32 bits of the PCI address generated in response to a PLB access to range 1. See *PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)* on page 382.

Figure 19-15. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA)			
31:0		PCI High Address	

19.6.2.9 PMM 2 Local Address Register (PCIL0_PMM2LA)

PCIL0_PMM2LA defines the PLB starting address of range 2 in PLB space that is mapped to PCI memory. See *PMM 0 Local Address Register (PCIL0_PMM0LA)* on page 381.

Figure 19-16. PMM 2 Local Address Register (PCIL0_PMM2LA)			
31:0		PLB Local Address	

19.6.2.10 PMM 2 Mask/Attribute Register (PCIL0_PMM2MA)

PCIL0_PMM2MA defines the size and attributes of range 2 in PLB space that is mapped to PCI memory. See *PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)* on page 381.

Figure 19-17. PMM 2 Mask/Attribute Register (PCIL0_PMM2MA)			
31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM2LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as 0x11111.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable 1 Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable 1 Memory mapping is enabled.	Note that PCIL0_PMM2LA, PCIL0_PMM2PCIHA, and PCIL0_PMM2PCILA must be initialized before enabling.

19.6.2.11 PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)

PCIL0_PMM2PCILA defines the low-order 32 bits of the PCI address generated in response to a PLB access to range 2. See *PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)* on page 382.

Figure 19-18. PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)			
31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

19.6.2.12 PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)

PCIL0_PMM2PCIHA defines the high-order 32 bits of the PCI address generated in response to PLB access to range 2. See *PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)* on page 382.

Figure 19-19. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)			
31:0		PCI High Address	

19.6.2.13 PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)

PCIL0_PTM1MS defines the size and attributes of the of PCI memory region mapped to local (PLB) space through PTM 1. PCIL0_PTM1MS affects the operation of PCI PTM 1 BAR.

Figure 19-20. PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)			
31:12	MASK	Defines the size of the region of PCI memory space that is mapped to local (PLB) space using PTM 1.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
11:1		Reserved	Returns 0 when read.
0	ENA	Determines if range 1 is enabled to map PCI memory space to PLB space.	

19.6.2.14 PTM 1 Local Address Register (PCIL0_PTM1LA)

PCIL0_PTM1LA defines the local (PLB) address that is generated in response to a PCI access to local (PLB) space through PTM 1. Only bits that are 1 in PCIL0_PTM1MS are passed to the PLB address. The other (least significant) bits of the PLB address are passed through from the PCI address. Only bits 31:12 are writable. Bits 11:0 are always 0.

Figure 19-21. PTM 2 Local Address Register (PCIL0_PTM1LA)			
31:12	WLA	Writable PTM 1 Local Address	Writable
11:0		PTM 1 Local Address	Always 0

19.6.2.15 PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)

PCIL0_PTM2MS defines the size of the region of PCI memory space mapped to local (PLB) space through PTM 2.

Figure 19-22. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)			
31:12	MASK	Defines the size of the region of PCI memory space mapped to local (PLB) space using PTM 2.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
11:1		Reserved.	Returns 0 when read.

0	ENA	Determines if range 2 is enabled to map PCI memory space to PLB space.	When ENA is disabled, PCIC0_PTM2BAR cannot be written. Set PCIC0_PTM2BAR to 0 before disabling ENA.

19.6.2.16 PTM 2 Local Address Register (PCIL0_PTM2LA)

This register defines the local (PLB) address generated in response to a PCI access to local (PLB) space through PTM 2. See *PTM 1 Local Address Register (PCIL0_PTM1LA)* on page 385 for more information.

Figure 19-23. PTM 2 Local Address Register (PCIL0_PTM2LA)

31:0		PTM 2 Local Address	

19.6.3 PCI Configuration Registers

The PCI configuration registers can be accessed from both the PLB and PCI buses. PLB side configuration is supported using the mechanism defined in the *PCI Local Bus Specification* Version 2.2. This mechanism uses PCIC0_CFGADDR and PCIC0_CFGDATA to access the configuration registers indirectly. These registers reside at addresses 0xEEC00000 and 0xEEC00004, respectively. To access (from the PLB side) the configuration space of other devices on the PCI bus, write a value to PCIC0_CFGADDR that specifies the following:

- Bus number
- Device number on that bus
- Register number to be accessed

The value must also set PCIC0_CFGADDR[EN] = 1. An access to PCIC0_CFGDATA then results in a configuration cycle on the PCI bus.

To access the bridge configuration registers from the PLB side, use the same mechanism as described above, but set PCIC0_CFGADDR[BN, DN] = 0. The bridge is assumed to reside on PCI bus 0 and to have a device number of 0.

The bridge configuration registers can be accessed from the PCI side by Type 0 configuration reads or writes, with the PCIIDSel pin asserted to the bridge. There are some restrictions on PCI side accesses that are noted in the register descriptions that follow.

PCIC0_CFGADDR and CONFIG_DATA should be accessed with single-beat PLB commands. All registers are byte addressable. Reading of reserved bit locations produces unpredictable values. Software must use appropriate masks to extract the desired bits. Writes must preserve the values of reserved bit positions by first reading the register, merging the new value, and writing the result.

19.6.3.1 PCI Configuration Address Register (PCIC0_CFGADDR)

PCIC0_CFGADDR controls the type of cycle generated when PCIC0_CFGDATA is accessed. Its fields are shown in *Figure 19-24*.

Figure 19-24. PCI Configuration Address Register (PCIC0_CFGADDR)

31	EN	Enable 0 Disabled 1 Enabled	

30:24		Reserved	
23:16	BN	Bus Number	
15:11	DN	Device Number	
10:8	FN	Function Number	
7:2	RN	Register Number	
1	0		
0	0		

See the *PCI Local Bus Specification* Version 2.2 for details about how the fields are used.

19.6.3.2 PCI Configuration Data Register (PCIC0_CFGDATA)

Accessing PCIC0_CFGDATA causes one of three things to happen, depending on the value of PCIC0_CFGADDR.

1. Generation of a Type 0 configuration cycle on the PCI bus (PCIC0_CFGADDR[BN] = 0, PCIC0_CFGADDR[DN] > 0).
2. Generation of a Type 1 configuration cycle on the PCI bus (PCIC0_CFGADDR[BN] > 0).
3. Access of a PCI bridge PCI configuration register (PCIC0_CFGADDR[BN, DN] = 0).

Figure 19-25 illustrates the PCIC0_CFGDATA register.

Figure 19-25. PCI Configuration Data Register (PCIC0_CFGDATA)

31:0		Configuration Data	

19.6.3.3 PCI Vendor ID Register (PCIC0_VENDID)

PCIC0_VENDID identifies the manufacturer of a PCI device. This register contains 0x1014 (index 0x00 = 0x14, index 0x01 = 0x10) at reset. This vendor ID is assigned for all PCI devices. The local CPU (PLB master) can write a different value to this register.

Figure 19-26. PCI Vendor ID Register (PCIC0_VENDID)

15:0		Vendor ID	

19.6.3.4 PCI Device ID Register (PCIC0_DEVID)

PCIC0_DEVID identifies the PCI device. This value is 0x0156 (index 0x03 = 0x01, index 0x02 = 0x56) at reset. The local CPU (PLB master) can write a different value to this register.

Figure 19-27. PCI Device ID Register (PCIC0_DEVID)

15:0		PCI Device ID	

19.6.3.5 PCI Command Register (PCIC0_CMD)

PCIC0_CMD controls the operation of the PCI bridge on the PCI bus. *Figure 19-28* describes the bits.

Figure 19-28. PCI Command Register (PCIC0_CMD)

15:10		Reserved	.
9	FBB	Fast Back-to-Back Write Enable	Enables PCI masters to perform fast back-to-back transactions. Because he PCI bridge does not perform fast back-to-back transactions; FBB is read-only and returns 0 when read.
8	SE	$\overline{\text{PCISErr}}$ Enable 0 Disabled 1 Enabled	Enables driving $\overline{\text{PCISErr}}$ when a PCI bus parity error is detected when the PCI bridge is the PCI target. PCIC0_CMD[PER] must be enabled for address parity errors. PCIC0_CMD[PER] and PCIC0_ERREN[WDPE] must be enabled for write data parity errors.
7	AS	Address stepping wait states.	The PCI bridge does not address step (except for address stepping when generating a Config Type 0 cycle); AS is read-only and returns 0 when read.
6	PER	Parity error response 0 Disabled 1 Enabled	This bit is enabled for all types of PCI bus parity errors, including the following: • PCI data bus parity errors while PCI is master. • PCI data bus parity errors while PCI is target. • PCI address bus parity errors. When parity error response is disabled, detection of these errors is masked and $\overline{\text{PCIPErr}}$ (PERR#) is not asserted, although parity is still generated.
5	PS	Palette Snooping	Enable special palette snooping. The PCI bridge is not a VGA device; PS is read-only and returns 0 when read
4	MWI	Memory Write and Invalidate Enable	The PCI bridge does not generate this command; MWI is read-only and returns 0 when read.
3	SC	Special Cycle Operations Enable	The PCI bridge never monitors special cycles; SC is read-only and returns 0 when read.
2	ME	Master Enable 0 Disabled 1 Enabled	Enables PCI bridge-to-master cycles on the PCI bus. When ME is 0, the PCI bridge only responds as a PLB slave to PCIC0_CFGADDR, PCIC0_CFGDATA, and PCI bridge local configuration register access. Except for configuration cycles, the PCI bridge cannot master cycles to the PCI bus. If the pin strapping setting reflected in CPC0_PSR[RL] = 1, ME resets to 1.
1	MA	Memory Access 0 Disabled 1 Enabled	Controls PCI bridge response as a PCI memory target. MA is disabled at reset.
0	IOA	I/O Access	Controls the PCI bridge response as a PCI I/O target. The PCI bridge does not respond to I/O space accesses; IOA is read-only and returns 0 when read.

19.6.3.6 PCI Status Register (PCIC0_STATUS)

PCIC0_STATUS is a read/bit-reset register used to record status information for PCI bus events. Bits in PCIC0_STATUS are set only as a result of specific events occurring on the PCI bus. They are reset by writing a 1 to the desired bit. Writing a 0 to a bit location leaves that bit unchanged.

Figure 19-29. PCI Status Register (PCIC0_STATUS)

15	DEPE	Detected Parity Error Write 1 to clear.	The PCI bridge sets DEPE when the PCI bridge detects a PCI bus parity error, regardless of the setting of any enable bits (DEPE is non-maskable). The following events set DEPE: PCI address bus parity error detected when PCI bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PCI bridge is the target). PCI data bus parity error detected when PCI bridge masters a PCI read cycle.
14	SSE	Signaled System Error Write 1 to clear.	The PCI bridge sets SSE if the PCI bridge asserts PCISErr (see *Error Handling* on page 403 for causes of PCISErr assertion).
13	RMA	Received Master Abort Write 1 to clear.	The PCI bridge sets RMA when a PCI cycle for which the PCI bridge is the master is terminated with master abort.
12	RTA	Received Target Abort Write 1 to clear.	The PCI bridge sets RTA when a PCI cycle for which it is the master is terminated with target abort.
11	STA	Signaled Target Abort Write 1 to clear.	The PCI bridge sets STA when a PCI cycle for which it is the target is terminated with target abort.
10:9	DST	$\overline{\text{PCIDevSel}}$ Response Timing Read-only.	The PCI bridge asserts $\overline{\text{PCIDevSel}}$ on the second clock after PCIFframe is asserted (called medium response time). Read-only; always returns 0b01 when read.
8	DPE	Data Parity Error Detected Write 1 to clear.	DPE is set when the following conditions are met: The PCI bridge detects a data parity error ($\overline{\text{PCIPErr}}$ is asserted) when the PCI bridge is the master on a PCI read cycle, or is the master when it samples $\overline{\text{PCIPErr}}$ asserted on a PCI write cycle. PCIC0_CMD[PER] = 1.
7	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Indicates that the PCI target can accept fast back-to-back transactions when the transactions are not to the same agent. The PCI bridge target does not accept this type of fast back-to-back transaction.
6	UDFS	UDF Supported Read-only; returns 0 when read.	Indicates device support of user-definable features. The PCI bridge does not support user-definable features.
5	66C	66 MHz Capable At reset PCI bridge is configured for 66MHz operation.	Indicates that the device can run at 66 MHz. The PCI bridge can be configured to run at 33 MHz max or 66 MHz. The local CPU (PLB master) sets 66C to 1 if PCI bridge is configured for 66 MHz operation.
4	CL	Capabilities List This bit is read only and returns 1 when read.	Indicates that the value at offset 0x34 is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits return 0s when read.

19.6.3.7 PCI Revision ID Register (PCIC0_REVID)

PCIC0_REVID holds the current incremental revision number. The reset value is the version number of the PCI bridge macro (current version is 20). However, the local CPU (PLB master) can write a value to this register.

Figure 19-30. PCI Revision ID Register (PCIC0_REVID)			
7:0		Revision ID	Revision level of device.

19.6.3.8 PCI Class Register (PCIC0_CLS)

This register holds the class code. This register is 0x060000 at reset, which indicates that the PCI bridge is a bridge device located between the PLB and the PCI bus; however, the local CPU (PLB master) can write a value to this register for the case where PCI bridge is not the host bridge.

Class information is defined in the *PCI Local Bus Specification*, Version 2.2.

Figure 19-31. PCI Class Register (PCIC0_CLS)			
23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device. Users of the RISCWatch debugger must use the PCIC0_BASECC register to access this field.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge. Users of the RISCWatch debugger must use the PCIC0_SUBCLS register to access this field.
7:0	INT	Interface Class	Reset to 00. Users of the RISCWatch debugger must use the PCIC0_INTCLS register to access this field.

19.6.3.9 PCI Cache Line Size Register (PCIC0_CACHELS)

PCIC0_CACHELS determines the size of a PCI cache line. PCI bridge does not support a PCI cache. Therefore, this register is read-only and returns 0x00 when read.

Figure 19-32. PCI Cache Line Size Register (PCIC0_CACHELS)			
7:0		PCI Cache Line Size	

19.6.3.10 PCI Latency Timer Register (PCIC0_LATTIM)

PCIC0_LATTIM holds the value of the PCI latency timer. The granularity of the latency timer is 8 PCI cycles. POR value is 0x00.

Figure 19-33. PCI Latency Timer Register (PCIC0_LATTIM)			
7:0		PCI Latency Timer	

PCI Local Bus Specification Version 2.2 specifies that PCI masters capable of multi-beat bursts must, after losing their PCI grant, get off the bus when PCIC0_LATTIM has decremented to 0.

The actual number of clock cycles to disconnect varies somewhat when in asynchronous mode. If PCIC0_LATTIM is programmed in asynchronous mode to a value that is less than 64, the PCI bridge PCI master interface could timeout, regardless of the state of its grant line.

In asynchronous mode, the PCI master starts its timer and can timeout regardless of the state of the PCI grant. This is strictly a performance issue and does not limit functionality or affect data integrity.

Several factors affect the frequency of time outs.

- The amount of PCI bus traffic. In moderate to heavily loaded systems, this is less of an issue because a PCI master tends to lose its grant more often after gaining the bus.
- Fast targets that introduce few or no wait states reduce the chances of time outs occurring.

19.6.3.11 PCI Header Type Register (PCIC0_HDTYPE)

PCIC0_HDTYPE (bits 0:6) identifies the second part of the PCI header, which begins at offset 0x10. It also determines whether a device contains multiple functions (bit 7). The PCI bridge implements the standard header and is not a multi-function device; therefore, PCIC0_HDTYPE is read-only and returns 0x00 when read.

Figure 19-34. PCI Header Type Register (PCIC0_HDTYPE)

7:0		PCI Header Type	

19.6.3.12 PCI Built-In Self Test (BIST) Control Register (PCIC0_BIST)

PCIC0_BIST is used for control and status of BIST. PCI bridge does not implement BIST. PCIBIST is read-only and returns 0x00 when read.

Figure 19-35. PCI Built-in Self Test Control Register (PCIC0_BIST)

7:0		PCI BIST Control	

19.6.3.13 Unused PCI Base Address Register Space

PCI base address register space is defined to begin at offset 0x10; however, the first 32 bits of this space are unused by PCI bridge, and the defined base address registers begin at offset 0x14.

19.6.3.14 PCI PTM 1 BAR (PCIC0_PTM1BAR)

PCIC0_PTM1BAR defines a space in PCI memory space mapped to PLB space (system memory or ROM).

Figure 19-36. PCI PTM 1 BAR Register (PCIC0_PTM1BAR)			
31:12	BA	Base Address These bits determine where in PCI memory address space this region is located.	Only corresponding bits in PCIL0_PTM1MS that are set to 1 are writable. Bits in PCIL0_PTM1MS that are set to 0 cause the corresponding Base Address register bits to be always 0. PCIL0_PTM1MS must be initialized by a PLB master before any PCI device is allowed to configure this register.
11:4	BAZ	Base Address Always Zero	BAZ = 0x00 because the minimum size of this range is 4KB.
3	PF	Prefetchable	PF = 1 to indicate that prefetching is allowed.
2:1	LT	Location Type	LT = 0b00 to indicate that the memory space can be located anywhere in the 32-bit address space.
0	MSI	Memory Space Indicator	MSI = 0 to indicate memory space, rather than I/O space.

19.6.3.15 PCI PTM 2 BAR (PCIC0_PTM2BAR)

PCIC0_PTM2BAR defines a second space in PCI memory space that is mapped to PLB space (system memory or ROM). Note that if PCIC0_PTM2BAR is disabled using PCIL0_PTM2MS, PCIC0_PTM2BAR cannot be written. Set PCIC0_PTM2BAR to 0 before disabling this bit. If disabled in this way, reads to PCIC0_PTM2BAR always return 0.

Figure 19-37. PCI PTM 2 BAR Register (PCIC0_PTM2BAR)			
31:12	BA	Base Address These bits determine where in PCI Memory address space this region is located.	Only corresponding bits in PCIL0_PTM2MS that are set to 1 are writable. Bits in PCIL0_PTM2MS that are set to 0 cause the corresponding Base Address register bits to be always 0. PCIL0_PTM2MS must be initialized by a PLB master before any PCI device can configure this register.
11:4	BAZ	Base Address Always Zero	BAZ = 0x00 because the minimum size of this range is 4KB.
3	PF	Prefetchable	PF = 1 to indicate that prefetching is allowed.
2:1	LT	Location Type	LT = 0b00 to indicate that the memory space can be located anywhere in the 32-bit address space.
0	MSI	Memory Space Indicator	MSI = 0 to indicate memory space, rather than I/O space.

19.6.3.16 PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)

PCIC0_SBSYSVID holds the vendor ID for a subsystem or add-in board.

Figure 19-38. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)			
15:0		PCI Subsystem Vendor ID	

19.6.3.17 PCI Subsystem ID Register (PCIC0_SBSYSID)

PCIC0_SBSYSID holds the device ID of a subsystem or add-in board.

Figure 19-39. PCI Subsystem ID Register (PCIC0_SBSYSID)			
15:0		PCI Subsystem ID	

19.6.3.18 PCI Capabilities Pointer (PCIC0_CAP)

PCIC0_CAP contains an 8-bit pointer in configuration space to the next capability. This data structure is indicated by PCIC0_STATUS[CL]. PCIC0_CAP points to the first item in the list of capabilities at address offset 0x58, which is the PCI power management capability structure.

Figure 19-40. PCI Capabilities Pointer (PCIC0_CAP)			
7:0		PCI Capabilities Pointer	

19.6.3.19 PCI Interrupt Line Register (PCIC0_INTLN)

PCIC0_INTLN contains interrupt line routing information.

Figure 19-41. PCI Interrupt Line Register (PCIC0_INTLN)			
7:0		PCI Interrupt Line	

19.6.3.20 PCI Interrupt Pin Register (PCIC0_INTPN)

PCIC0_INTPN specifies the PCI interrupt line that the device uses. The value 0x01 indicates INTA#.

Figure 19-42. PCI Interrupt Pin Register (PCIC0_INTPN)			
7:0		PCI Interrupt Pin	

19.6.3.21 PCI Minimum Grant Register (PCIC0_MINGNT)

PCIC0_MINGNT specifies the burst period length of a PCI device. PCIC0_MINGNT is read-only and returns 0x00 when read.

Figure 19-43. PCI Minimum Grant Register (PCIC0_MINGNT)

7:0		PCI Minimum Grant	

19.6.3.22 PCI Maximum Latency Register (PCIC0_MAXLTNCY)

PCIC0_MAXLTNCY specifies how often a PCI device needs to access to the PCI bus. PCIC0_MAXLTNCY is read-only and returns 0x00 when read.

Figure 19-44. PCI Maximum Latency Register (PCIC0_MAXLTNCY)

7:0		PCI Maximum Latency	

19.6.3.23 PCI Interrupt Control/Status Register (PCIC0_ICS)

A PLB master or a PCI master device may generate an interrupt to the PCI bus by writing a 1 to bit 0. Clearing this bit clears the interrupt. Bit 0 also reports the status of the interrupt. A value of 1 means that the interrupt is asserted, a value of 0 means that the interrupt is deasserted.

Figure 19-45. PCI Interrupt Control/Status Register

7:1		Reserved	These bits return 0 when read.
0	API	Assert PCI interrupt	When software sets this bit, the PCI bridge asserts its Interrupt pin.

19.6.3.24 Error Enable Register (PCIC0_ERREN)

ERREN enables detection and reporting of various errors for the PCI bridge (see *Error Handling* on page 403).

Figure 19-46. PCI Error Enable Register (PCIC0_ERREN)

7		Reserved	
6	TAEE	Target Abort Error Enable 0 Disabled 1 Enabled	While the PCI bridge is the PCI master, this bit enables the detection of a target abort as an error condition. If TAEE is enabled, the PCI bridge reports PLB bus errors.
5:4	MERE	PLB Bus Error Response Enable 00 No action is taken. 01 The PCI target should drive PCISErr on the PCI bus. 10 Target should target abort the offending read. 11 Indicates the PCI target should drive PCISErr and target abort.	MERE controls the response taken by the PCI bridge on the PCI bus (as the PCI target) when PLB bus errors are asserted to the PCI bridge PLB master. **Note:** Only reads can be target aborted. Modes 10 and 11 cannot be used in asynchronous mode.

3	MEDE	PLB Master Error Detection Enable 0 Disables detection of PLB master errors. 1 Enables detection of PLB master errors.	MEDE enables the detection of PLB bus errors when the PCI bridge is a PLB master.
2	MEAE	PLB Bus Error Assertion Enable 0 Disabled 1 Enabled	MEAE enables the reporting of a PLB bus error when the PCI bridge is a PLB slave.
1	WDPE	Write Data Parity $\overline{\text{PCISErr}}$ Enable 0 Disabled 1 Enabled.	The PCI bridge drives $\overline{\text{PCISErr}}$ when a data parity error is detected on a write cycle when the PCI bridge is the PCI target. PCIC0_CMD[SE] must also be 1.
0	MAEE	Master Abort Error Enable 0 Disabled 1 Enabled	MAEE enables the detection of a master abort as an error condition when the PCI bridge is the master. The PCI bridge drives SI_MErr on the PLB bus in response to a master abort. If this bit is disabled, driving of SI_Merr in response to master abort is masked.

19.6.3.25 Error Status Register (PCIC0_ERRSTS)

PCIC0_ERRSTS contains status for detected error conditions (see *Error Handling* on page 403). Bits in PCIC0_ERRSTS can be set to 1 only as the result of a system error. Writing a 1 to a PCIC0_ERRSTS bit clears the bit. Writing a 0 to a bit leaves that bit unchanged.

Figure 19-47. PCI Error Status Register (PCIC0_ERRSTS)

7:5		Reserved	
4	SARME	$\overline{\text{PCISErr}}$ Asserted on Received PLB Bus Error	Set when PCI bridge asserts $\overline{\text{PCISErr}}$ on the PCI bus in response to PCI bridge receiving a PLB bus error while PLB master.
3	MED	PLB Bus Error Detected 1 Error detected	Set when a PLB bus error signal is asserted when PCI bridge is the PLB master. MED is set regardless of whether the PCI bridge is enabled to treat this as an error condition (the setting of MED is not maskable).
2	MEAE	PLB Bus Error Assertion Event 1 An PCI bridge error, which can cause a PLB bus error, occurred.	Set when an error occurs that would cause PCI bridge (as PLB slave) to assert a PLB bus error signal. MEAE is set regardless of whether the PLB bus error assertion is enabled (the setting of MEAE is not maskable).
1	WDPE	$\overline{\text{PCISerr}}$ on Write Data Parity Error	Set when the PCI bridge drives $\overline{\text{PCISErr}}$ in response to a data parity error detected on a PCI write to PLB memory. $\overline{\text{PCIPErr}}$ is also driven.
0	PUR	PLB Unsupported Request	Set when the PCI bridge is a PLB slave and detects an unsupported request from a PLB master to an address range that PCI bridge decodes. The PCI bridge allows such requests to time out.

19.6.3.26 Bridge Options 1 Register (PCIC0_BRDGOPT1)

PCIC0_BRDGOPT1 controls various operating parameters of the PCI bridge. The parameters must be initialized before PCI masters access the PCI bridge.

Figure 19-48. PCI Bridge Options 1 Register (PCIC0_BRDGOPT1)

15:8	PMLTCR	PLB Master Latency Timer Count Register	PMLTCR contains the value used by the PLB master to load its latency timer. The granularity of this timer is 16 PLB cycles; therefore, the low-order bits of this register are read-only and are hard wired to 1.
7	PLESE	PLB Lock Error Status Enable 0 Slave error locking is disabled. 1 Slave error locking is enabled.	PLESE controls the handling of slave error locking.
6:5	PRP	PLB Request Priority 11 Highest 10 Next highest 01 Next highest 00 Lowest	PRP controls the request priority for PLB accesses.
4	PGMAE	PLB Guarded Memory Access Enable 0 Bridge PLB master memory accesses are unguarded. 1 Bridge PLB master memory accesses are guarded.	PGMAE controls whether PLB accesses are guarded or unguarded.
3	PAPM	PCI Arbiter Park Mode 0 The arbiter parks on requester 0 (the bridge PCI master). 1 The arbiter parks on the last master granted the bus.	PAPM defines how the internal PCI arbiter handles bus parking.
2:1	PTMRCI	PCI Target Memory Read Command Interpretation 00 Memory Read 01 Memory Read Line 10 Memory Read Multiples 11 Reserved	PTMRCI enables the PCI bridge to be forced to treat a PCI memory read as a memory read multiple, or as a memory read line, with respect to the burst size implied by the read commands. This is for masters that use memory read for multiple beat bursts.
0	APLRM	Atomic PLB Line Read Mode 0 1 PLB slave asserts AddrAck and begins its data tenure immediately after the PCI master receives the first read data word.	APLRM controls the behavior of the bridge PLB slave with respect to PLB line reads. APLRM must not be se t to 1 unless all PCI target devices can guarantee no disconnects for PLB line reads.

19.6.3.27 PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)

PCIC0_PLBBESR0 stores information about errors reported by the PCI bridge PLB slave. There are four groups of errors, one each for PLB masters 0–3. PCIC0_PLBBESR0[MxET] fields (x represents a particular PLB master ID) contain information about the type of error. PCI parity errors set PCIC0_PLBBESR0[MxET] to 0b001. Master and target aborts set PCIC0_PLBBESR0[MxET] to 0b101 (non-configured bank error). The PCIC0_PLBBESR0[MxRWS] fields show whether the transaction causing the error was a read or write.

Each error field can be locked by PCIC0_PLBBESR0[MxFL], which is set by a PLB lock error signal to the bridge PLB slave. If the PCIC0_PLBBESR0[MxFL] field associated with a master is 0, the PLB lock error signal is driven high to the bridge PLB slave, and an error associated with that master occurs. The error is then reported, and PCIC0_PLBBESR0[MxFL] is set. Subsequent errors do not set PCIC0_PLBBESR0 fields for that master until

PCIC0_PLBBESR0[MxFL] is cleared. If PCIC0_PLBBESR0[MxFL] = 0, and the PLB lock error signal is low, the error is reported, and PCIC0_PLBBESR0[MxFL] is not set. Additional errors are also reported. Only software can clear PCIC0_PLBBESR0[MxFL].

Writing 1 to a PCIC0_PLBBESR0 field clears the field.

Figure 19-49. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)

31:29	M0ET	Master 0 Error Type 000 No Error 001 Parity Error 010 Reserved 011 Reserved 100 Reserved 101 Non-configured Bank Error 110 Reserved 111 Reserved	
28	M0RWS	Master 0 Read/Write Status 0 Error operation was a write 1 Error operation was a read	
27	M0FL	Master 0 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBB ESR0 unlocked 1 PCIC0_PLBB ESR0 locked	
26	M0AL	Master 0 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 0 1 PCIC0_PLBBEAR locked by Master 0	
25:23	M1ET	Master 1 Error Type	See PCIC0_PLBBESR0[M0ET]
22	M1RWS	Master 1 Read/Write Status 0 Error operation was a write 1 Error operation was a read	
21	M1FL	Master 1 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBB ESR0 unlocked 1 PCIC0_PLBB ESR0 locked	
20	M1AL	Master 1 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 1 1 PCIC0_PLBBEAR locked by Master 1	
19:17	M2ET	Master 2 Error Type	See PCIC0_PLBBESR0[M0ET]
16	M2RWS	Master 2 Read/Write Status 0 Error operation was a write 1 Error operation was a read	
15	M2FL	Master 2 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBB ESR0 unlocked 1 PCIC0_PLBB ESR0 locked	
14	M2AL	Master 2 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 2 1 PCIC0_PLBBEAR locked by Master 2	
13:11	M3ET	Master 3 Error Type	See PCIC0_PLBBESR0[M0ET]

10	M3RWS	Master 3 Read/Write Status 0 Error operation was a write 1 Error operation was a read	
9	M3FL	Master 3 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBBESR0 unlocked 1 PCIC0_PLBBESR0 locked	
8	M3AL	Master 3 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR Unlocked by Master 2 1 PCIC0_PLBBEAR Locked by Master 2	
7:0		Reserved	

19.6.3.28 PLB Slave Error Syndrome Register 1 (PCIC0_PLBBESR1)

PCIC0_PLBBESR1 stores information about errors reported by the bridge PLB slave. There are four groups of errors, one each for PLB masters 4–7. See *PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)* on page 396 for additional information about the fields of this register.

Only software can clear PCIC0_PLBBESR1[MxFL]. The PCIC0_PLBBESR1[MxAL] fields control the and PCIC0_PLBBESR0 and PCIC0_PLBBESR1 in the same way. Writing a 1 to a field of the PCIC0_PLBBESRx clears the bit.

Figure 19-50. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1)

31:29	M4ET	Master 4 Error Type 000 No Error 001 Parity Error 010 Reserved 011 Reserved 100 Reserved 101 Non-configured Bank Error 110 Reserved 111 Reserved	
28	M4RWS	Master 4 Read/Write Status 0 Write error operation 1 Read error operation	
27	M4FL	Master 4 PCIC0_PLBBESR1 Field Lock 0 PCIC0_PLBBESR1 unlocked 1 PCIC0_PLBBESR1 locked	
26	M4AL	Master 4 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 4 1 PCIC0_PLBBEAR locked by Master 4	
25:23	M5ET	Master 5 Error Type	See PCIC0_PLBBESR1[M4ET]
22	M5RWS	Master 5 Read/Write Status 0 Write error operation 1 Read error operation	
21	M5FL	Master 5 PCIC0_PLBBESR1 Field Lock 0 PCIC0_PLBBESR1 Unlocked 1 PCIC0_PLBBESR1 Locked	

20	M5AL	Master 5 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 5 1 PCIC0_PLBBEAR locked by Master 5	
19:17	M6ET	Master 6 Error Type	See PCIC0_PLBBESR1[M4ET]
16	M6RWS	Master 6 Read/Write Status 0 Write error operation 1 Read error operation	
15	M6FL	Master 6 PCIC0_PLBBESR1 Field Lock 0 PCIC0_PLBBESR1 unlocked 1 PCIC0_PLBBESR1 locked	
14	M6AL	Master 6 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 6 1 PCIC0_PLBBEAR locked by Master 6	
13:11	M7ET	Master 7 Error Type	See PCIC0_PLBBESR1[M4ET]
10	M7RWS	Master 7 Read/Write Status 0 Write error operation 1 Read error operation	
9	M7FL	Master 7 PCIC0_PLBB ESR1 Field Lock 0 PCIC0_PLBBESR1 unlocked 1 PCIC0_PLBBESR1 locked	
8	M7AL	Master 7 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 7 1 PCIC0_PLBBEAR locked by Master 7	
19:0		Reserved	

19.6.3.29 PLB Slave Error Address Register (PCIC0_PLBBEAR)

PCIC0_PLBBEAR contains addresses associated with errors, as indicated by the PLB slave asserting Sl_MErr for transactions initiated by the PCI bridge on the PCI bus. PCIC0_PLBBEAR is read-only.

Figure 19-51. PLB Slave Error Address Register (PCIC0_PLBBEAR)

31:0		PLB Slave Error Address

19.6.3.30 Capability Identifier (PCIC0_CAPID)

When PCIC0_CAPID contains 0x01, the PCI bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

Figure 19-52. PCI Capability Identifier (PCIC0_CAPID)

7:0		PCI Capability Identifier	

19.6.3.31 Next Item Pointer (PCIC0_NEXTIPTR)

PCIC0_NEXTIPTR describes the location of the next item in the capability list of the function. PCIC0_NEXTIPTR is set to 0x00 to indicate that this is the last item on the capability list.

Figure 19-53. PCI Next Item Pointer (PCIC0_NEXTIPTR)			
7:0		PCI Next Item Pointer	

19.6.3.32 Power Management Capabilities (PCIC0_PMC)

PCIC0_PMC provides information about the capabilities of the function related to power management. A value of 0x0202 indicates no specific capabilities.

Figure 19-54. Power Management Capabilities Register (PCIC0_PMC)			
15:11	PMES	PME Support	The PCI bridge does not support PME#; therefore, PMES is hard wired to 0b00000.
10	D2S	D2 Support Determines if the D2 power management state is supported.	The PCI bridge does not support the D2 power management state; therefore, D2S is hard wired to 0.
9	D1S	D1 Support Determines if the D1 power management state is supported.	The PCI bridge supports the D1 power management state; therefore, D1S is hard wired to 1.
8:6	AUXCUR	Auxiliary Current Support	The PCI bridge does not support Aux_Current; therefore, AUXCUR is hard wired to 0b000.
5	DSI	Device Specific Initialization 0 after reset	This bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it.
4		Reserved	Always read as 0.
3	PMECLK		This bit is hard wired to 0 indicating that the function does not support PME# generation in any state.
2:0	VERS		Returns 0b010 on reads, indicating that PMC complies with Revision 1.1 of *PCI Power Management Interface Specification*.

19.6.3.33 Power Management Control/Status Register (PCIC0_PMCSR)

PCIC0_PMCSR is used to manage the PCI power management state and to enable and monitor PMEs.

Figure 19-55. Power Management Control/Status Register (PCIC0_PMCSR)			
15	PMEST		The PCI bridge does not support PME#; therefore, PMEST is hard wired to 0.
14:13	DSCAL		The PCI bridge does not support data register; therefore, DSCAL is hard wired to 0b00.
12:9	DSEL		The PCI bridge does not support a data register; therefore, DSEL is hard wired to 0b0000.
8	PMEEN		The PCI bridge does not support PME generation; therefore, PMEEN is hard wired to 0.
7:2		Reserved	Returns 0 when read.
1:0	PSTAT	Determine the current power state of a function and sets the function into a new power state. 00 D0 01 D1 10 D2 11 D3 Hot	If software attempts to write a value for an unsupported power state to PSTAT, its value does not change. Writing this field may change PCIC0_PMSCRR.

19.6.3.34 PMCSR PCI-to-PCI Bridge Support Extensions (PCIC0_PMCSRBSE)

PCIC0_PMCSRBSE is required for all PCI-to-PCI bridges. The PCI bridge in not a PCI-to-PCI bridge; therefore, it returns 0 when this register is read.

Figure 19-56. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBSE)			
7:0		PCI to PCI Bridge Support Extensions	

19.6.3.35 PCI Data Register (PCIC0_DATA)

PCIC0_DATA is an optional register that provides a mechanism for the function to report state dependent operating data such as power consumed or heat dissipation. The PCI bridge does not implement this register; therefore, it returns 0 when this register is read.

Figure 19-57. PCI Data Register (PCIC0_DATA)			
7:0		PCI Data	

19.6.3.36 Bridge Options 2 Register (PCIC0_BRDGOPT2)

PCIC0_BRDGOPT2 controls various operating parameters of the PCI bridge.

Figure 19-58. PCI Bridge Options 2 Register (PCIC0_BRDGOPT2)			
15:14		Reserved	
13	EWPCI	External Write to PCI Command Interrupt 0 No write to PCIC0_CMD has occurred. 1 External PCI master has written to PCIC0_CMD.	Software can set or clear this bit. Setting this bit also causes UIC0_SR[PCIIS] to be set.
12	DPR	Drive PCI Reset 0 Normal operation 1 Causes PCIReset pin to be asserted.	Software that asserts this bit must leave it asserted long enough to guarantee the PCI pulse width requirements. DPR does not reset PLB bus interface registers or PCI bridge registers. PCIReset is also asserted when the PCI bridge is reset.
11:8	PSTLTD	Subsequent Target Latency Timer Duration Specifies the number of PCI clocks that a PCI master burst can be held in a wait state before a target disconnect is initiated.	Only set on reads. In synchronous mode, PSTLTD equals the maximum number of PCI clocks to disconnect. In asynchronous mode, PSTLTD plus 3 equals the maximum number of PCI clocks to disconnect. The asynchronous value must be 2 or less.
7:3		Reserved	
2	PDTD	PCI Discard Timer Disable 0 Disabled 1 Enabled	When enabled, the PCI bridge never discards delayed read data.
1		Reserved	
0	HCE	Host Configuration Enable 0 Disabled 1 Enabled	HCE controls host PCI access to the PCI bridge configuration registers. All host attempts to access the PCI bridge PCI configuration registers are retried. This give the local CPU (PLB master) time to initialize them before the host sees them.

In synchronous mode, the PCI subsequent target latency timer duration equals the maximum number of PCI clocks to disconnect. In asynchronous mode, PCI subsequent target latency timer duration plus 3 equals the maximum number of PCI clocks to disconnect. The asynchronous value must be 2 or less.

19.6.3.37 Power Management State Change Request Register (PCIC0_PMSCRR)

PCIC0_PMSCRR provides a method of informing the local processor of a power management state change request and prevents the completion of the write to PCIC0_PMCSR until the local processor indicates it is ready for the state change. All writes to PCIC0_PMCSR are retried until the local processor sets PCIC0_PMSCRR[APW] = 1. PCIC0_PMSCRR is used with the registers in the capability structure for power management. Descriptions of each bit are shown in *Figure 19-59*.

Figure 19-59. Power Management State Change Request Register (PCIC0_PMSCRR)			
7:5		Reserved	Always read as 0.
4	APW	Accept PCIC0_PMCSR Writes Always 1 if DWE is 0.	The local processor sets APW when the local processor is ready to change the power management state. APW is cleared when the host configuration writes to the PCIC0_PMCSR register is accepted. The local processor can write 0 to APW.

3	SCR	State Change Request	The PCI bridge sets SCR when a host writes PCIC0_PMCSR to request a power management state change. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear SCR and set APW = 1 when the local processor is ready to change the state. After SCR is cleared, new requests are not detected until the outstanding delayed write is accepted. The local processor can set SCR = 1. Note that any host side write to any byte (0x5C–0x5F) is considered a power state change request.
2:1	REQST	Request State	Indicates the new power management state requested by a delayed host write to PCIC0_PMCSR. This field is read-only from the PLB side.
0	DWE	Delayed Write Enable 0 Immediate write 1 Delayed write	When DWE is set to 1, any configuration write to the PCIC0_PMCSR is completed as a delayed write. All writes to PCIC0_PMCSR are retried until the local processor sets the "Accept PCIC0_PMCSR Write bit" (bit 4). When 0, any configuration write to the PCIC0_PMCSR is completed immediately. DWE is a don't care if a host write to PCIC0_PMCSR requests a state change from D3hot to D0.

19.7 Error Handling

The PCI bridge detects and reports several types of errors, which are reported to the PLB or the PCI. Status information is saved in the configuration registers to enable error type determination.

All errors are associated with either a cycle on the PLB or a cycle on the PCI bus.

Each error that can be detected is associated with a mask. If the mask is set, detection of that error condition is disabled. There are also masks for the $\overline{\text{PCISErr}}$, $\overline{\text{PCIPErr}}$, and PLB bus error signals that prevent reporting of any error by these signals. The masks do not prevent error detection.

The error types are as follows:

- PLB unsupported transfer type
- PCI master abort generated (while PCI master)
- PCI target abort received (while PCI master)
- PCI target data bus parity error detection
- PCI master data bus parity error detection
- PCI target address parity error detection
- PLB bus error detection

The following sections describe in detail how these errors are generated, what actions are taken for each, and how to reset a given error.

19.7.1 PLB Unsupported Transfer Type

This error occurs when the bridge PLB slave encounters an unsupported PLB transfer type. *Table 19-11* outlines transfers not supported by the bridge PLB slave.

Table 19-11. PLB Unsupported Transfer Types

PLB Transaction	PCI Address Space
4- and 8-word line read/write	Nonmemory
16-word line read/write	Any
Burst	Nonmemory

Upon detection of this error, the bridge sets PCIC0_ERRSTS[PUR] = 1.

19.7.2 PCI Master Abort

This error is generated by the bridge PCI master when no target responds with $\overline{\text{PCIDevSel}}$ within the required time-out window and error detection is enabled. The bridge PLB slave may assert a PLB bus error signal on the PLB in response to this error, as explained below.

Two masks are associated with a PCI master abort. PCIC0_ERREN[MAEE] masks error reporting. If the error is detected, a PLB bus error signal is asserted if PCIC0_ERREN[MAEE] = 1. For reads, the bridge PLB slave still completes the transfer on the PLB, but drives 1s on the read data bus and the appropriate PLB bus error signal for each data beat. For posted writes, a PLB bus error is asserted for 1 cycle, asynchronously to the corresponding write data beat on the PLB. For connected writes, a PLB bus error signal is asserted with the data transfer, and the data is discarded. If PCIC0_ERREN[MAEE] = 0, error reporting is masked. No PLB bus error signal is asserted, regardless of the setting of PCIC0_ERREN[MEAE].

The following status bits are set:

1. If a master abort is signalled, PCIC0_STATUS[RMA] = 1. Setting of this field is non-maskable. Writing a 1 to PCIC0_STATUS[RMA] resets the field.
2. If master abort is detected as an error, PCIC0_ERRS[MEAE] is set to 1 to indicate an event that would cause a PLB bus error to be asserted by the bridge PLB slave, regardless of the setting of PCIC0_ERREN[MEAE]. This field can be reset by writing a 1 to PCIC0_ERREN[MEAE].
3. PCIC0_PLBBEAR and PCIC0_PLBBESRx are updated as follows:

   The address of the aborted request is saved in PCIC0_PLBBEAR if all PCIC0_PLBBESRx[MxAL] = 0 (PCIC0_PLBBEAR is unlocked). If all PCIC0_PLBBESRx[MxFL] = 0, PCIC0_PLBBESRx[MxET] = 0b101 to indicate a non-configured bank error; and PCIC0_PLBBESRx[MxRWS] is set to 0 on a write, 1 on a read. If PCIC0_ERREN[MAEE] = 0 or PCIC0_ERREN[MEAE] = 0, no PCIC0_PLBBEAR or PCIC0_PLBBESRx update is performed.

19.7.3 Bridge PCI Master Receives Target Abort While PCI Bus Master

This error is generated when the bridge PCI master receives a target abort while mastering a cycle on the PCI bus. Upon detection of this error, the bridge PLB slave may assert a PLB bus error signal on the PLB in response to this error, as explained below.

Two masks are associated with a target abort. PCIC0_ERREN[TAEE] masks error reporting. If the error is detected, a PLB bus error signal is asserted if PCIC0_ERREN[TAEE] = 1. For reads, the bridge PLB slave still completes the transfer on the PLB and drives the appropriate PLB bus error line for each data beat (note that for line reads, a PLB bus error signal is asserted for all data beats). For posted writes, if PCIC0_ERREN[TAEE] = 1, the bridge PLB slave asserts a PLB bus error for 1 cycle, asynchronously to the corresponding write data beat on the PLB. For connected writes, a PLB bus error signal is asserted with the data transfer, and the data is discarded. If PCIC0_ERREN[TAEE] = 0, error reporting is masked No PLB bus error signal is asserted, regardless of the setting of PCIC0_ERREN[MEAE]. If prefetching is occurring when a target abort is received, data preceding the target abort is kept in a prefetch buffer.

The following status bits are set:

1. If a target abort is received, PCIC0_STATUS[RTA] = 1. Setting this field is non-maskable. Writing a 1 to PCIC0_STATUS[RTA] clears the field.
2. If a target abort is detected as an error, PCIC0_ERRSTS[MEAE] = 1 to indicate an event that would cause the bridge PLB slave to assert a PLB bus error signal, regardless of the setting of PCIC0_ERREN[MEAE]. Writing a 1 to PCIC0_ERRSTS[MEAE] resets the field.
3. PCIC0_PLBBEAR and PCIC0_PLBBESRx are updated as follows:

   The address of the aborted request is saved in PCIC0_PLBBEAR if all PCIC0_PLBBESRx[MxAL] = 1 (PCIC0_PLBBEAR is unlocked). If all PCIC0_PLBBESRx[MxFL] = 1, PCIC0_PLBBESRx[MxET] = 0b101 to indicate a nonconfigured bank error, and PCIC0_PLBBESRx[MxRWS] is set to 0 on a write, or to 1 on a read. If PCIC0_ERREN[MAEE] = 0 or PCIC0_ERREN[MEAE] = 0, no PCIC0_PLBBEAR or PCIC0_PLBBESRx update is performed.

19.7.4 PCI Target Data Bus Parity Error Detection

This error is generated when the bridge PCI target detects a data bus parity error on write data from a PCI master doing a write cycle to PLB memory. PCI uses even parity.

Setting PCIC0_CMD[PER] = 0 masks this error.

The following status bits are set:

1. PCIC0_STATUS[DEPE] = 1 to indicate a PCI bus parity error. Setting this field is non-maskable. Writing a 1 to PCIC0_STATUS[DEPE] clears the field.
2. PCIC0_STATUS[SSE] = 1 if PCIC0_ERREN[WDPE] = 1. Writing a 1 to PCIC0_STATUS[SSE] clears the field.
3. PCIC0_ERRSTS[WDPE] = 1 if PCIC0_ERREN[WDPE] = 1. Writing a 1 to PCIC0_ERRSTS[WDPE clears the field.

19.7.5 PCI Master Data Bus Parity Error Detection

This error is generated when a data bus parity error is detected on the PCI bus during a cycle mastered by the bridge PCI master. The bridge PCI master checks parity on read cycles and samples $\overline{\text{PCIPErr}}$ on write cycles. The bridge PCI master may assert $\overline{\text{PCIPErr}}$ if the master detects a parity error on a read. PCI uses even parity.

Setting PCIC0_CMD[PER] = 0 masks this error. PCIC0_STATUS[DEPE] = 1. If a parity error is detected, writing a 1 to PCIC0_STATUS[DEPE] = 1 clears the field.

If PCIC0_ERREN[MEAE] = 1 and the error is detected as described, the PLB slave asserts a PLB error signal on the PLB in response to the error. For reads, a PLB bus error is asserted for each data beat in which bad parity was detected. For writes, a PLB bus error is asserted for each data beat in which bad parity was detected, but asynchronously to the actual transfer of write data on the PLB.
The following status bits are set:

1. PCIC0_STATUS[DEPE] = 1 if the bridge PCI master detects bad parity on read data, regardless of the state of PCIC0_CMD[PER]. Writing a 1 to PCIC0_STATUS[DEPE] clears the field.
2. If a data bus parity error is detected as an error, PCIC0_ERRSTS[MEAE] = 1 to indicate an event that would cause a PLB bus error signal to be asserted by the bridge PLB slave, regardless of the state of PCIC0_ERREN[MEAE]. Writing a 1 to PCIC0_ERRSTS[MEAE] = 1 clears the field.
3. PCIC0_PLBBEAR and the PCIC0_PLBBESRx are updated as follows:

   The address of the PCI transaction where parity errors occurred is saved in the PCIC0_PLBBEAR. PCIC0_PLBBEAR is set if all PCIC0_PLBBESRx[MxAL] = 1 (PCIC0_PLBBEAR is unlocked). If PCIC0_PLBBESRx[MxFL] = 1, PCIC0_PLBBESRx[MxET] = 0b001 to indicate a parity error, and PCIC0_PLBBESRx[MxRWS] is set to 0 on a write, 1 on a read. If PCIC0_CMD[PER] = 0 or PCIC0_ERRSTS[MEAE] = 0, no PCIC0_PLBBEAR or PCIC0_PLBBESRx update is performed.

Note: For clock ratios greater than 2:1 (independent of asynchronous/synchronous mode), the PCI bridge detects errors but does not assert a PLB bus error signal or log the error in PCIC0_PLBBEAR, PCIC0_PLBBESRx, and PCIC0_ERRSTS[MEAE].

19.7.6 PCI Address Bus Parity Error While PCI Target

This error occurs when a PCI address bus parity error is detected during the address phase of a cycle in which the bridge is the PCI target. PCI uses even parity.

Setting PCIC0_CMD[PER] = 0 masks this error. This error does not have an explicit status bit, however the following actions are taken:

1. PCIC0_STATUS[SSE] = 1 to indicate assertion of $\overline{\text{PCISErr}}$, if the mask at PCIC0_CMD[SE] = 1. Writing a 1 to PCIC0_STATUS[SSE] clears the field.
2. PCIC0_STATUS[DEPE] = 1 to indicate a PCI bus parity error, regardless of the state of PCIC0_CMD[PER]. PCIC0_STATUS[DEPE] = 1 when any type of PCI parity error is detected. Writing a 1 to PCIC0_STATUS[DEPE] clears the field.

19.7.7 PLB Master Bus Error Detection

This error occurs when the bridge PLB master detects a PLB bus error. If the bridge PLB master receives a PLB bus error while mastering a read, the master associates the error with the currently executing read. If the master receives a PLB bus error while mastering a write or while idle, the master associates the error with a write.

PLB bus error detection is masked by PCIC0_ERREN[MEDE] = 0. If PCIC0_ERREN[MEDE] = 1, PLB bus error detection is enabled.

PCIC0_ERREN[MERE] controls the response of the bridge PCI target to PLB bus error detection. If PCIC0_ERREN[MERE] = 10 or 11, the bridge PCI target will execute a target abort. If PCIC0_ERREN[MERE] = 01 or 11, the bridge PCI target asserts $\overline{\text{PCISErr}}$ and allows the transaction to continue. If PCIC0_ERREN[MERE] = 11, the bridge PCI target both target aborts and asserts $\overline{\text{PCISErr}}$.

The following status bits are set:

1. If the bridge PCI target executes a target abort, PCIC0_STATUS[STA] = 1. The setting of PCIC0_STATUS[STA] in such an event is non-maskable. Writing a 1 to PCIC0_STATUS[STA] clears the field.
2. f the bridge PCI target asserts $\overline{\text{PCISErr}}$, PCIC0_STATUS[SSE] = 1. The setting of PCIC0_STATUS[SSE] is non-maskable. Writing a 1 to PCIC0_STATUS[SSE] clears the field.

3. If the bridge PCI target asserts $\overline{\text{PCISErr}}$, PCIC0_ERRSTS[SARME] = 1 to indicate that the bridge PCI target asserted $\overline{\text{PCISErr}}$ in response to a received PLB bus error signal. The setting of PCIC0_ERRSTS[SARME] is non-maskable. Writing a 1 to PCIC0_ERRSTS[SARME] clears the field.
4. PCIC0_ERRSTS[MED] = 1 to indicate that the bridge PLB master received a PLB bus error signal. Setting of PCIC0_ERRSTS[MED] is non-maskable. Writing a 1 to PCIC0_ERRSTS[MED] clears the field.

19.8 PCI Power Management Interface

The PCI bridge supports *PCI Power Management Interface Specification* Revision 1.1 (PCI-PM).

19.8.1 Capabilities and Power Management Status and Control Registers

The PCI bridge has a capabilities structure in the PCI configuration space that indicates that the PCI bridge core is PCI Power Management capable. The capabilities structure includes the following registers:

- PCIC0_CAPID, value 0x01, indicates Power Management
- PCIC0_NEXTIPTR, points to next capabilities structure
- PCIC0_PMC, value 0x0202, indicates no specific capabilities
- PCIC0_PMCSR, indicates hold the current power state
- PCIC0_PMCSR_BSE, value 0x00, unused in PCI-to-PCI bridge
- PCIC0_Data, value 0x00, not used
- See *PCI Configuration Registers* on page 386 for details.

19.8.2 Power State Control

The current power management state is reported by reading PCIC0_PMCSR. The PCI bridge supports states D0, D1, D3hot, and D3cold. State D2 is not supported. When the state is not D0, the PCI bridge is masked from being a master or a memory or I/O target on the PCI bus. The PCI bridge can still be a config target. Thus, accesses claimed by the PCI bridge when in state D0 are no longer claimed, resulting in master aborts on the PLB or PCI if such an access is attempted. Note that this mask is independent of the state of the PCI Command register.

19.8.3 Changing Power States

The PCI bridge has two registers that control changing the power state. The host requests a change in the power state by writing to the PCIC0_PMCSR. The other register is PCIC0_PMSCRR, which provides a method of informing the local processor of a state change request and of preventing completion of the write to the PCIC0_PMCSR until the local processor indicates that it is ready for the state change.

Power state changes are handled as follows:

- If a host write to PCIC0_PMCSR requests an unsupported state change (such as a change to D2), the write is accepted but is ignored (no state change occurs).
- If a host write to PCIC0_PMCSR requests a change from D3hot to D0, the write is accepted. Then, the PCI bridge asserts the power management reset signal, which causes the entire SOC to be reset.

Note: The PCI bridge assumes that any requested state change from D3hot is always to D0.

- All other change requests are handled with the following sequence:
  1. The host requests a new power state by a PCI write to the PCIC0_PMCSR.

2. The host PCI write is retried (unless PCIC0_PMSCRR[DWE] = 0).
3. The host PCI write (retried or not) sets PCIC0_PMSCRR[SCR] = 1, which drives an interrupt to the local processor.
4. The local processor recognizes the interrupt. The local processor checks PCIC0_PMSCRR[SCR, REQST] to determine the nature of the request.
5. The local processor proceeds to power down the subsystem if the requested state is valid.
6. When the subsystem has been powered down and is ready to change state, the local processor clears the PCIC0_PMSCRR[SCR] and sets PCIC0_PMSCRR[APW] = 1.
7. When the host PCI write reoccurs:
   It is accepted.
   PCIC0_PMSCR is updated (only if the transition is valid).
   PCIC0_PMSCRR[APW] = 0, unless PCIC0_PMSCRR[DWE] = 0, in which case PCIC0_PMSCRR[APW] = 1 always.
   The PCI bridge enters a new power state.

The PCI bridge operates with the clock power management (CPM) logic to enable the bridge to be put into sleep mode under control of software. See *Clock and Power Management* on page 301 for discussion of the CPM function.

19.9 PCI Bridge Reset and Initialization

The following sections discuss resetting and initializing the PCI bridge.

19.9.1 Address Map Initialization

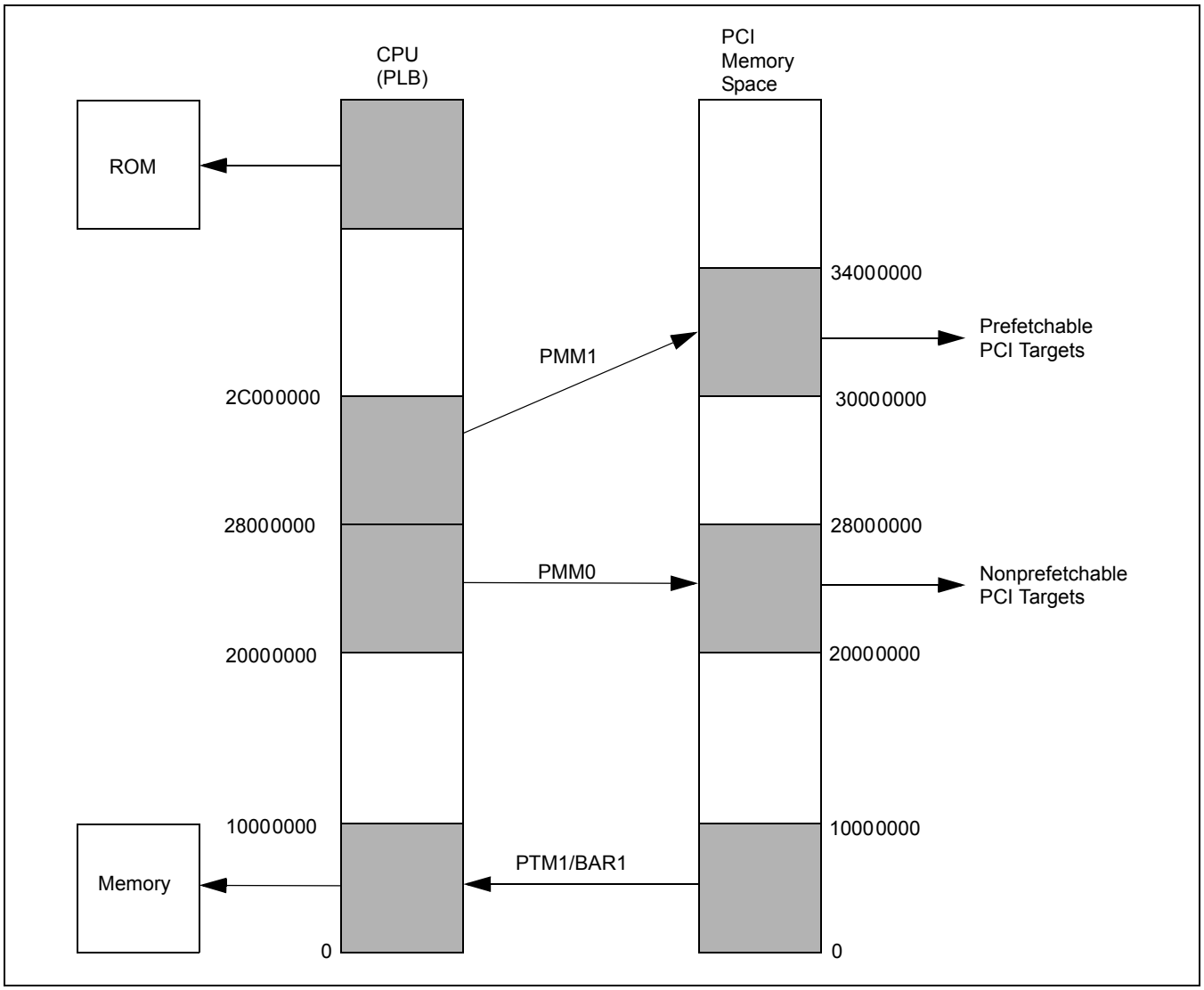
When the PCI bridge is the PCI master, it can generate memory, I/O, configuration, interrupt acknowledge, and special cycles. The method of cycle generation, and the associated address ranges, are fixed, except for memory cycles. PCI memory cycles are generated when the PCI bridge detects a cycle in one of three specified PLB address ranges. The sizes and address spaces of these ranges are specified using the PMM registers. Also, the address of the resulting PCI memory cycle can be an offset from the PLB address (address translation occurs). This translation is also specified in the PMM registers. The PMM registers *do not* default to usable values following reset; they must be initialized before attempting to generate PCI memory cycles.

When the PCI bridge is a target on the PCI bus, the PCI bridge can respond to memory cycles. The memory cycle address ranges that the PCI bridge responds to are specified in PCIC0_PTM1BAR and PCIC0_PTM2BAR. These registers are typically initialized as part of the standard PCI initialization process.

Figure 19-60 shows the desired address map. System memory resides from 0x00000000–0x0FFF FFFF in the CPU/PLB address space, which is accessible from the PCI in the same address space (PCI bridge as a memory target) as defined by PTM1/BAR1. PTM2/BAR2 is disabled in this example. The CPU/PLB master has two spaces

in which to access PCI Memory space. Range 0 is 0x20000000 to 0x27FFFFFF and is mapped to the same address on the PCI bus, and is nonprefetchable. Range 1 is 0x28000000 to 0x2BFFFFFF, and is translated to address range 0x30000000 to 0x33FFFFFF of PCI memory space. Range 2 is disabled.

Figure 19-60. Example Address Map



The following register values provide the address map shown in *Figure 19-60*:

Table 19-12. Address Map Register Values

Register Name	Value	Comments
PCIL0_PMM0LA	0x20000000	
PCIL0_PMM0MA	0xF8000001	128MB; enabled; read prefetching not allowed.
PCIL0_PMM0PCILA	0x20000000	
PCIL0_PMM0PCIHA	0x00000000	
PCIL0_PMM1LA	0x28000000	
PCIL0_PMM1MA	0xFC000003	64MB; enabled; prefetching allowed.

Table 19-12. Address Map Register Values (continued)

Register Name	Value	Comments
PCIL0_PMM1PCILA	0x30000000	
PCIL0_PMM1PCIHA	0x00000000	
PCIL0_PMM2LA	0x00000000	
PCIL0_PMM2MA	0x00000000	Not enabled.
PCIL0_PMM2PCILA	0x00000000	
PCIL0_PMM2PCIHA	0x00000000	
PCIL0_PTM1MSr	0xF0000001	256MB; enabled.
PCIL0_PTM1LA	0x00000000	
PCIL0_PTM2MS	0x00000000	Not enabled.
PCIL0_PTM2LA	0x00000000	
PCIC0_PTM1BAR	0x00000008	PCI memory space; address decode starts at PCI address 0x0000 0000.
PCIC0_PTM2BAR	0x00000000	

19.9.2 Other Configuration Register Initialization

Additional register initialization is required, as follows:

- Error handling is initially disabled (error detection is masked). If error handling is to be enabled, PCIC0_ERREN must be initialized appropriately.
- PCIC0_BRDGOPT1 contains options for controlling the PLB. Its default values can be used.
- PCIC0_BRDGOPT2 contains options for controlling the PCI bus. Its default values assume that the PCI is run synchronously to the PLB, and that the PCI bridge is the primary host bridge. If the PCI bridge is used otherwise, the values must be changed accordingly.

Note: PCI devices that are targets and support delayed reads may be attached to the PCI bus. To ensure that the PPC440EP does not deadlock when accessing such devices, the PLB priority of the PCI and of all the PLB masters that access PCI space must be set to the same value, and the arbitration mode within the PLB arbiter must be set to fair mode. Since the processor data cache unit has dynamic PLB priority, the higher priority value of the DCU must be the same as the PCI. For additional information on setting PLB priorities see *PLB Master and Slave Assignments* on page 64.

19.9.3 Target Bridge Initialization

The PCI bridge can also respond as a configuration target; however, the PCI bridge only responds as a configuration target when the PCIIDSel pin is attached, rather than pulled inactive. Note that if the size and local address of these ranges are not strapped to desired values at reset, the local CPU must specify them by setting the PTM Memory Size and Local Address registers before initializing PCIC0_PTM1BAR and PCIC0_PTM2BAR.

The local CPU must update the following registers (if the default value is not suitable or they were not strapped to appropriate values at reset) before setting the Host Config Enable bit:

- The address map registers (see *Address Map Initialization* on page 408)
- PCIC0_VENDID
- PCIC0_DEVID

- PCIC0_REVID
- PCIC0_CLS
- PCIC0_SBSYSID
- PCIC0_SBSYSVID

19.9.4 Local Processor Boot from PCI Memory

The PCI bridge has a mode that enables a PLB master to access a PCI memory range without initial configuration cycles. This mode is enabled when CPC0_ = 1. System designers can use this mode to enable a processor to access a boot ROM in PCI memory space.

The PCI bridge comes out of reset with PMM0 enabled and programmed for the address range, 0xFFFE0000–0xFFFFFFFF. Also, PCIC0_CMD[ME] = 1 after reset. Note that enabling PCI boot mode does not prevent subsequent updates to the PMM0 registers.

Note: The PPC440EP allows booting from PCI memory. See *Bootstrap Controller* on page 207 for more information.

19.9.5 Type 0 Configuration Cycles for Other Devices

Twenty-one devices can be accessed using the PCIIDSel mechanism. The PCI master asserts 1 bit of AD(31:11) for type 0 configuration cycles based on the value in the Device Number field. The mapping is as follows:

- If device number is 1, AD(11) is asserted
- If device number is 2, AD(12) is asserted
- .
- .
- .
- If device number is 21, AD(31) is asserted

If device number contains a value of 22–31, no bit of AD(31:11) is asserted.

19.10 Timing Diagrams

This section contains timing diagrams of several different PCI bridge operations. The following paragraphs describe each diagram in detail. Each description assumes basic knowledge of PCI and PLB protocols.

Each operation is shown in both synchronous and asynchronous modes. The PCI is clocked at 33 MHz in synchronous mode and 66 MHz in asynchronous mode. The PLB is clocked at 100 MHz in all cases.

The SDRAM uses a 32-bit, PC100 memory interface configured for $\overline{\text{CAS}}$ latency of 2, command leadoff of 2, and RAS to CAS delay (T_{rcd}) of 2. All memory accesses are page idle, unless indicated otherwise.

Note: The PLB signals shown in the following timing diagrams are not externally accessible. They are included for information purposes and as an aid to understanding the PCI operations. For more information on these signals, refer to *Processor Local Bus Architecture Specifications*.

19.10.1 PCI Timing Diagram Descriptions

The following sections briefly describe each of the timing diagrams that follow the descriptions. Each description covers both the asynchronous and synchronous clocking modes for that operation. The timing diagrams then follow with all of the asynchronous diagrams grouped together followed by all of the synchronous diagrams grouped together.

19.10.1.1 PCI Master Burst Read From SDRAM

Figure 19-61 PCI Master Burst Read From SDRAM on page 413 (asynchronous) and *Figure 19-83 PCI Master Burst Read From SDRAM* on page 435 (synchronous) show a PCI Master executing a 128-byte Read Multiple from SDRAM. PCI bridge retries the initial request and performs a delayed read. PCI bridge executes a variable-length, doubleword read burst on PLB to SDRAM, filling its 96-byte read buffer. The read is a page hit. When the PCI master re-requests its read, PCI bridge begins bursting out of its read buffer, while continuing to prefetch from SDRAM.

19.10.1.2 PCI Master Burst Write To SDRAM

Figure 19-65 PCI Master Burst Write To SDRAM on page 417 and *Figure 19-90 PCI Master Burst Write To SDRAM* on page 442 show a PCI Master executing a 128-byte Write to SDRAM. PCI bridge accepts several beats of data into its 64-byte write buffer before executing variable-length, doubleword write bursts on PLB to SDRAM. The final write burst is fixed-length, since the PCI write has completed, and PCI bridge knows the exact burst length.

19.10.1.3 CPU Read From PCI Memory Slave, Nonprefetching

In *Figure 19-69 CPU Read From PCI Memory Slave, Nonprefetching* on page 421 and *Figure 19-96 CPU Read From PCI Memory Slave, Nonprefetching* on page 448, a PLB Master (CPU) executes a single-beat 64-bit read from a region of PCI memory marked as nonprefetchable. PCI bridge responds as a 32-bit PLB slave, so the CPU executes conversion cycles for each read. PCI bridge executes a single-beat PCI read for both PLB read requests.

19.10.1.4 CPU Read From PCI Memory Slave, Prefetching

In *Figure 19-71 CPU Read From PCI Memory Slave, Prefetching* on page 423 and *Figure 19-98 CPU Read From PCI Memory Slave, Prefetching* on page 450, a PLB Master (CPU) executes 8, 64-bit, single-beat reads from a region of PCI memory marked as prefetchable. The first PLB read causes PCI bridge to execute a 64-byte Read Multiple to fill its 64-byte read prefetch buffer. The data for subsequent PLB reads is provided from the read buffer and no PCI cycles are generated. PCI bridge responds as a 32-bit PLB slave, so the CPU executes conversion cycles for each read.

19.10.1.5 CPU Write To PCI Memory Slave

Figure 19-75 CPU Write To PCI Memory Slave on page 427 and *Figure 19-102 CPU Write To PCI Memory Slave* on page 454 show a PLB Master (CPU) executing 4, 64-bit, single-beat writes to PCI memory. PCI bridge responds as a 32-bit slave, so the CPU executes conversion cycles for each write. PCI bridge posts the writes in its 4-entry write buffer, and executes a PCI single-beat Memory Write for each request.

19.10.1.6 PCI Memory To SDRAM DMA Transfer

Figure 19-78 PCI Memory To SDRAM DMA Transfer on page 430 and *Figure 19-107 PCI Memory To SDRAM DMA Transfer (Continued)* on page 459 show a DMA transfer of data from PCI memory to SDRAM. The DMA PLB Master executes a 4-doubleword read burst to PCI bridge followed by a 4-doubleword write burst to SDRAM. For the read, PCI bridge executes a 32-byte PCI Read Line.

19.10.1.7 SDRAM To PCI Memory DMA Transfer

Figure 19-80 SDRAM To PCI Memory DMA Transfer on page 432 and *Figure 19-109 SDRAM To PCI Memory DMA Transfer* on page 461 show a DMA transfer of data from SDRAM to PCI memory. The DMA PLB Master executes a 4-doubleword read burst from SDRAM followed by a 4-doubleword write burst to PCI memory. PCI bridge then executes a 32-byte write on the PCI bus.

19.10.2 Asynchronous

The following diagrams are for asynchronous clocking mode. Note that all of the diagrams flow across multiple pages. Each diagram begins with cycle 1 on the left facing page.

Figure 19-61. PCI Master Burst Read From SDRAM

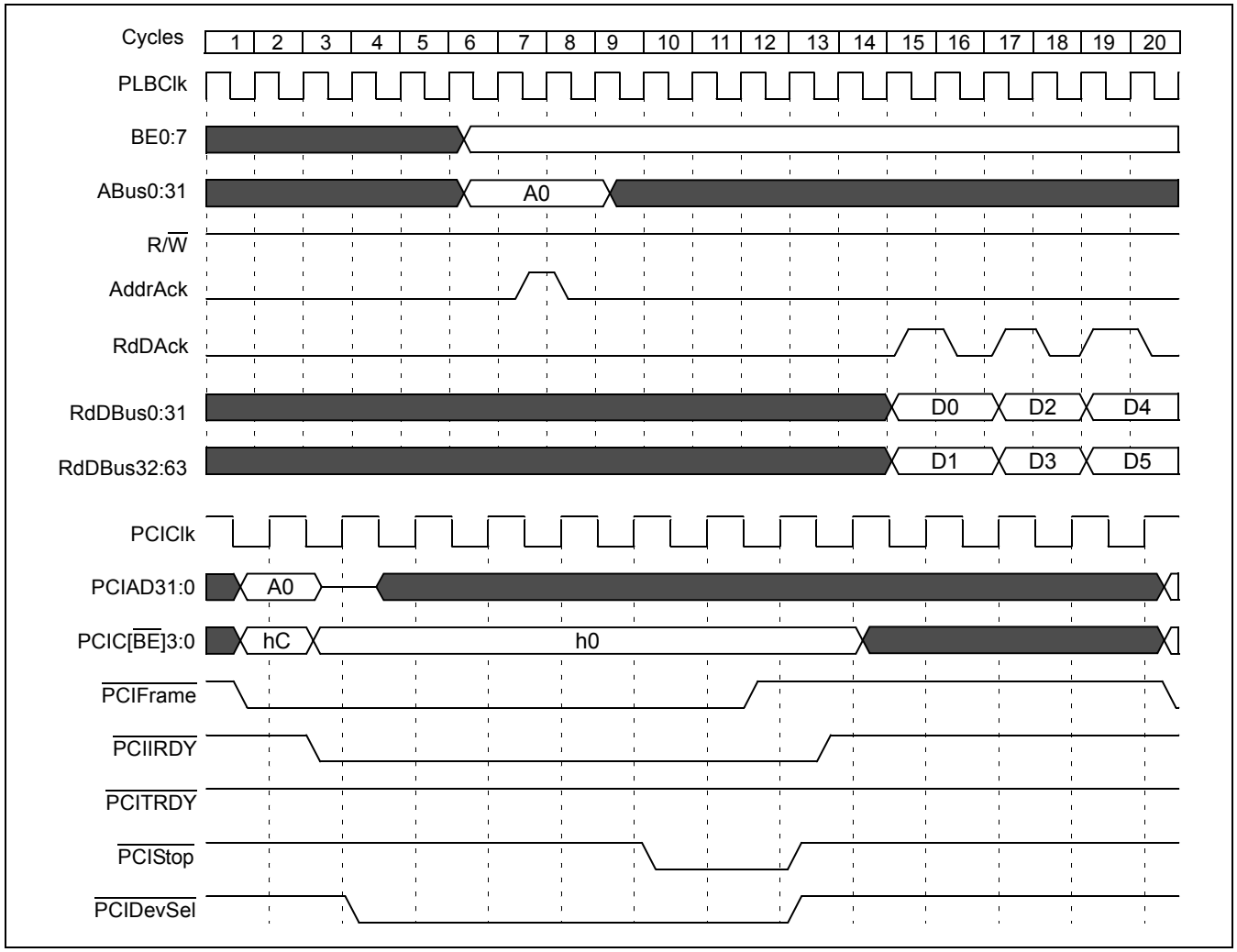


Figure 19-62. PCI Master Burst Read From SDRAM (Continued)

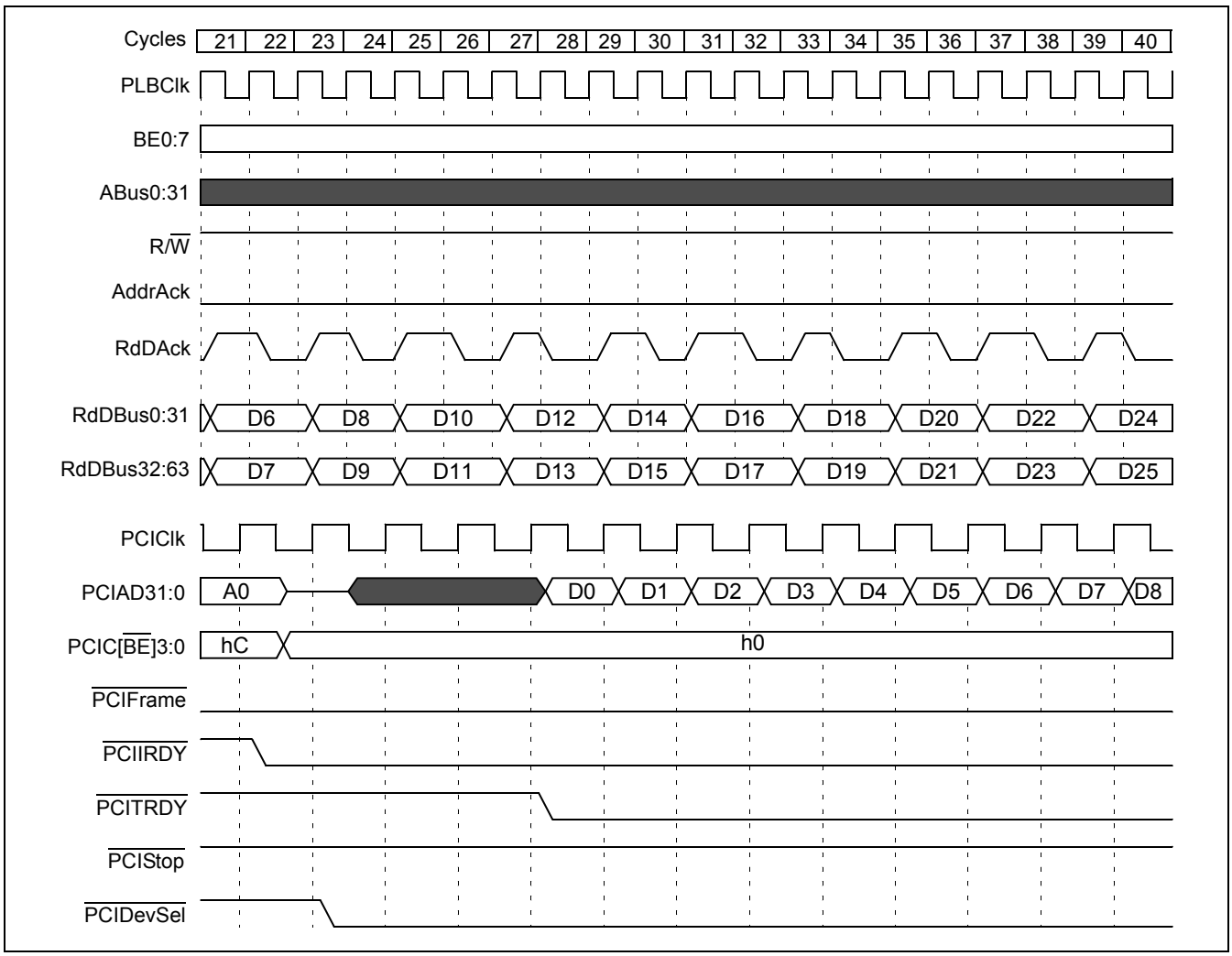


Figure 19-63. PCI Master Burst Read From SDRAM (Continued)

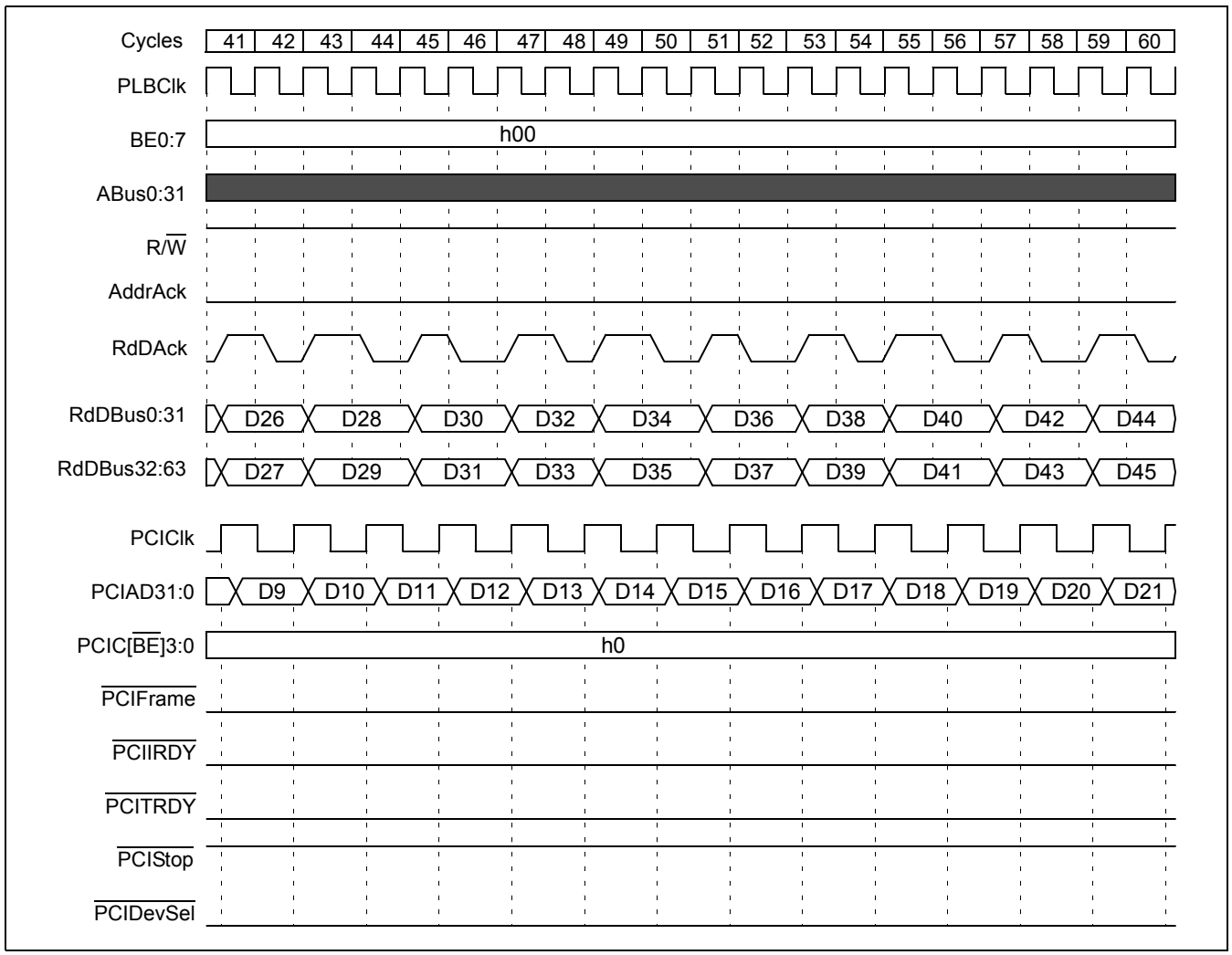


Figure 19-64. PCI Master Burst Read From SDRAM (Continued)

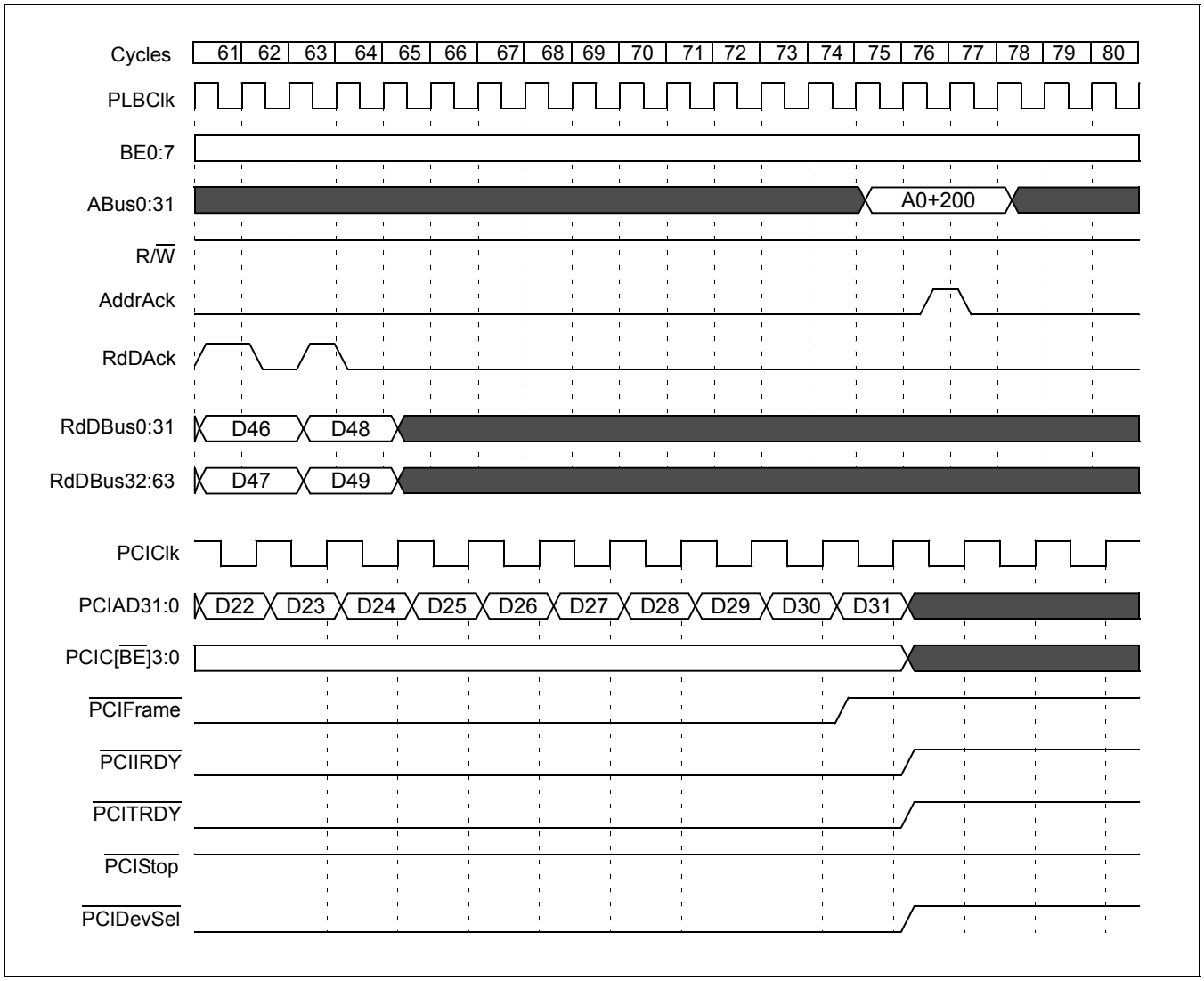


Figure 19-65. PCI Master Burst Write To SDRAM

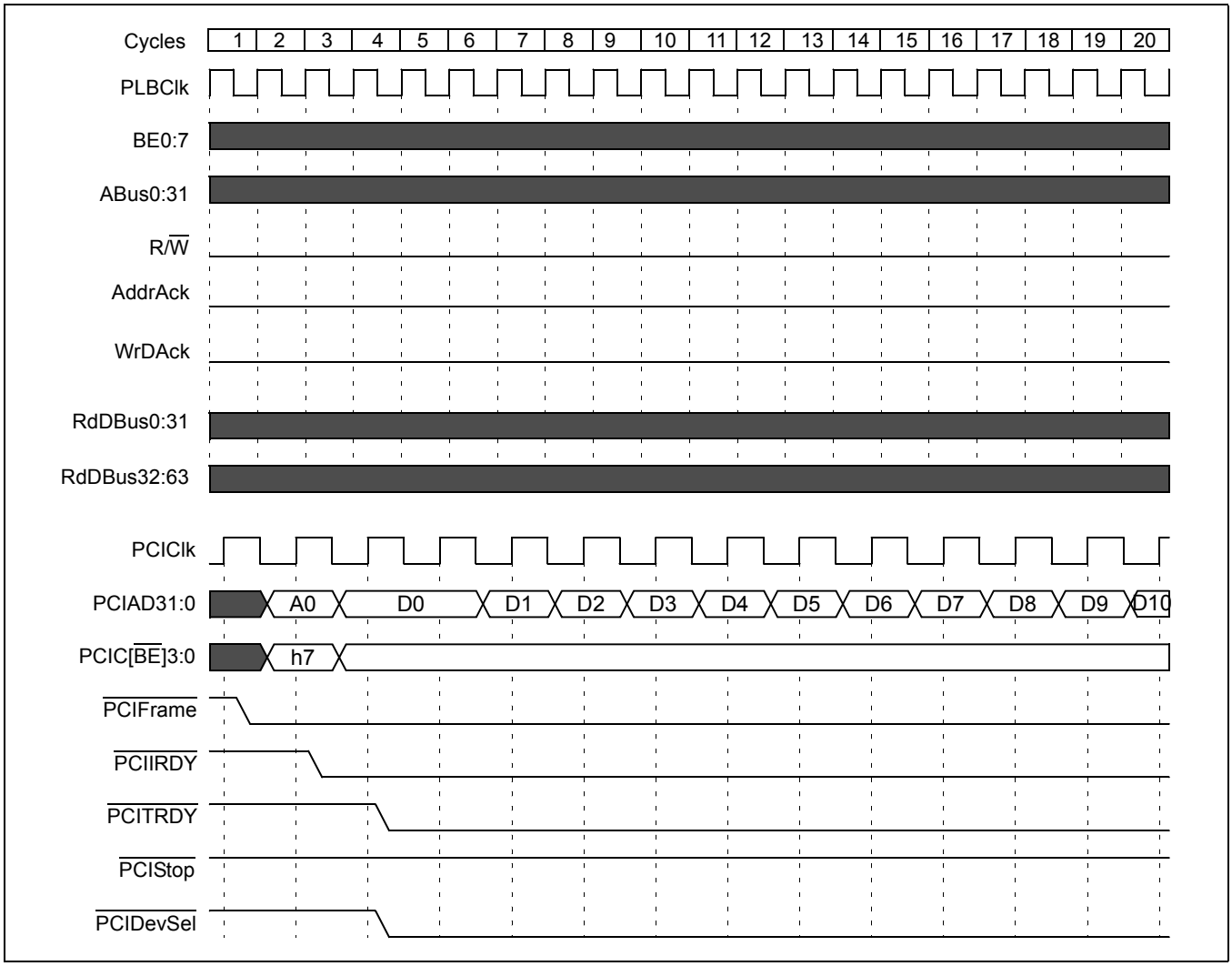


Figure 19-66. PCI Master Burst Write To SDRAM (Continued)

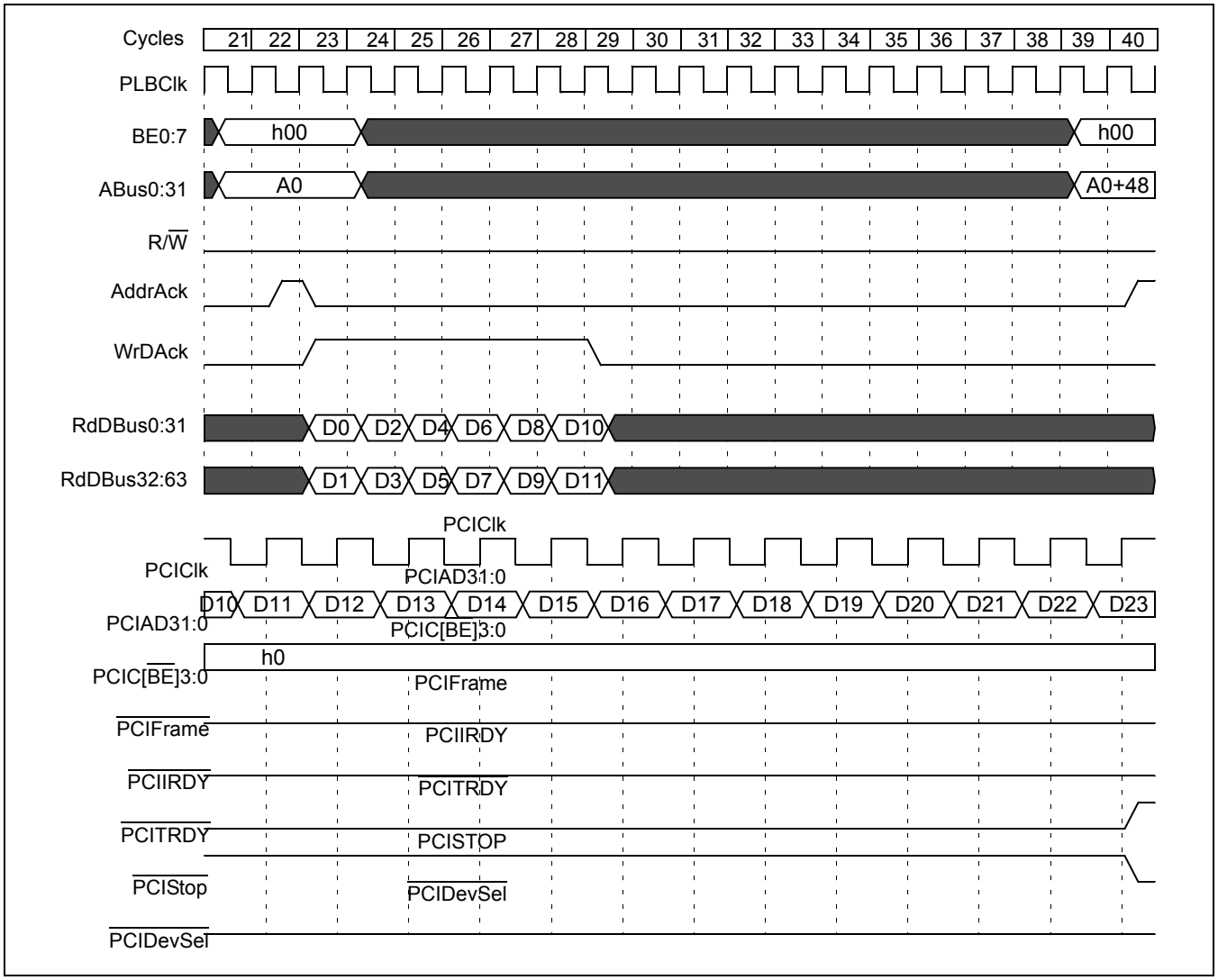


Figure 19-67. PCI Master Burst Write To SDRAM (Continued)

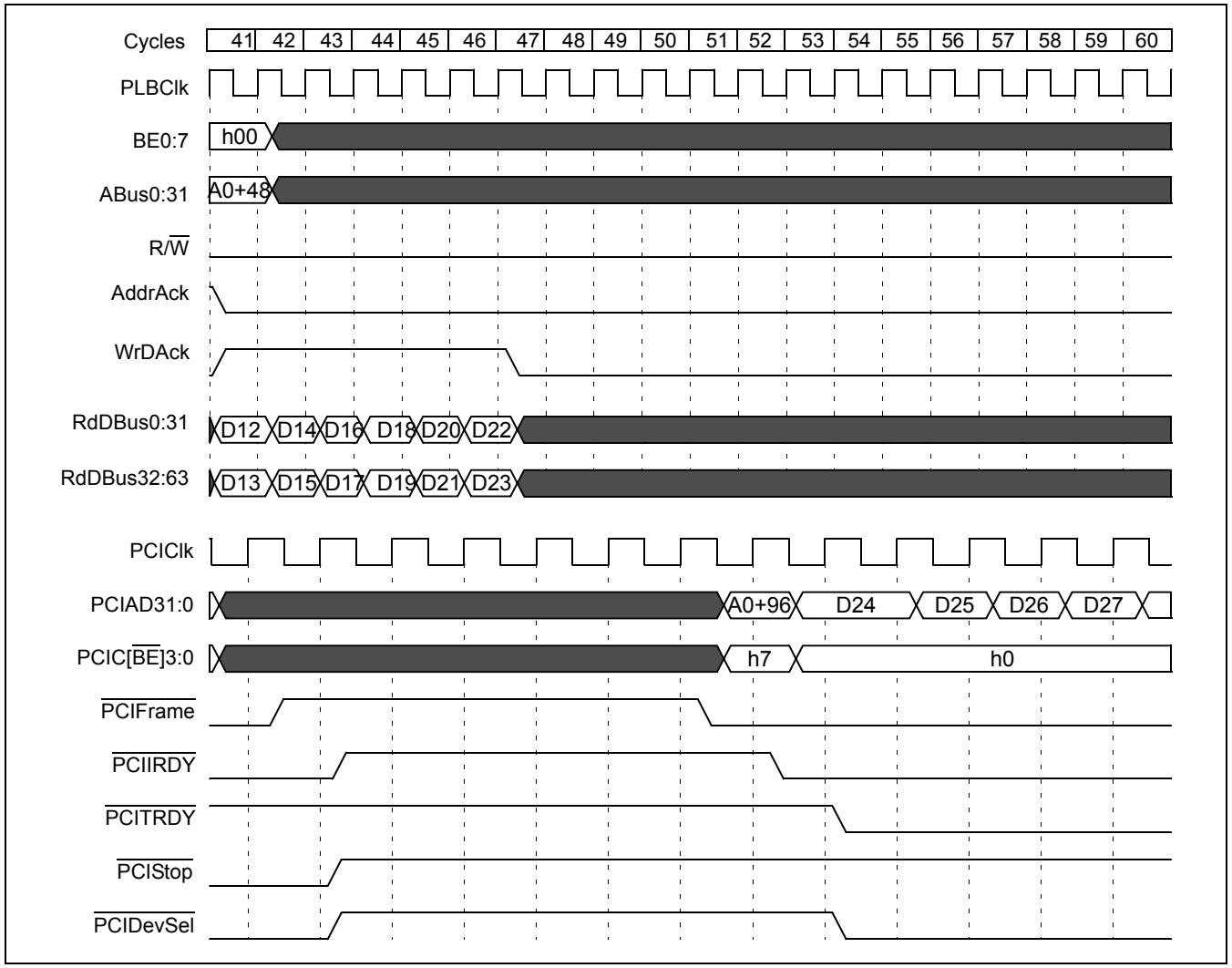


Figure 19-68. PCI Master Burst Write To SDRAM (Continued)

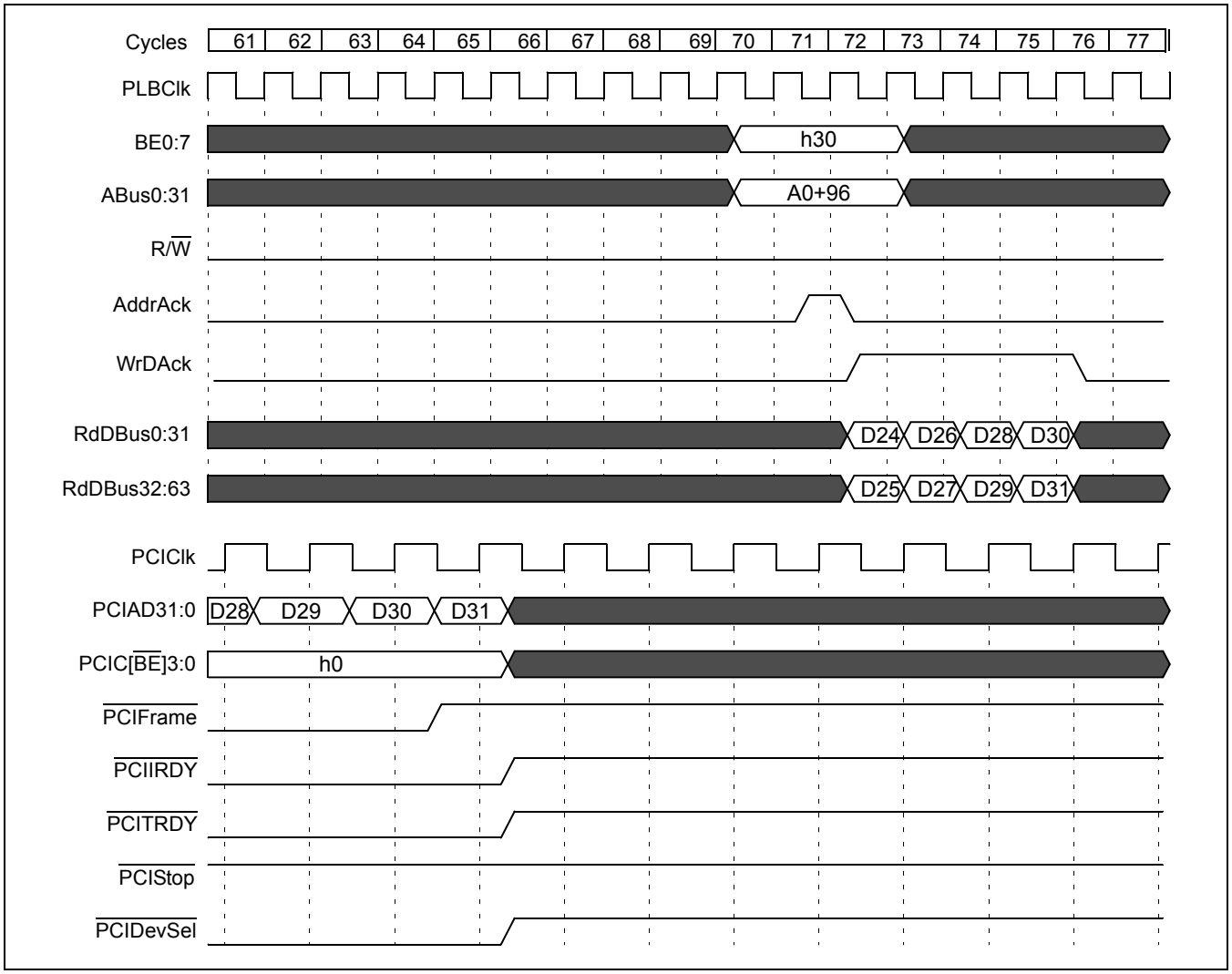


Figure 19-69. CPU Read From PCI Memory Slave, Nonprefetching

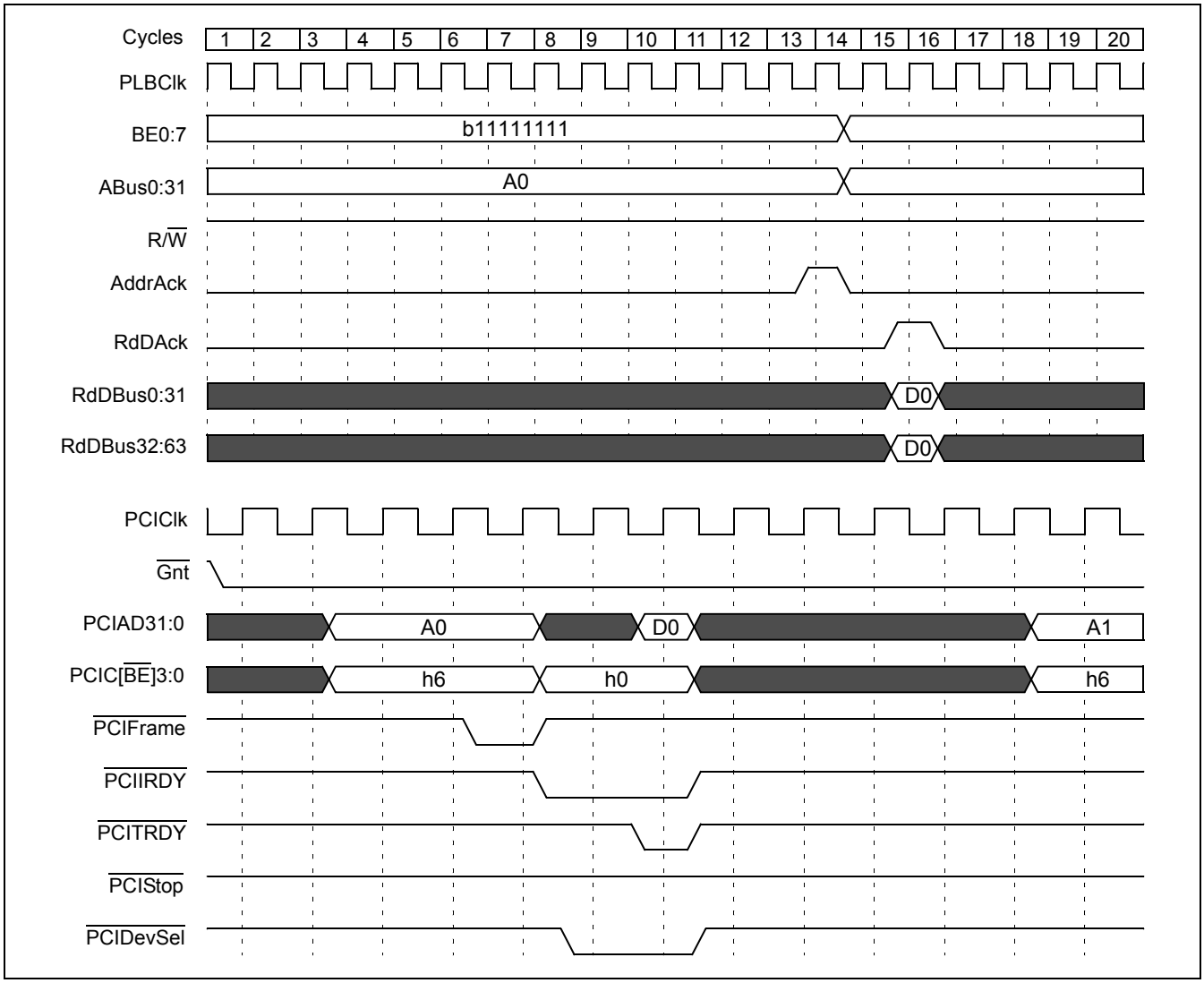


Figure 19-70. CPU Read From PCI Memory Slave, Nonprefetching (Continued)

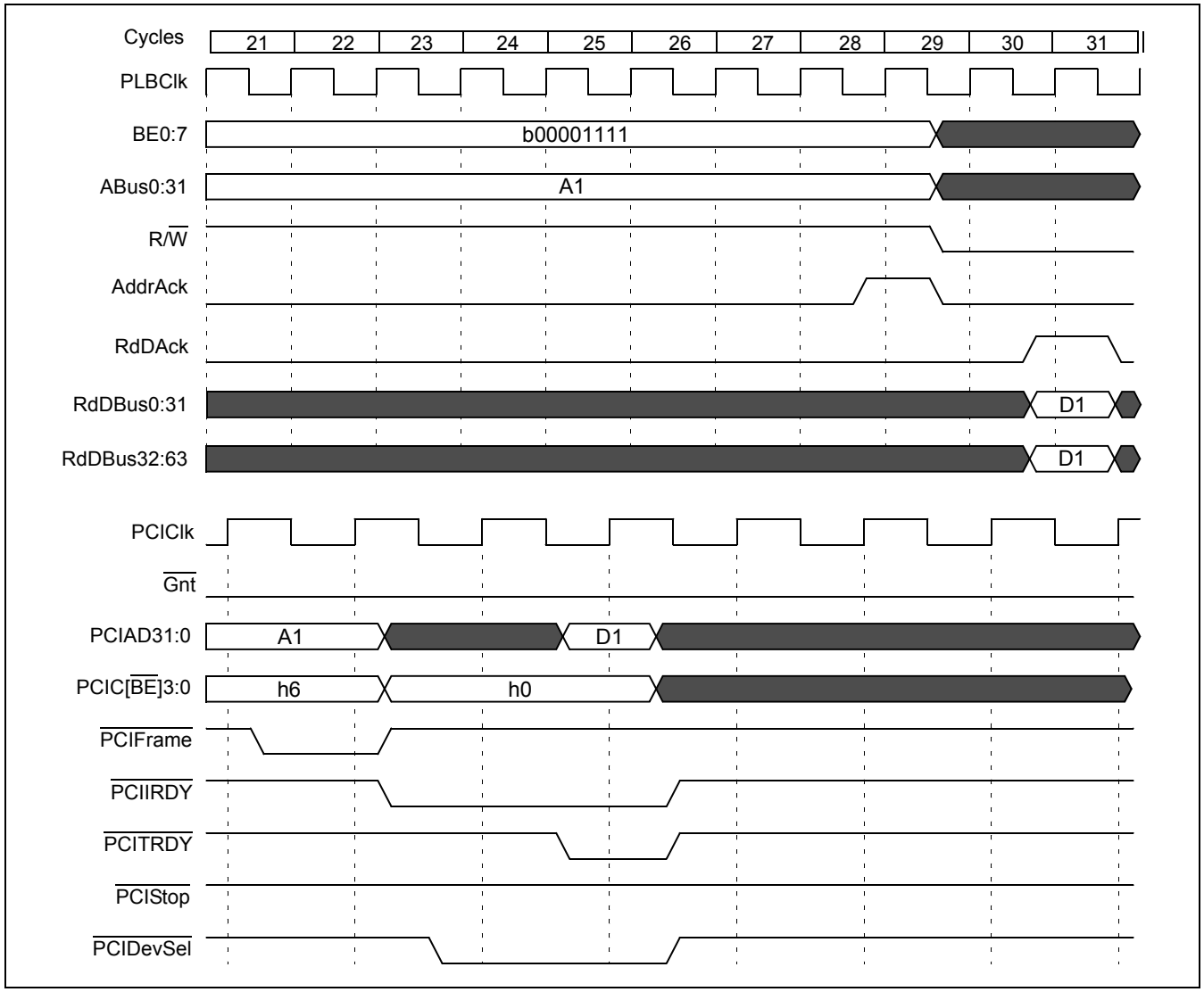


Figure 19-71. CPU Read From PCI Memory Slave, Prefetching

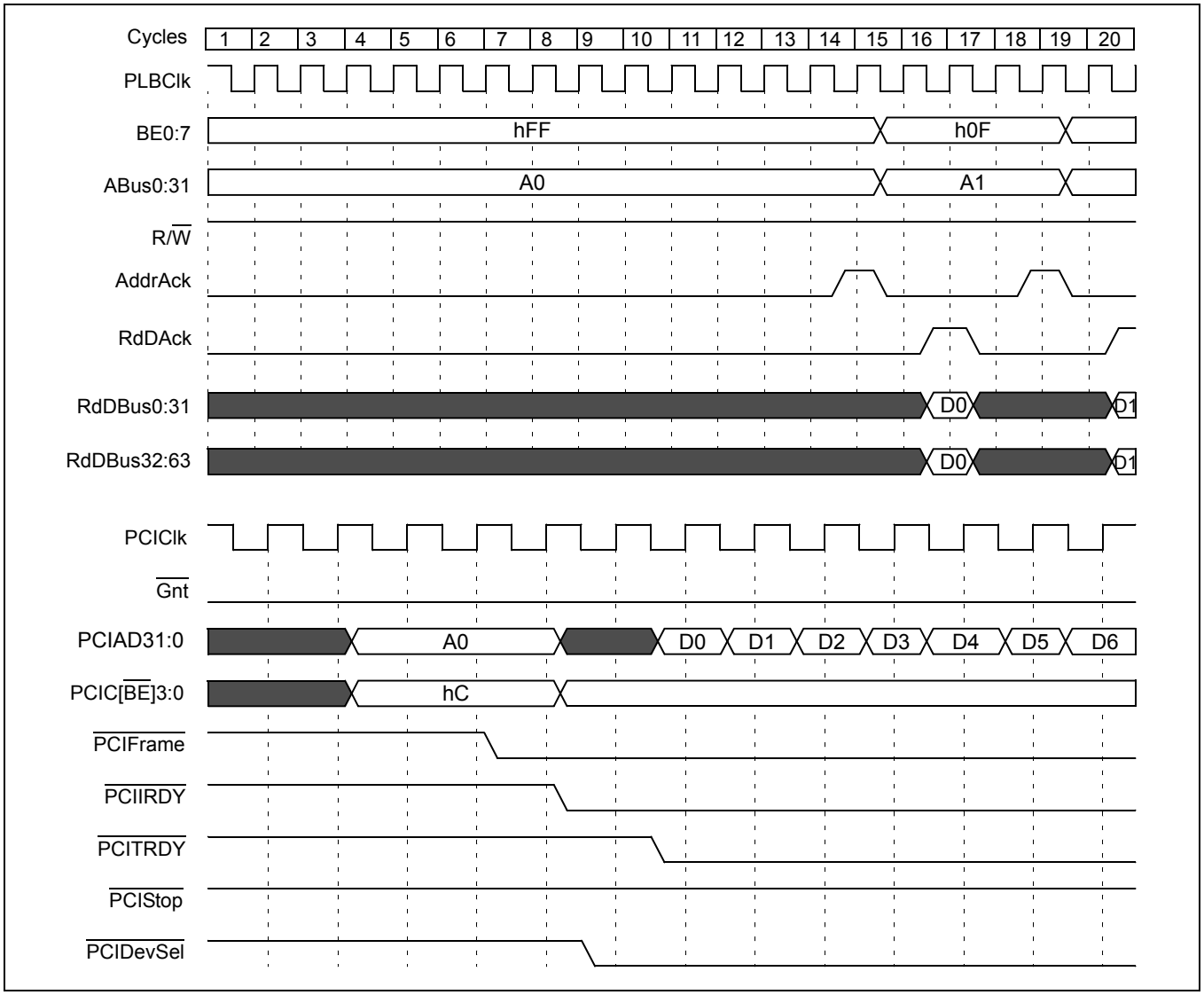


Figure 19-72. CPU Read From PCI Memory Slave, Prefetching (Continued)

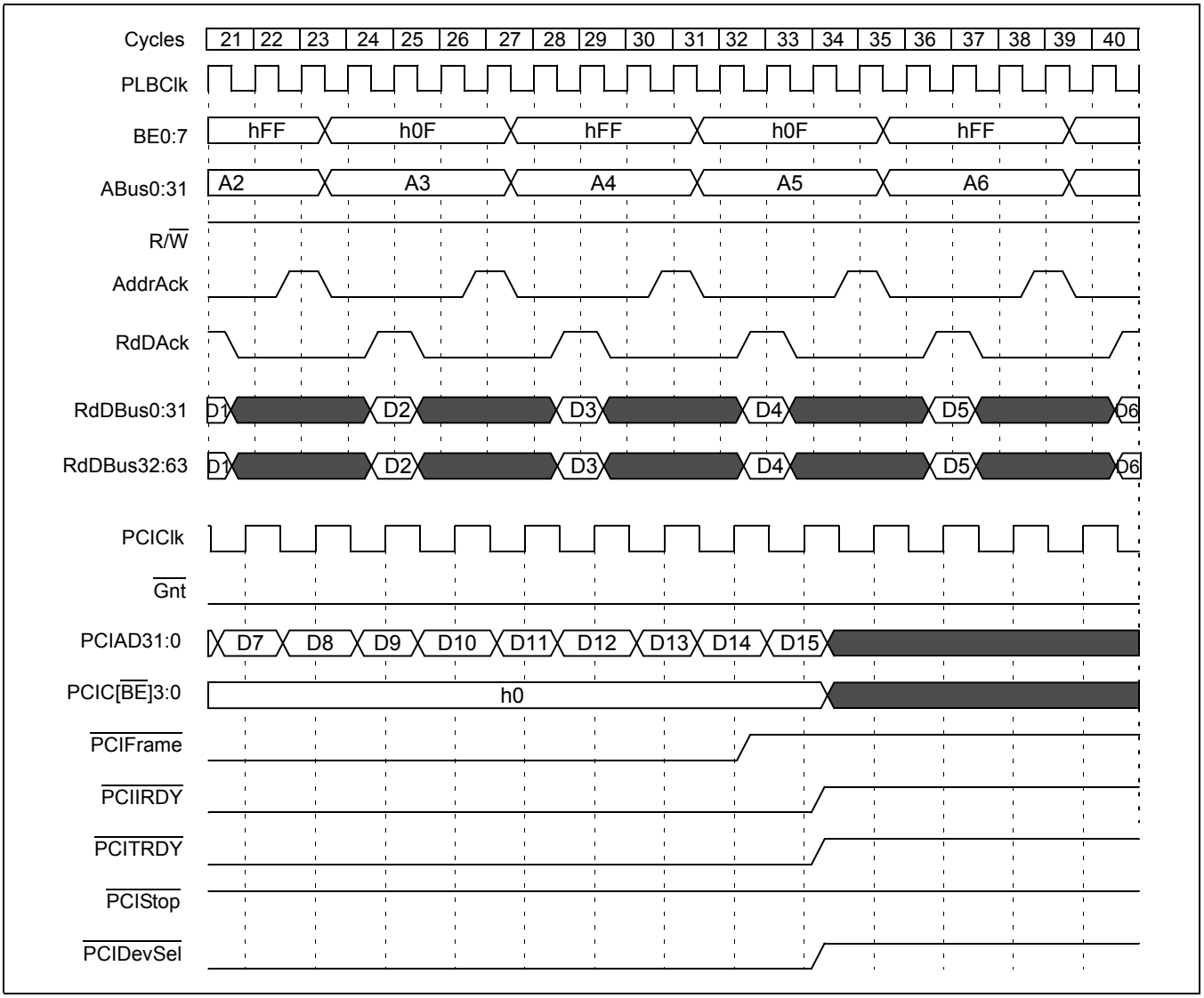


Figure 19-73. CPU Read From PCI Memory Slave, Prefetching (Continued)

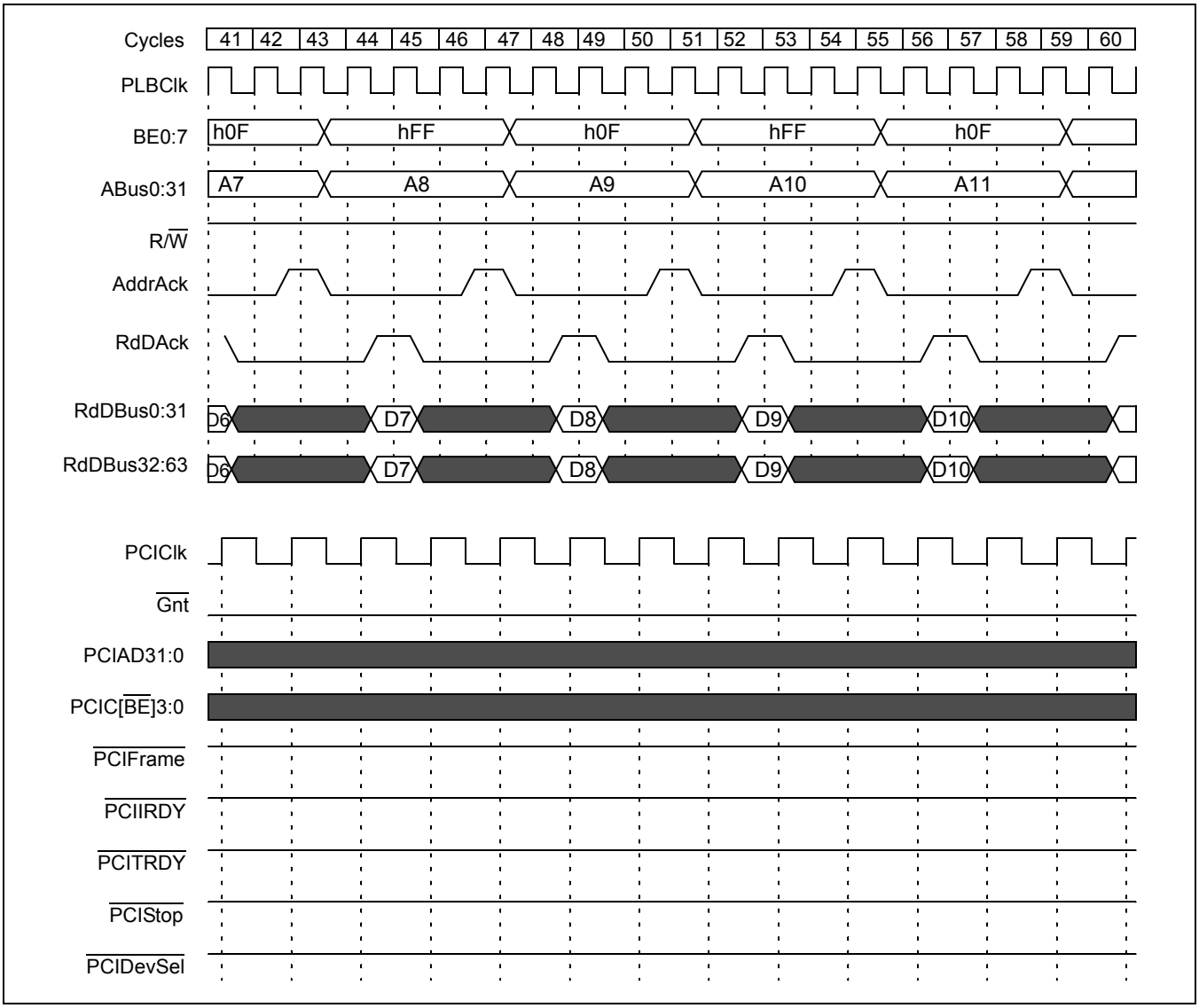


Figure 19-74. CPU Read From PCI Memory Slave, Prefetching (Continued)

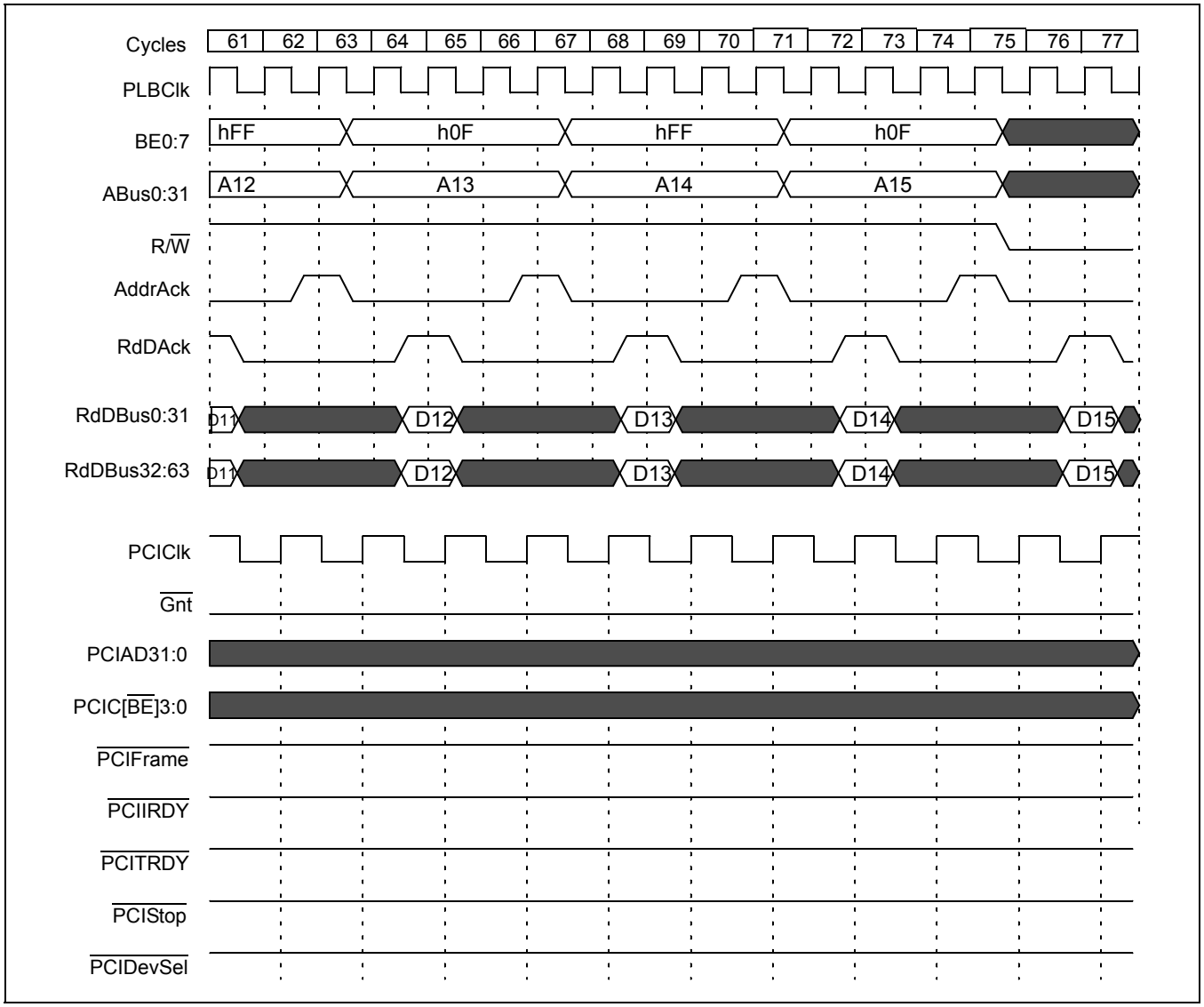


Figure 19-75. CPU Write To PCI Memory Slave

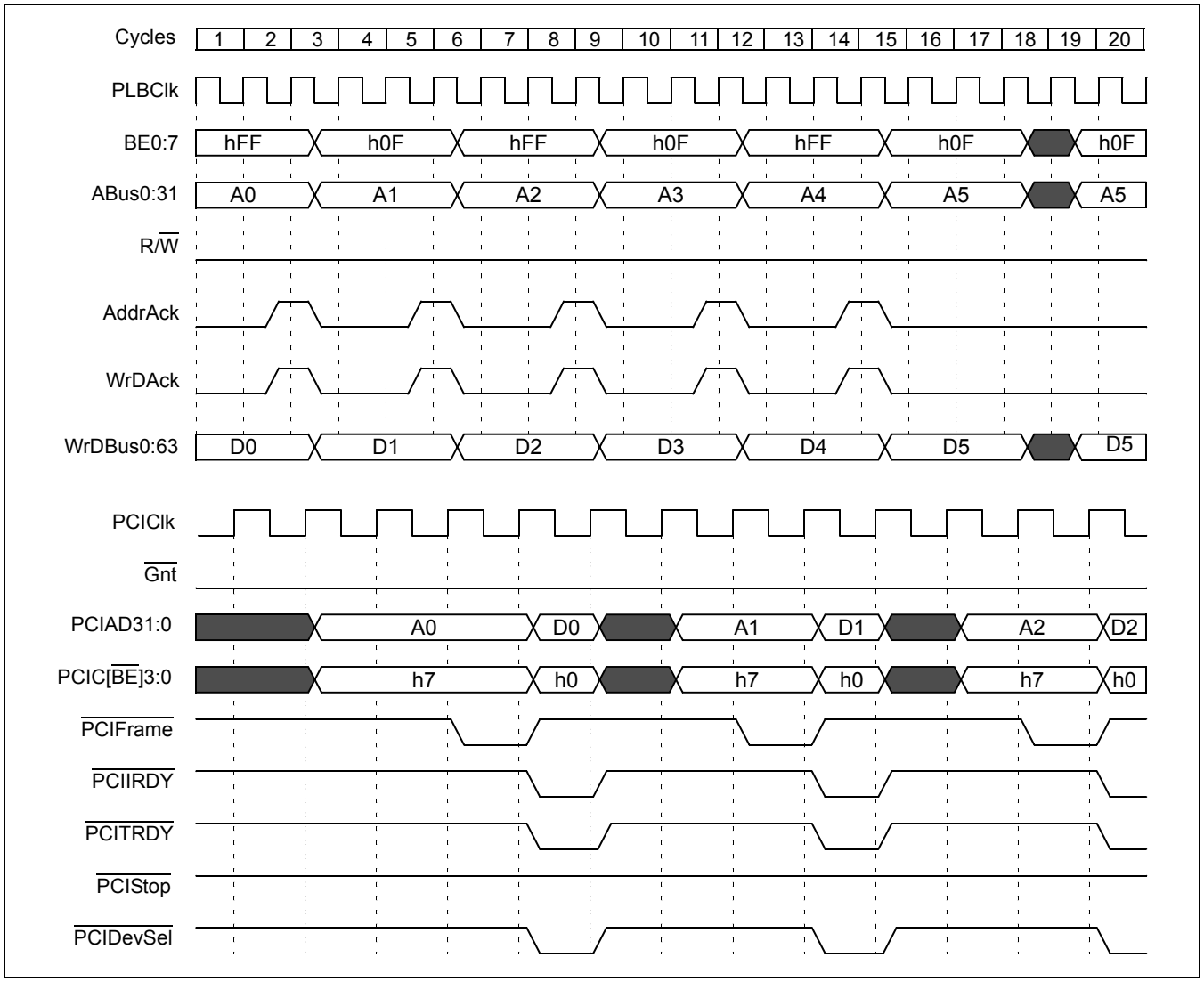


Figure 19-76. CPU Write To PCI Memory Slave (Continued)

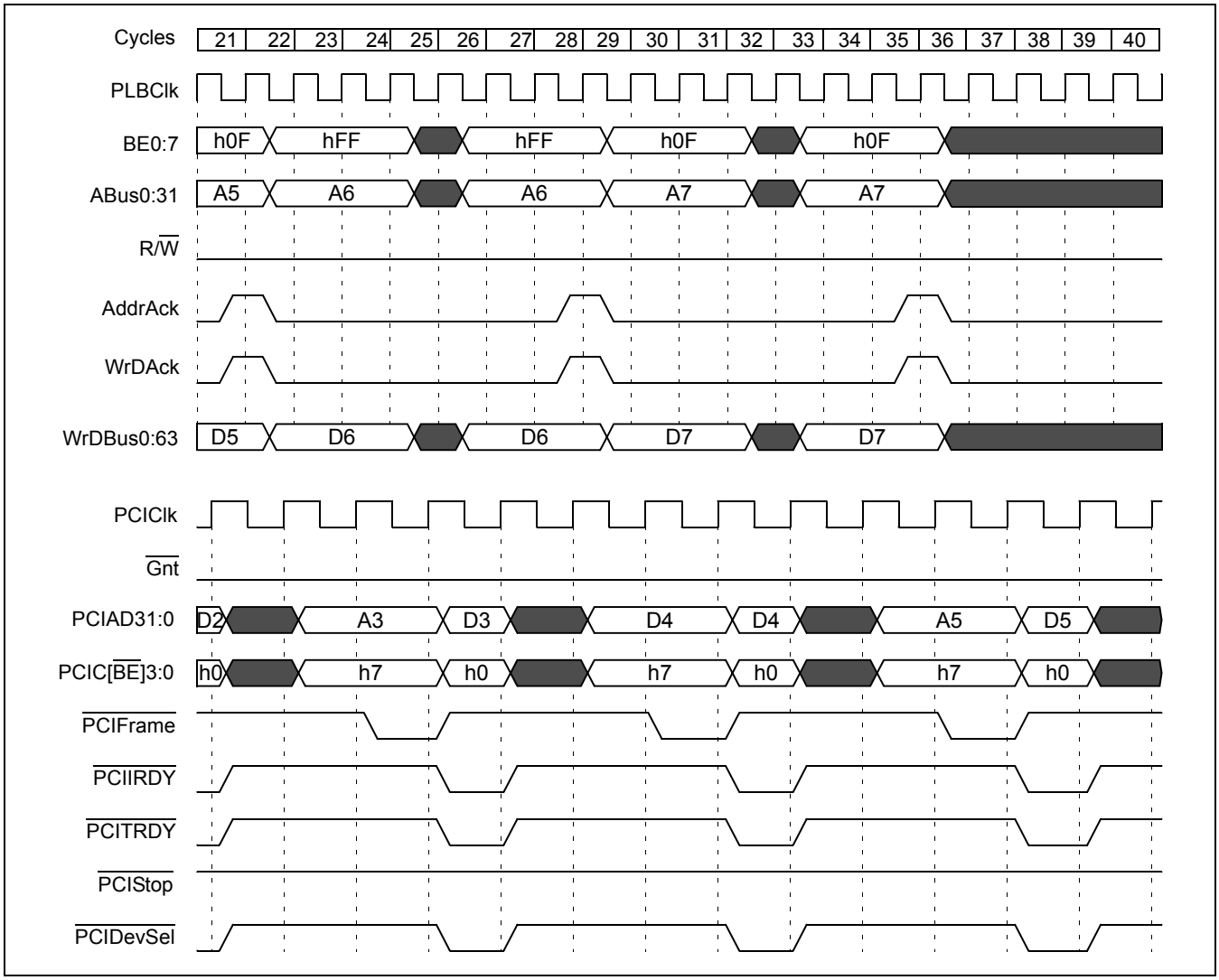


Figure 19-77. CPU Write To PCI Memory Slave (Continued)

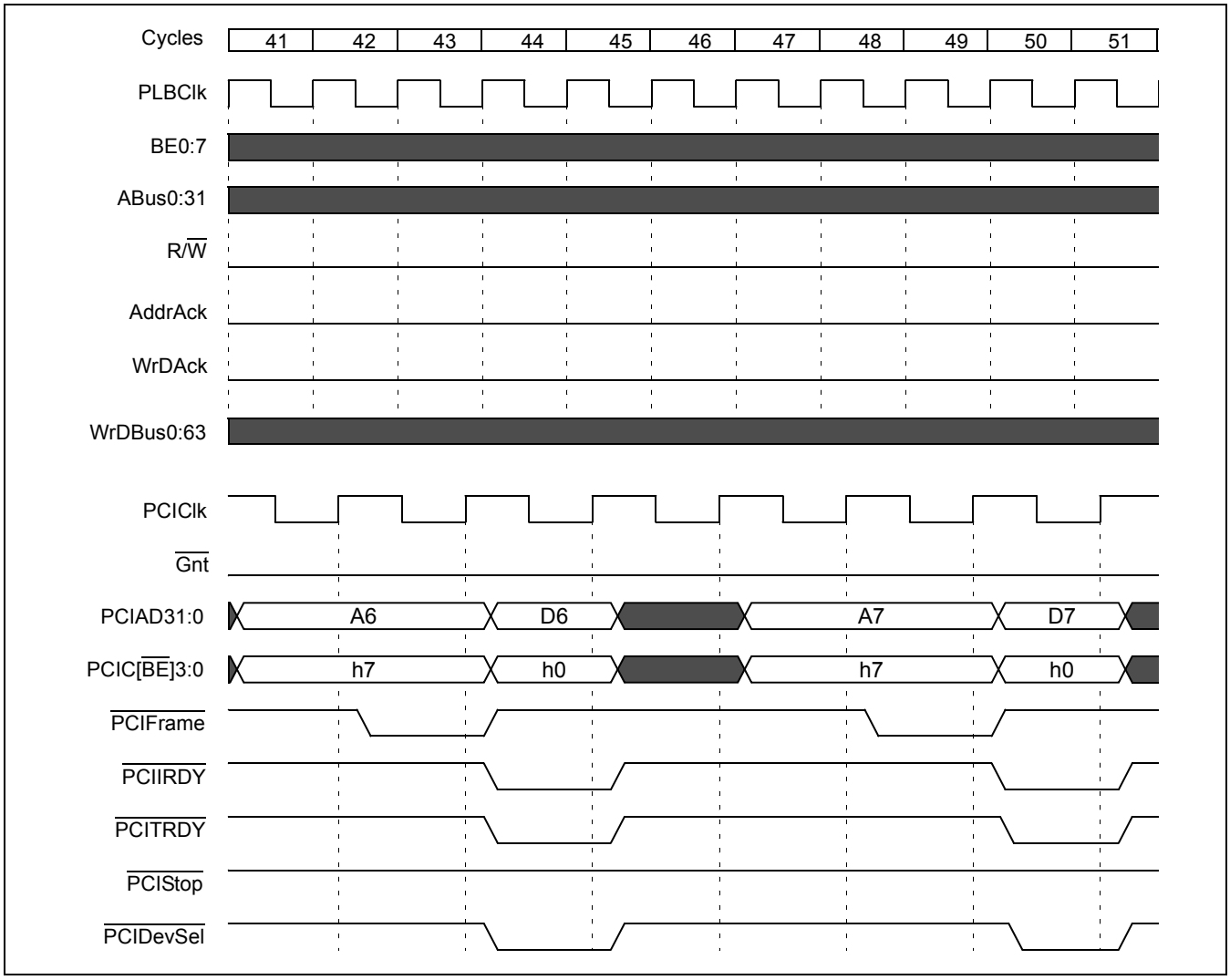


Figure 19-78. PCI Memory To SDRAM DMA Transfer

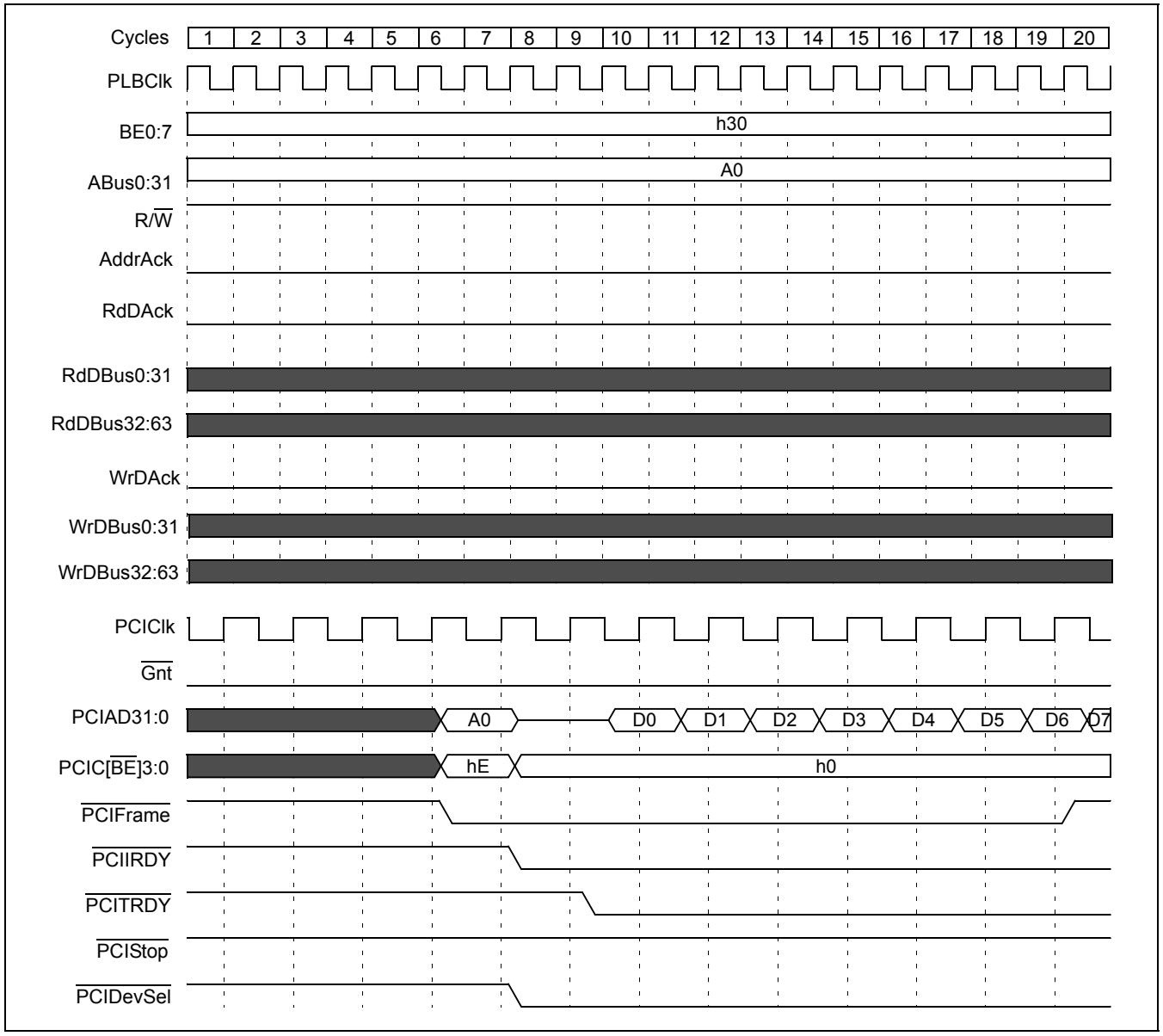


Figure 19-79. PCI Memory To SDRAM DMA Transfer (Continued)

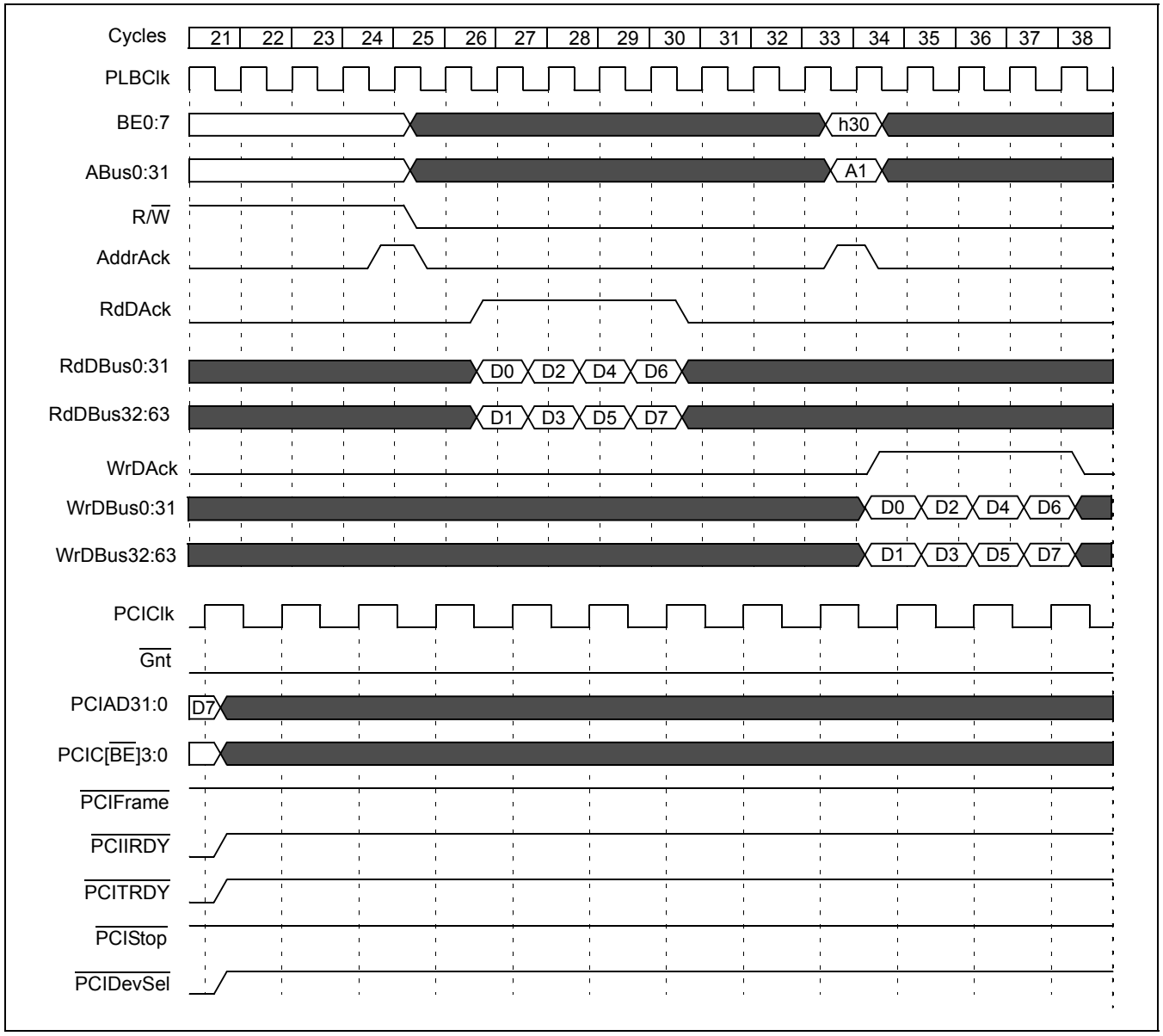


Figure 19-80. SDRAM To PCI Memory DMA Transfer

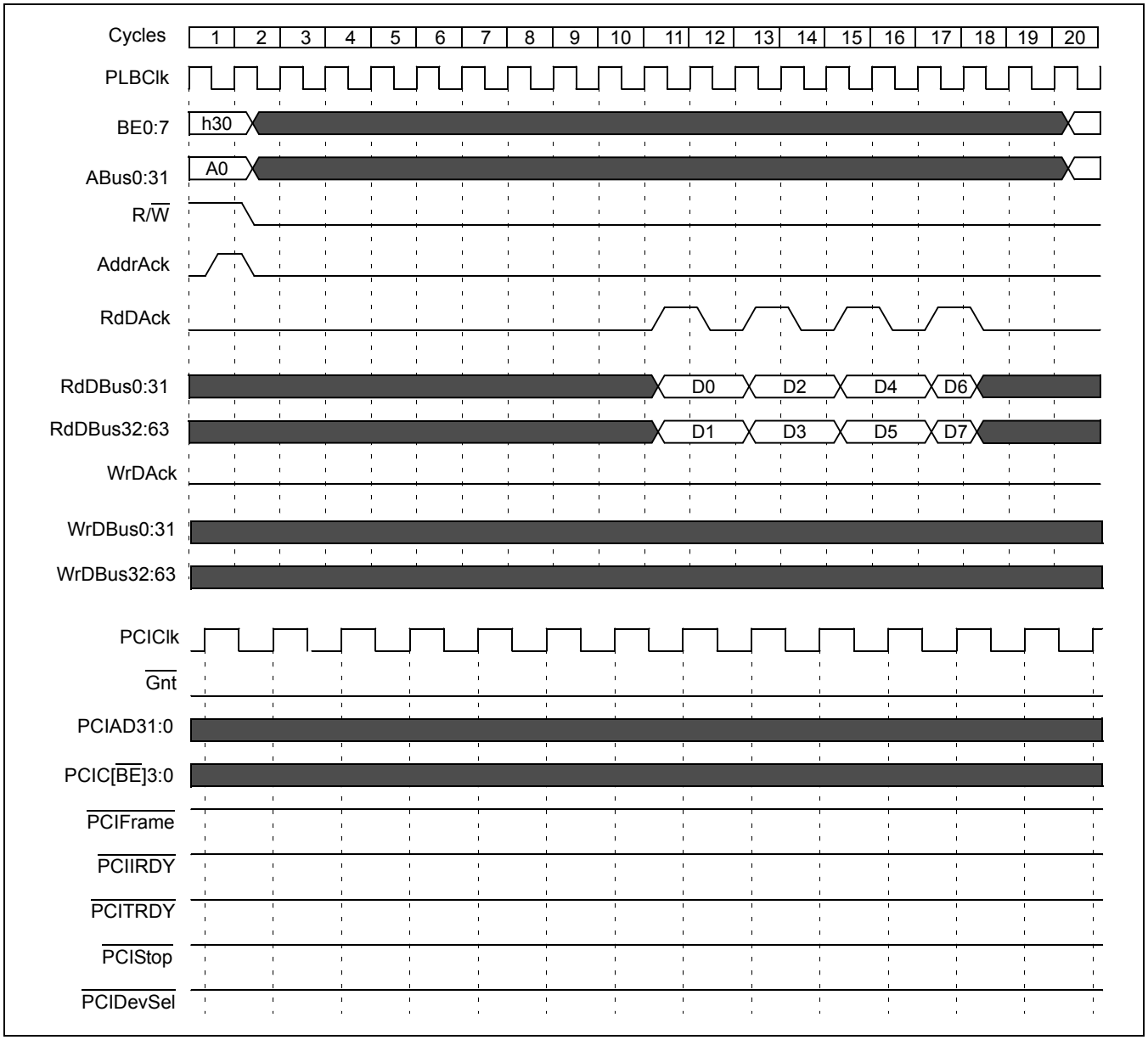


Figure 19-81. SDRAM To PCI Memory DMA Transfer (Continued)

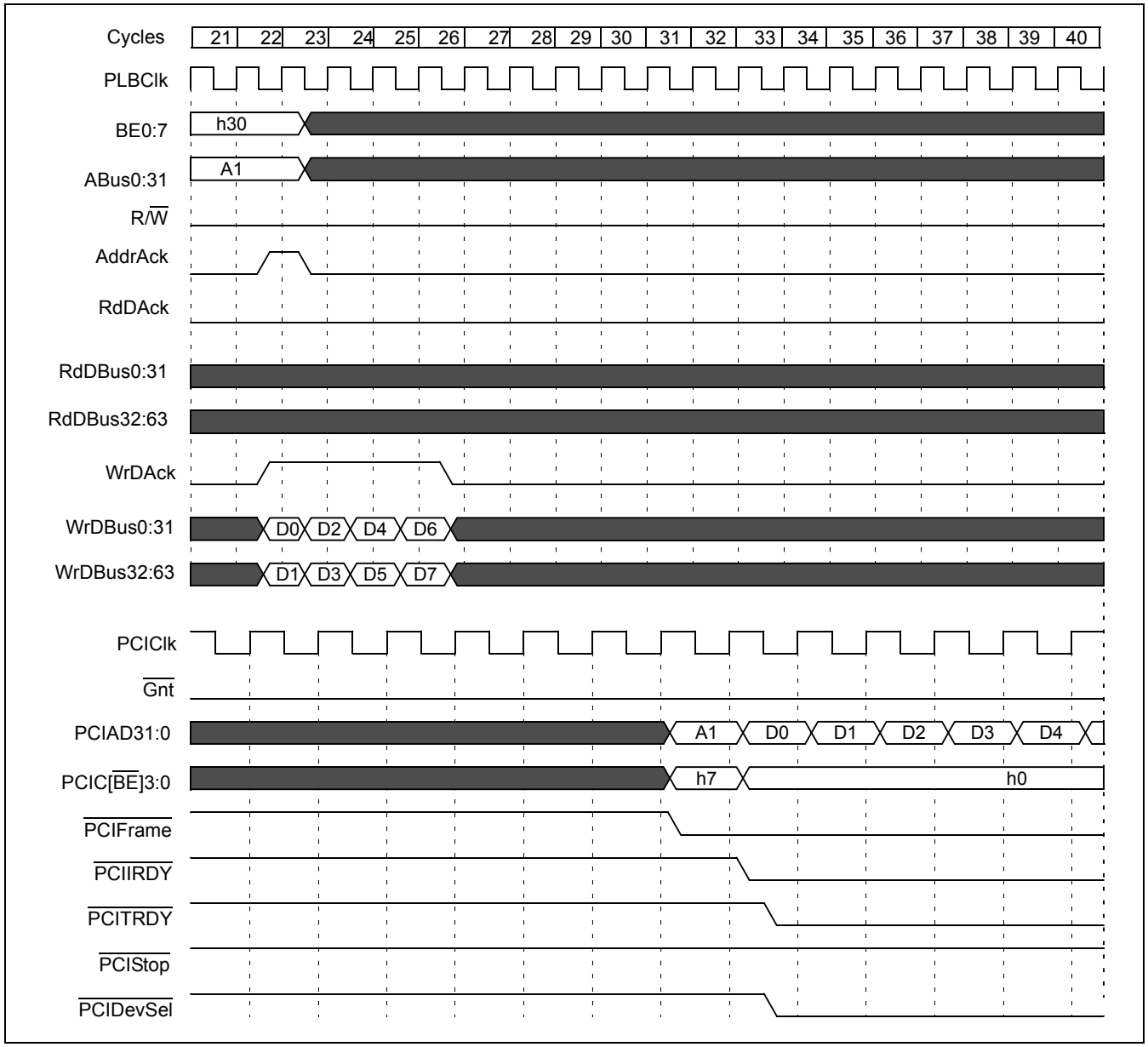
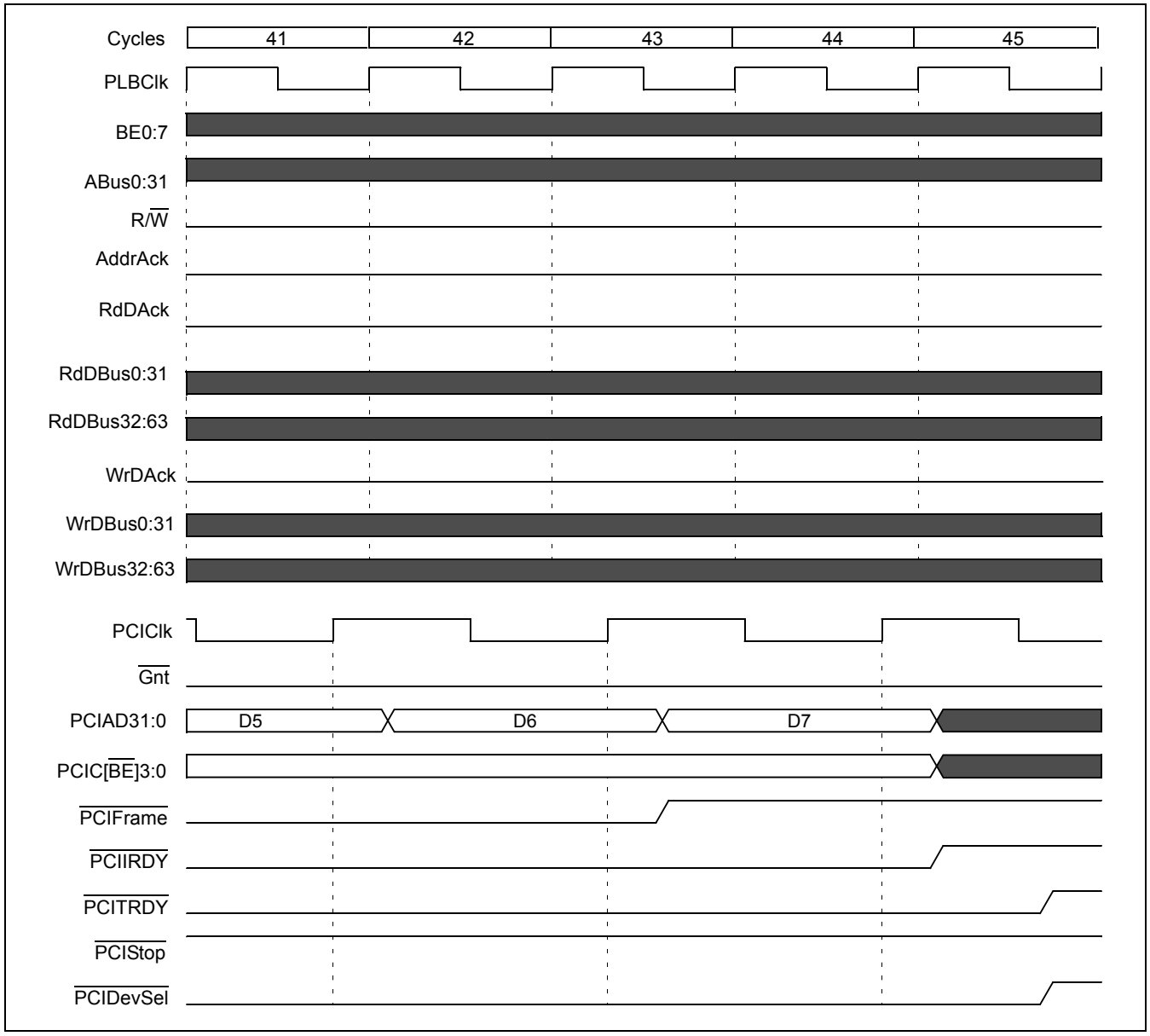


Figure 19-82. SDRAM To PCI Memory DMA Transfer (Continued)



19.10.3 Synchronous

The following diagrams are for synchronous clocking mode. Note that all of the diagrams flow across multiple pages. Each diagram begins with cycle 1 on the left facing page.

Figure 19-83. PCI Master Burst Read From SDRAM

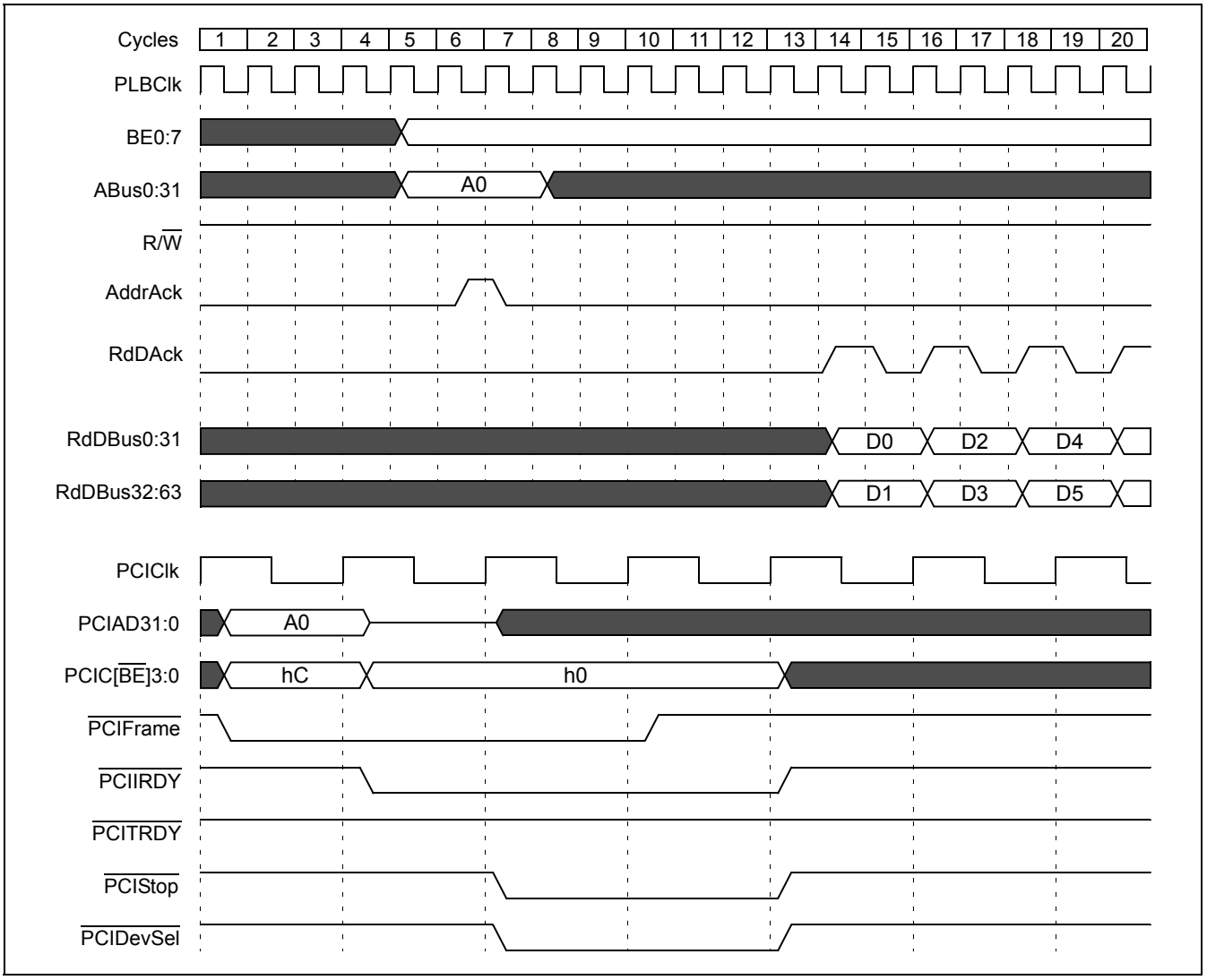


Figure 19-84. PCI Master Burst Read From SDRAM (Continued)

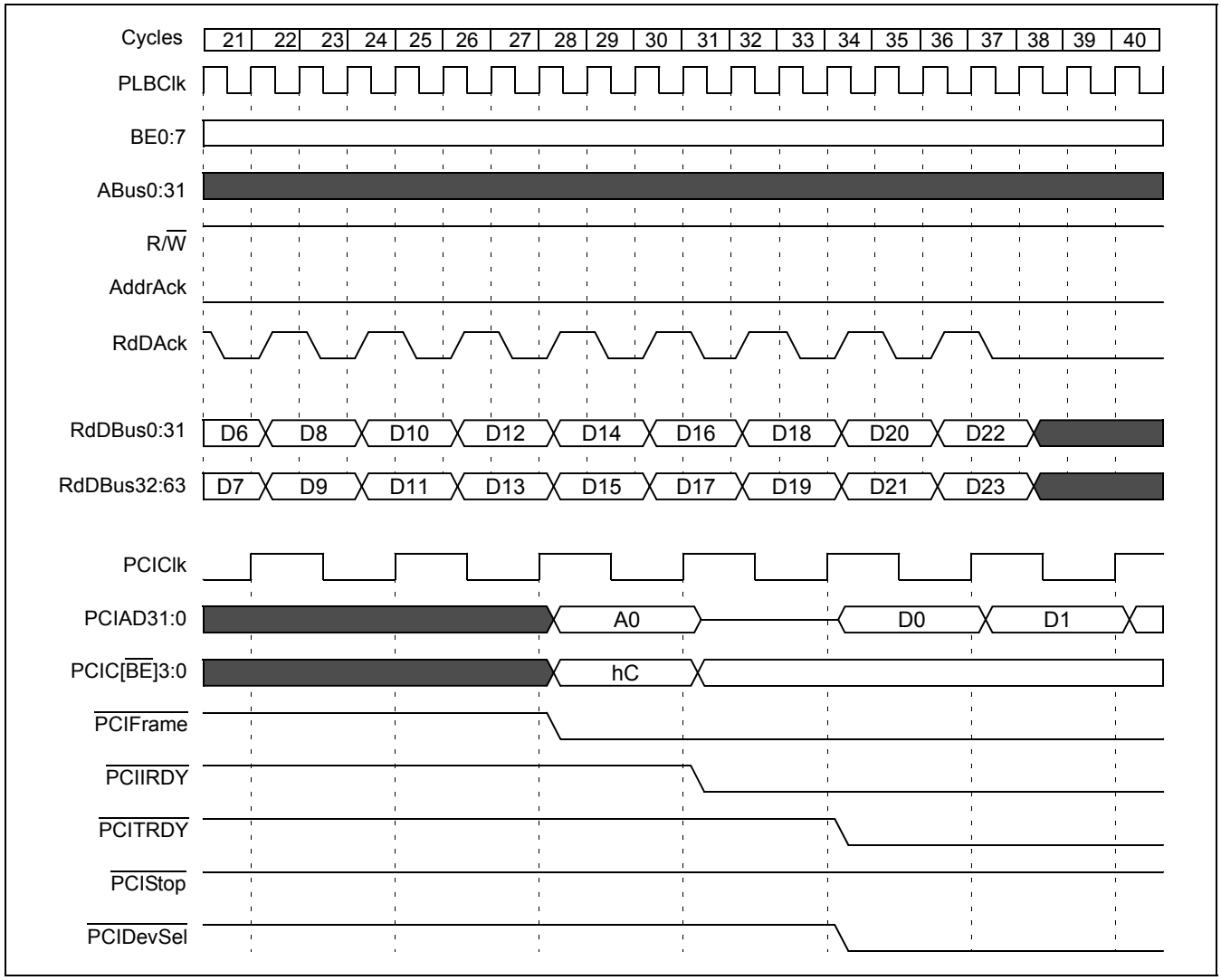


Figure 19-85. PCI Master Burst Read From SDRAM (Continued)

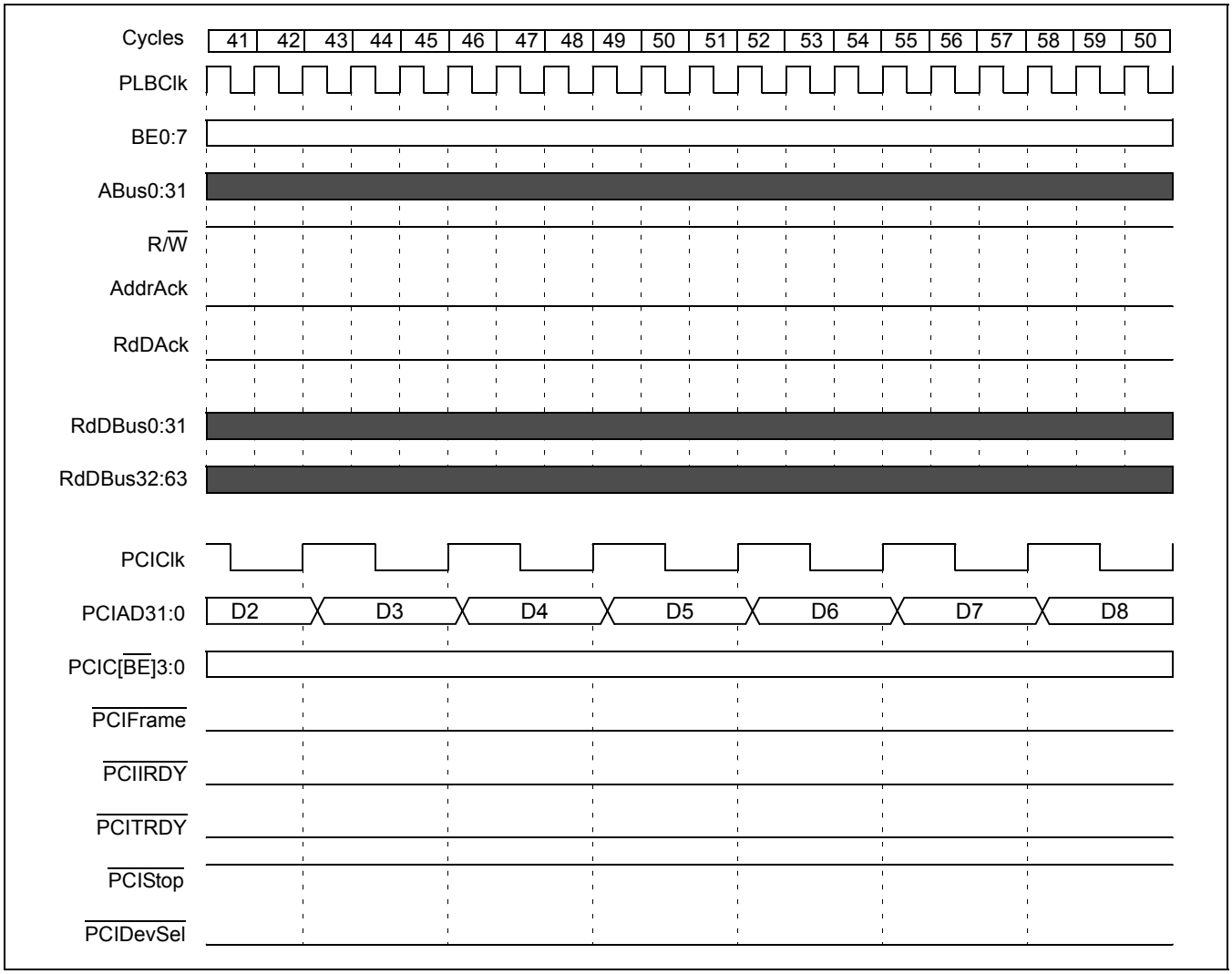


Figure 19-86. PCI Master Burst Read From SDRAM (Continued)

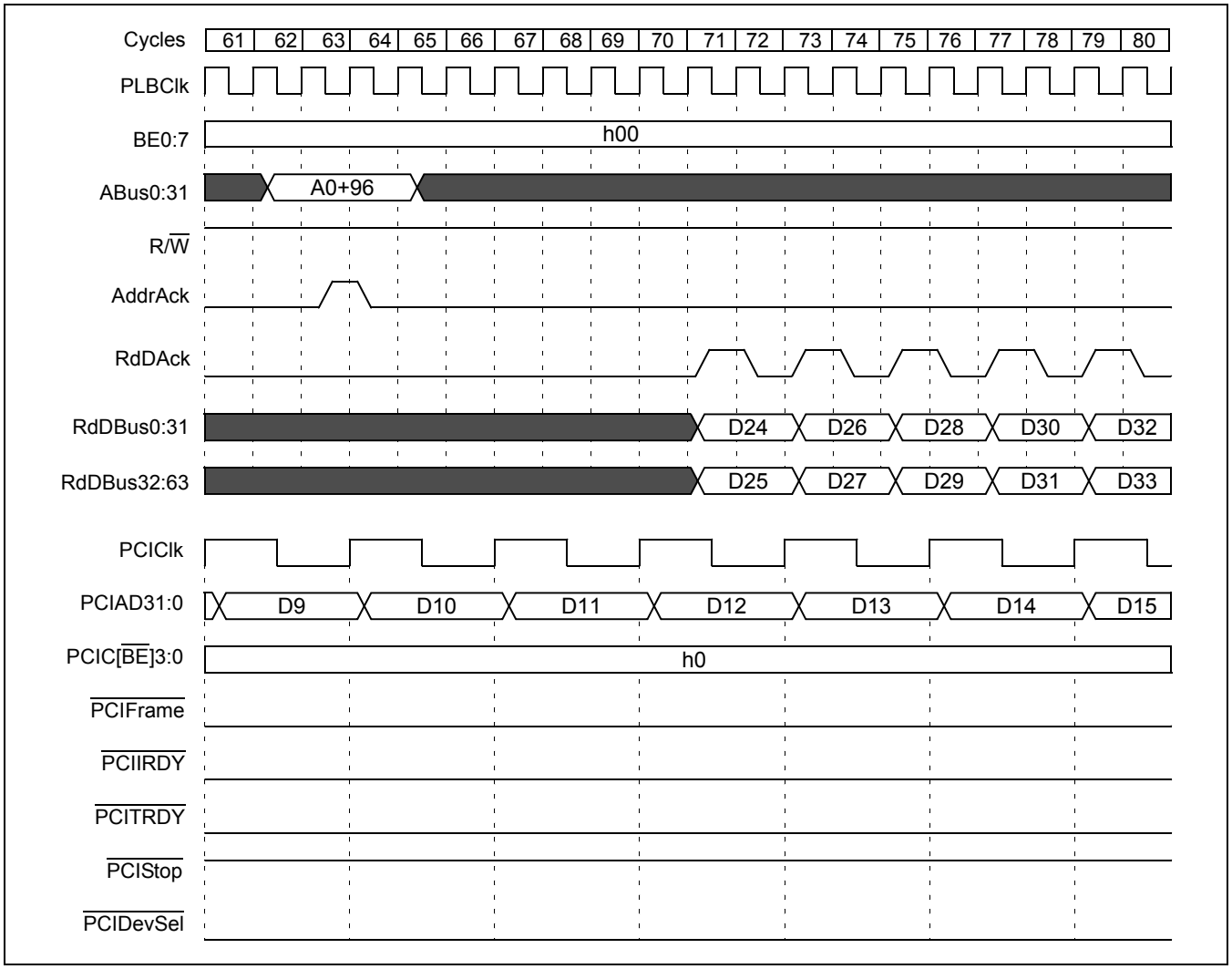


Figure 19-87. PCI Master Burst Read From SDRAM (Continued)

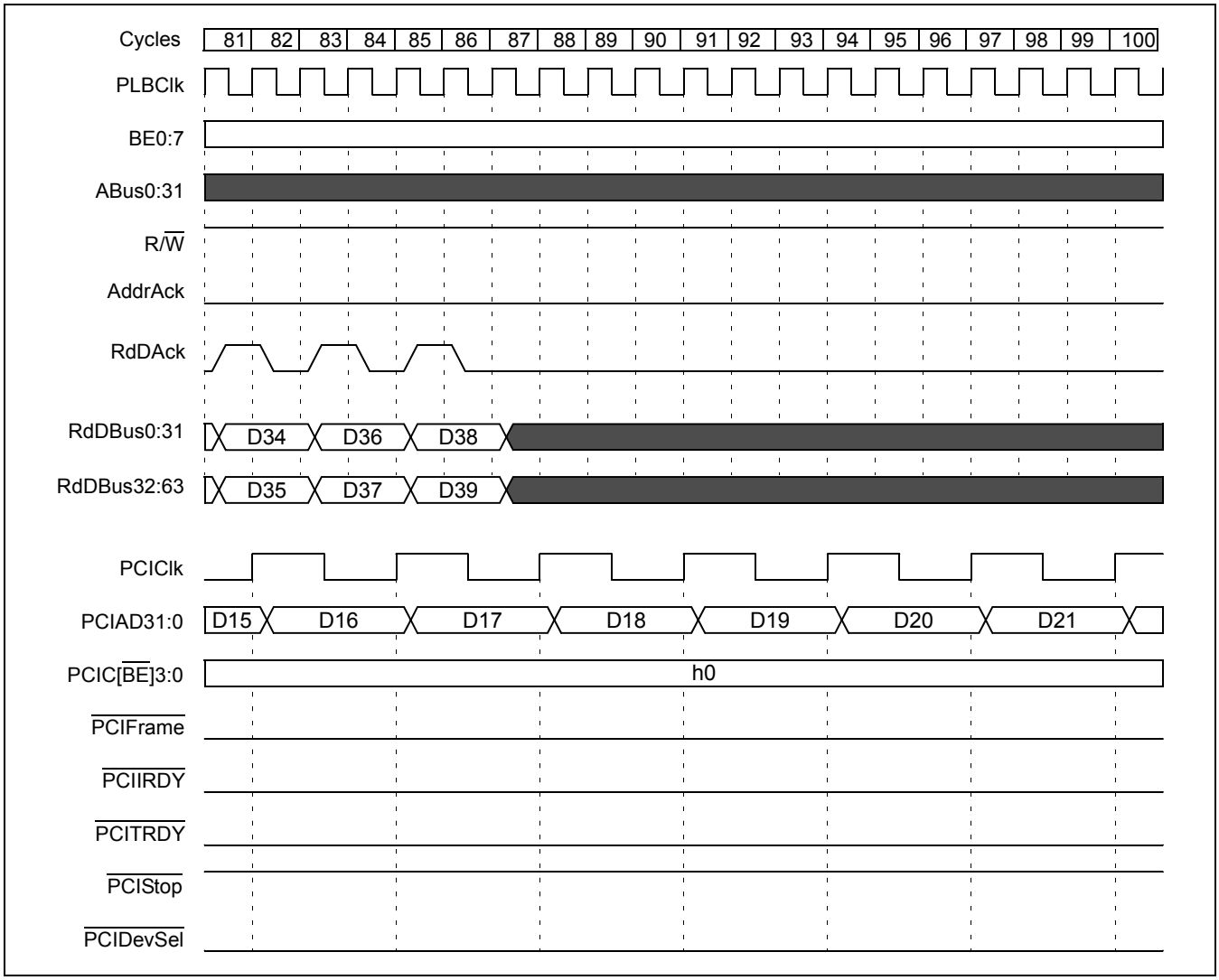


Figure 19-88. PCI Master Burst Read From SDRAM (Continued)

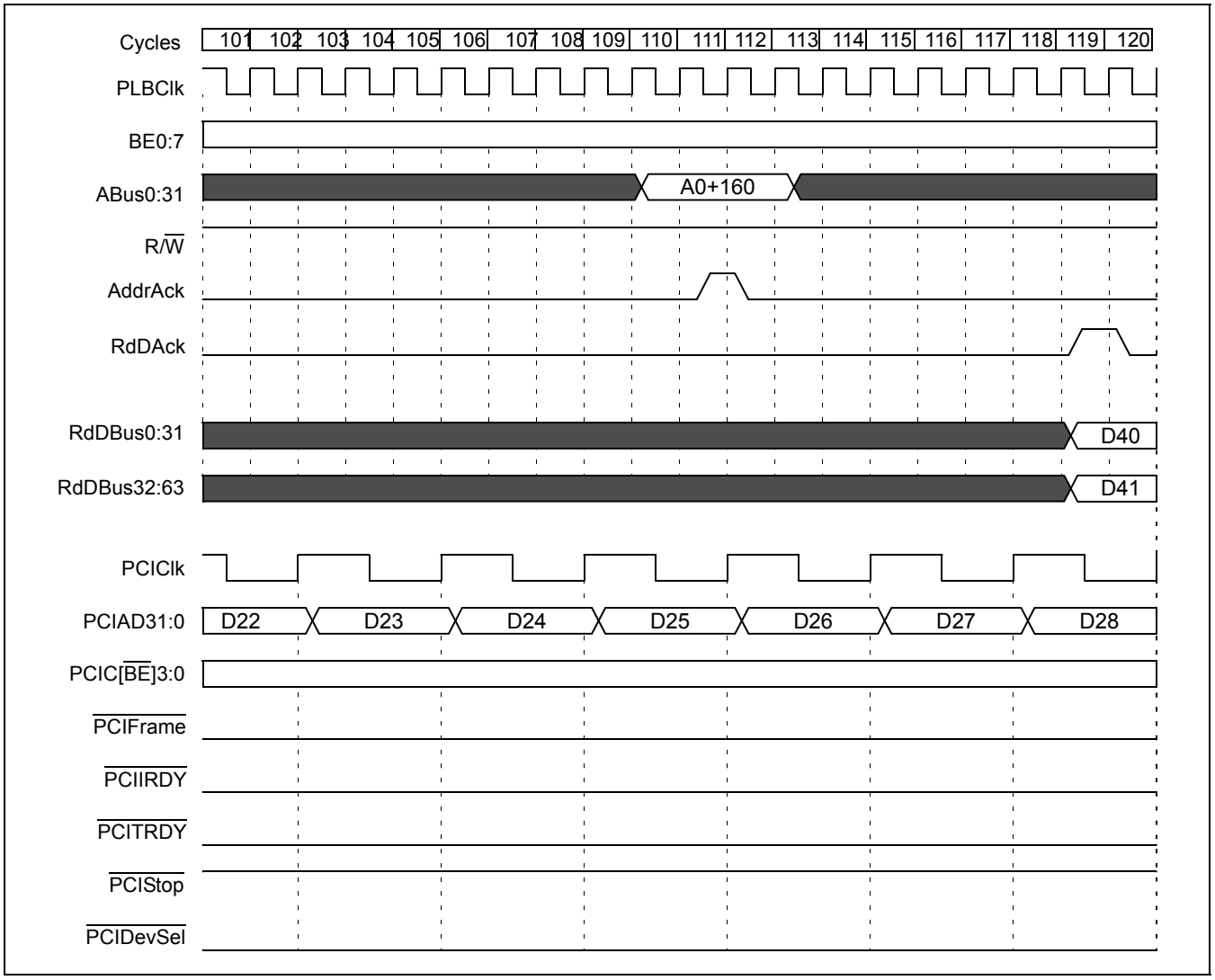


Figure 19-89. PCI Master Burst Read From SDRAM (Continued)

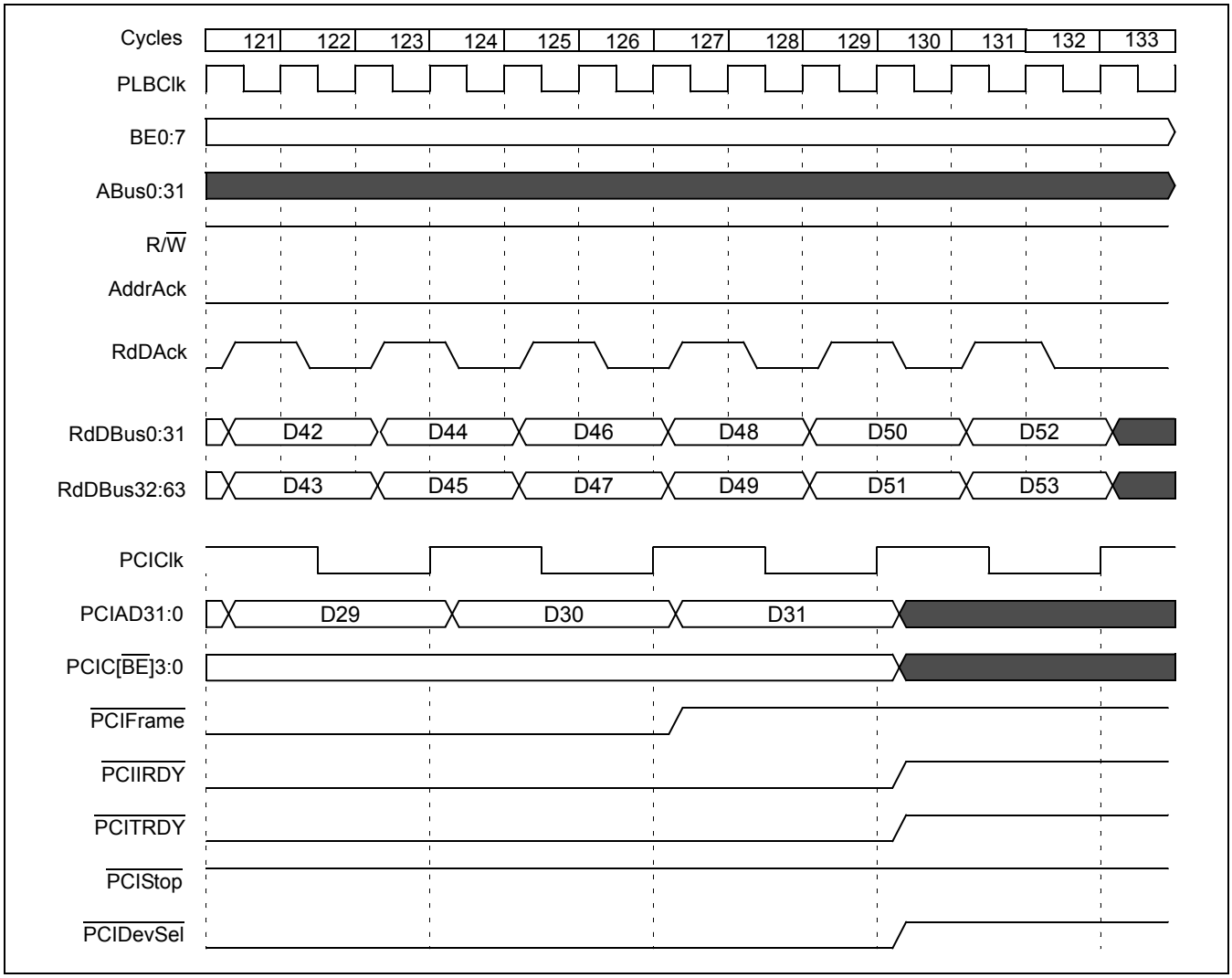


Figure 19-90. PCI Master Burst Write To SDRAM

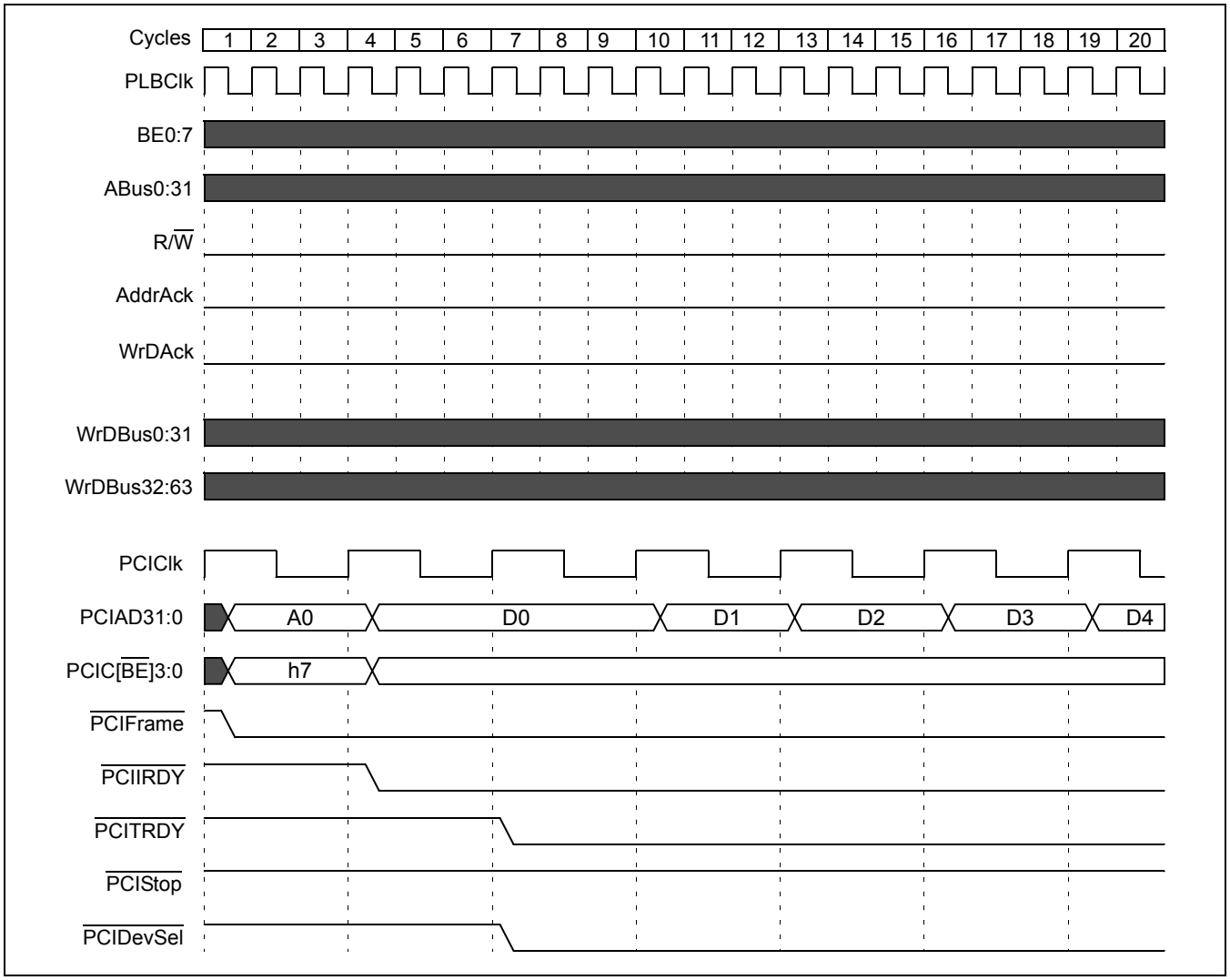


Figure 19-91. PCI Master Burst Write To SDRAM (Continued)

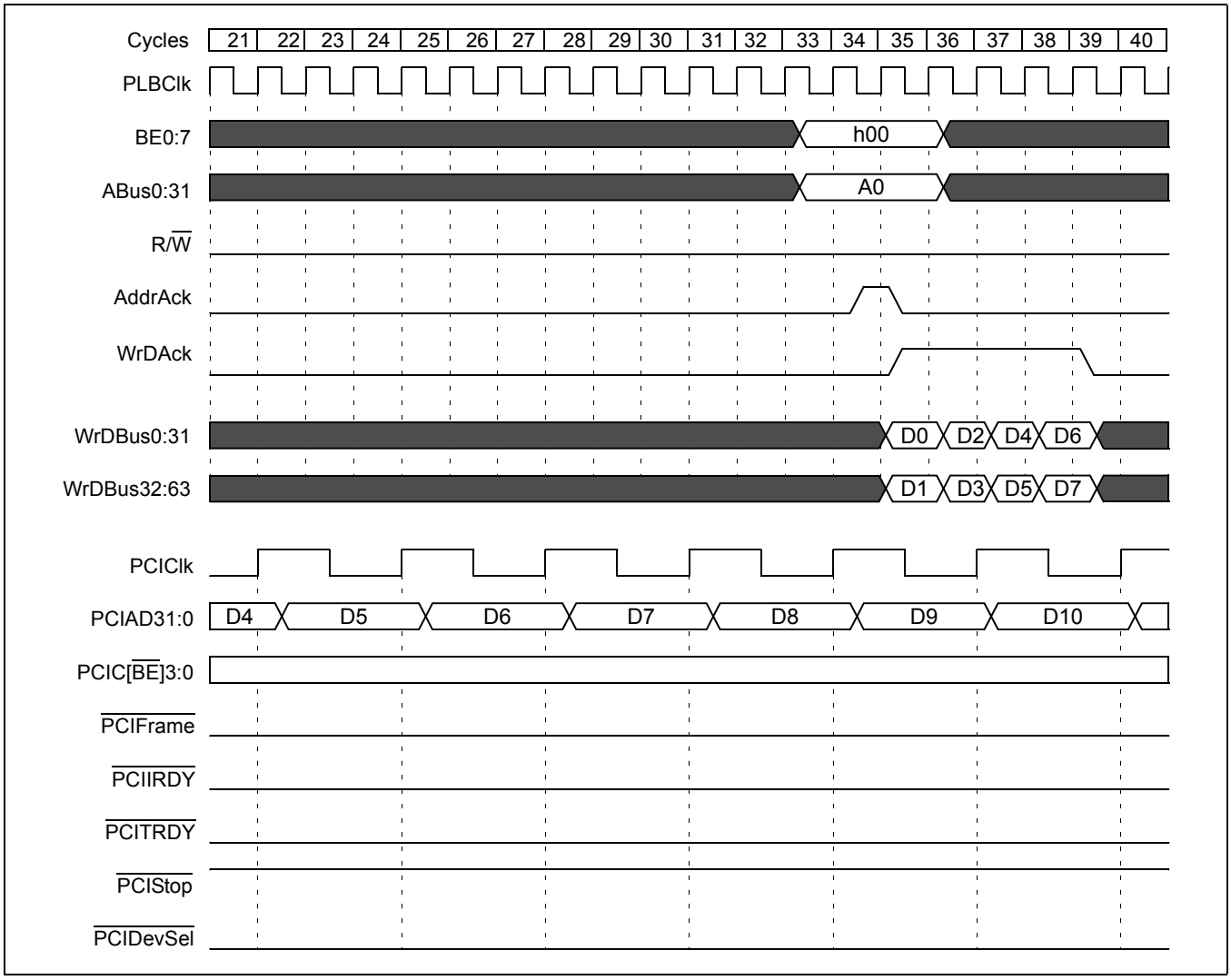


Figure 19-92. PCI Master Burst Write To SDRAM (Continued)

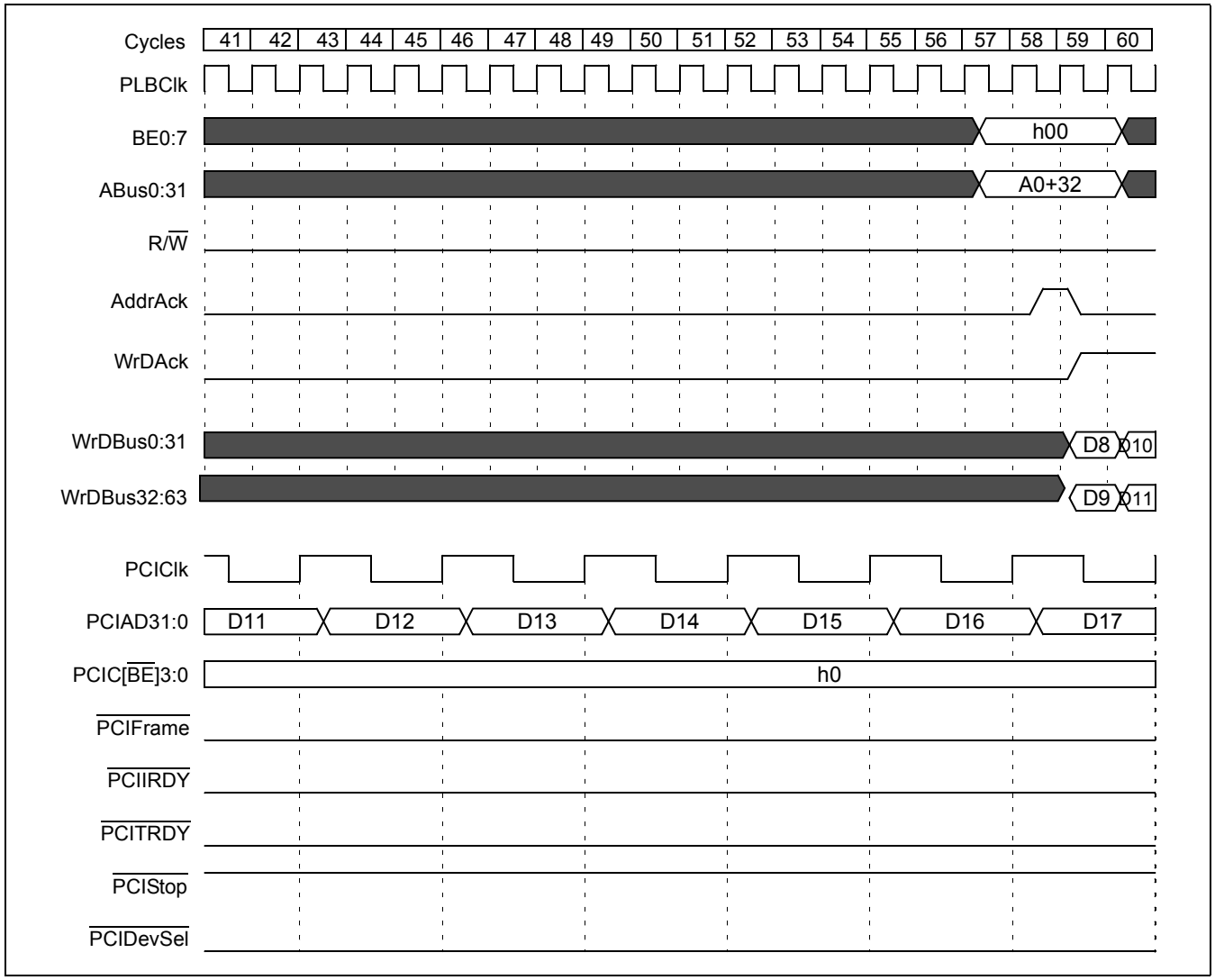


Figure 19-93. PCI Master Burst Write To SDRAM (Continued)

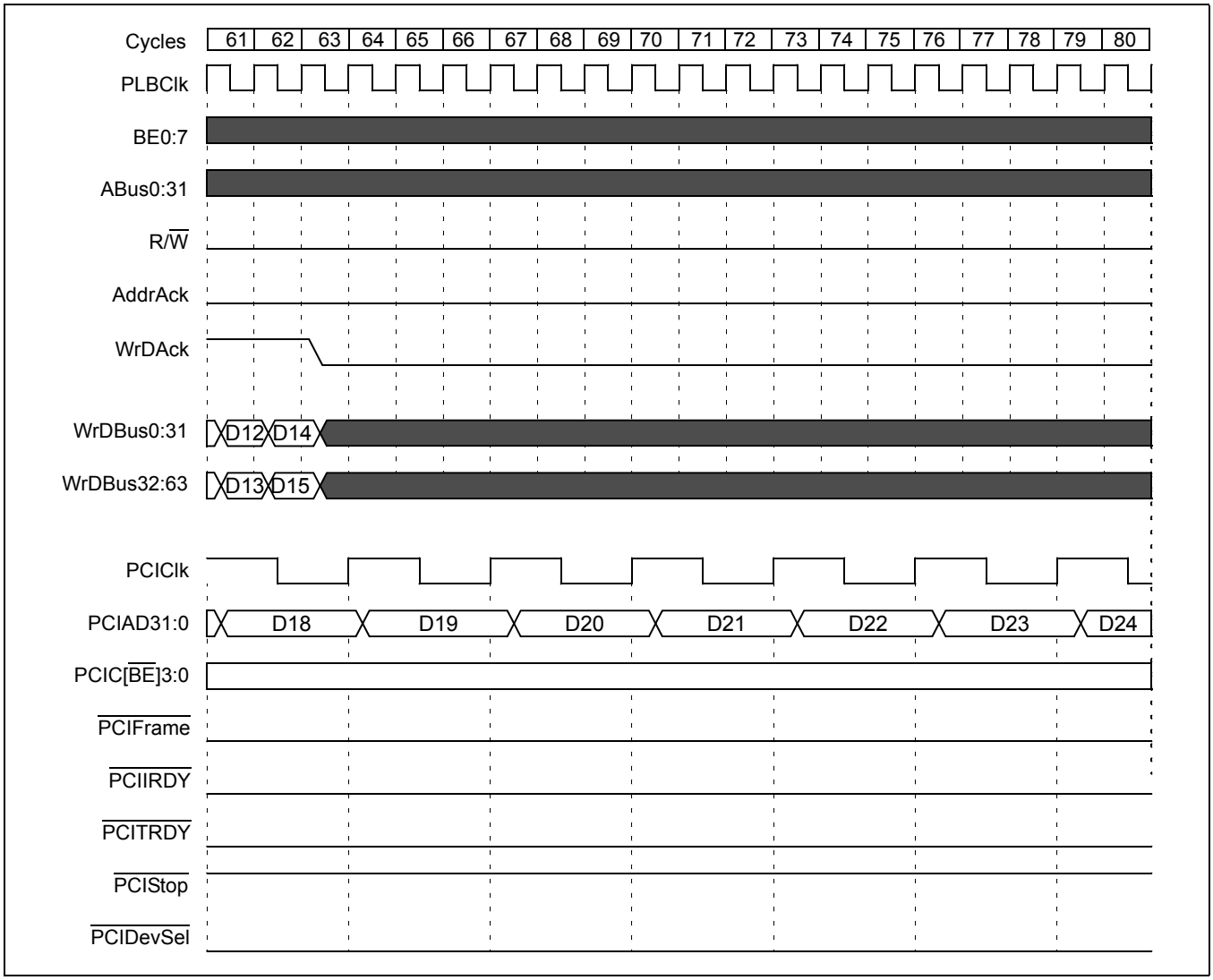


Figure 19-94. PCI Master Burst Write To SDRAM (Continued)

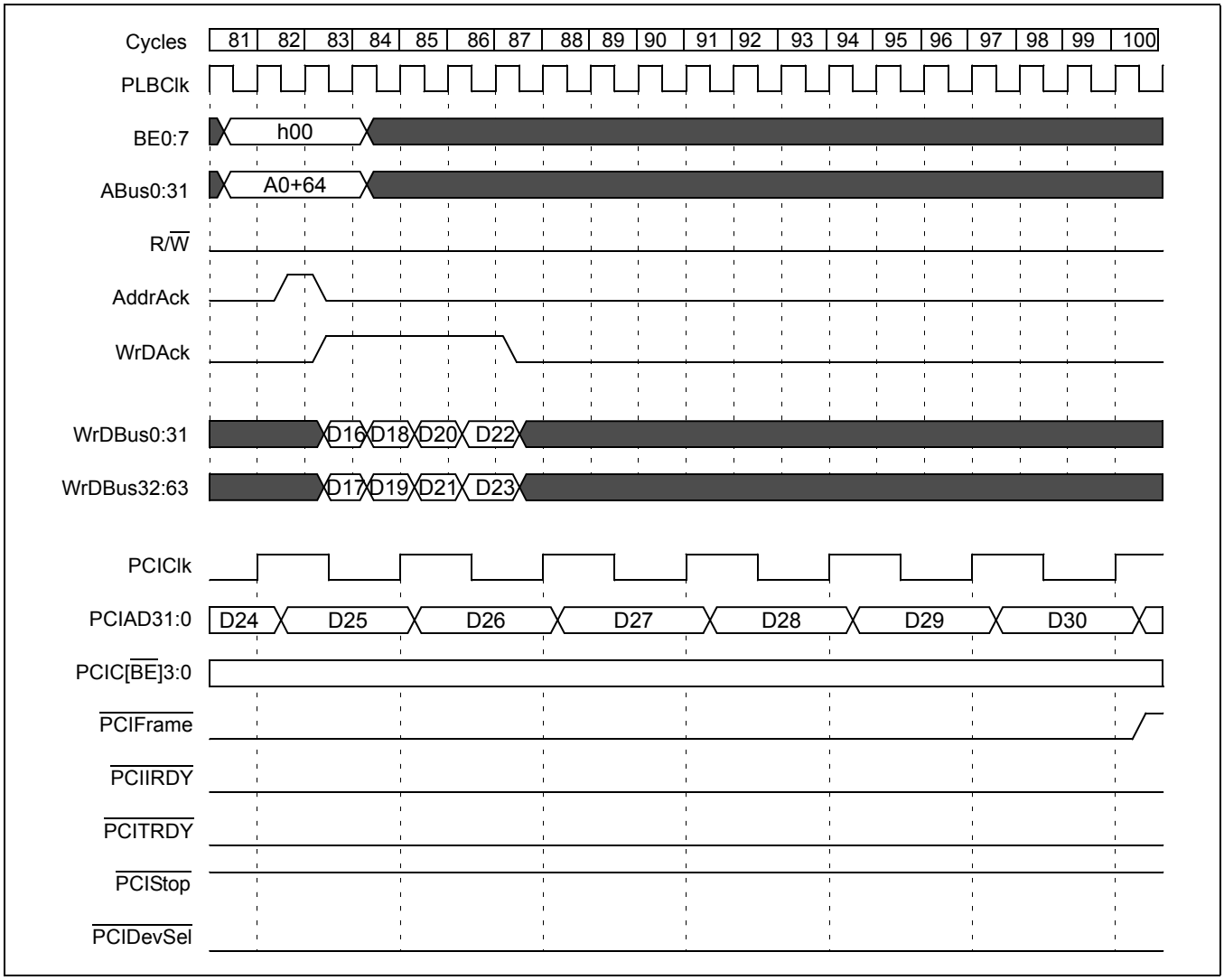


Figure 19-95. PCI Master Burst Write To SDRAM (Continued)

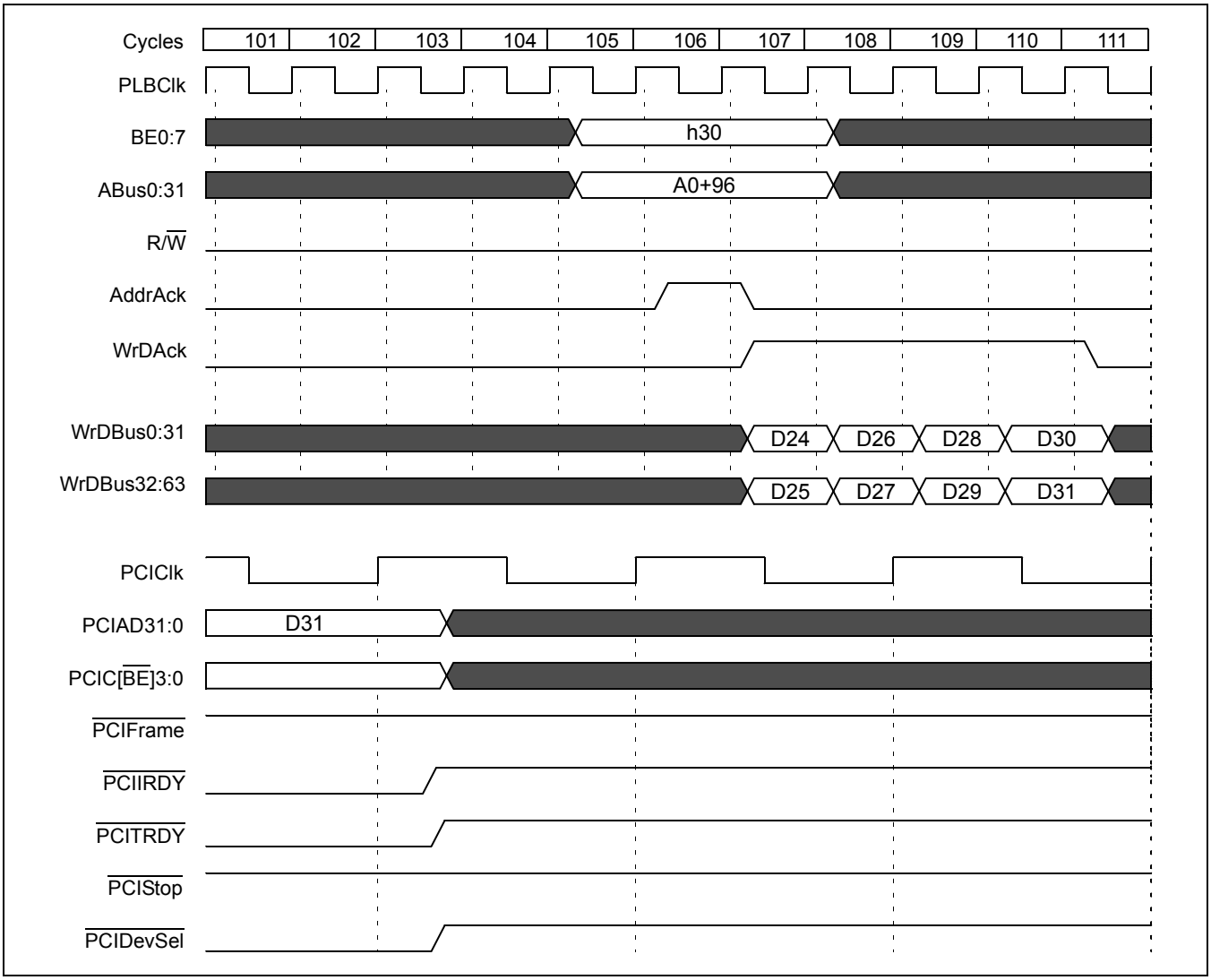


Figure 19-96. CPU Read From PCI Memory Slave, Nonprefetching

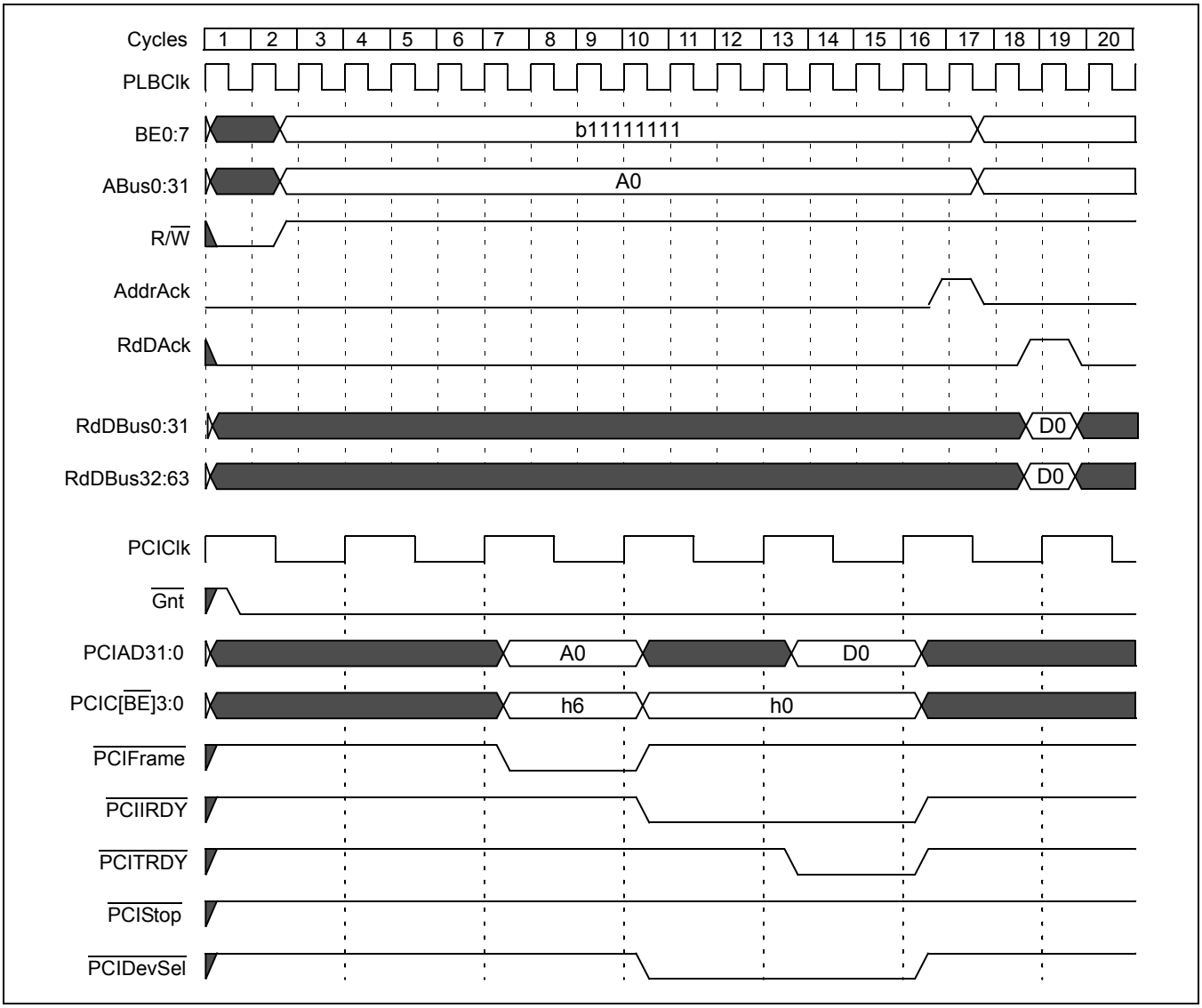


Figure 19-97. CPU Read From PCI Memory Slave, Nonprefetching (Continued)

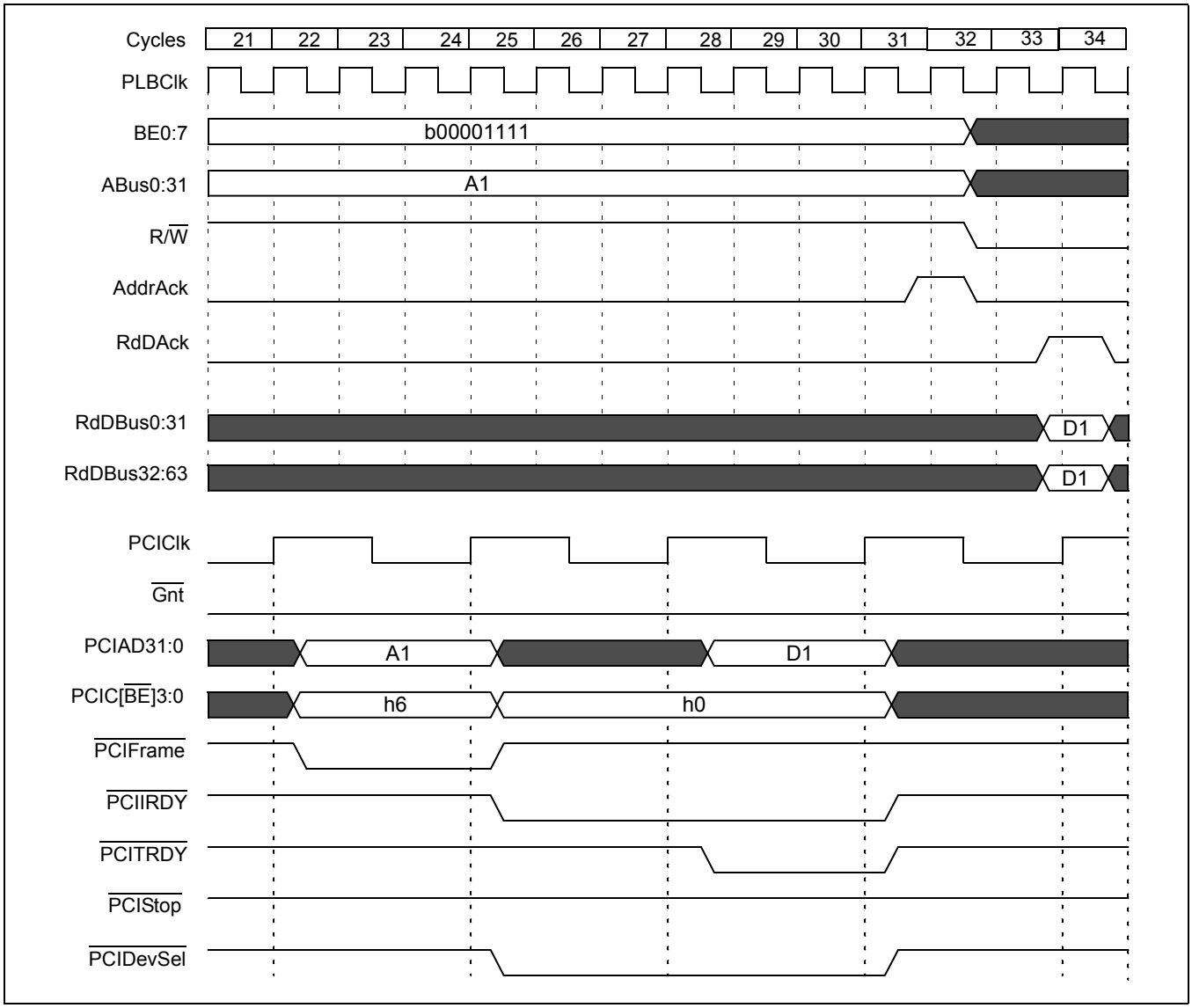


Figure 19-98. CPU Read From PCI Memory Slave, Prefetching

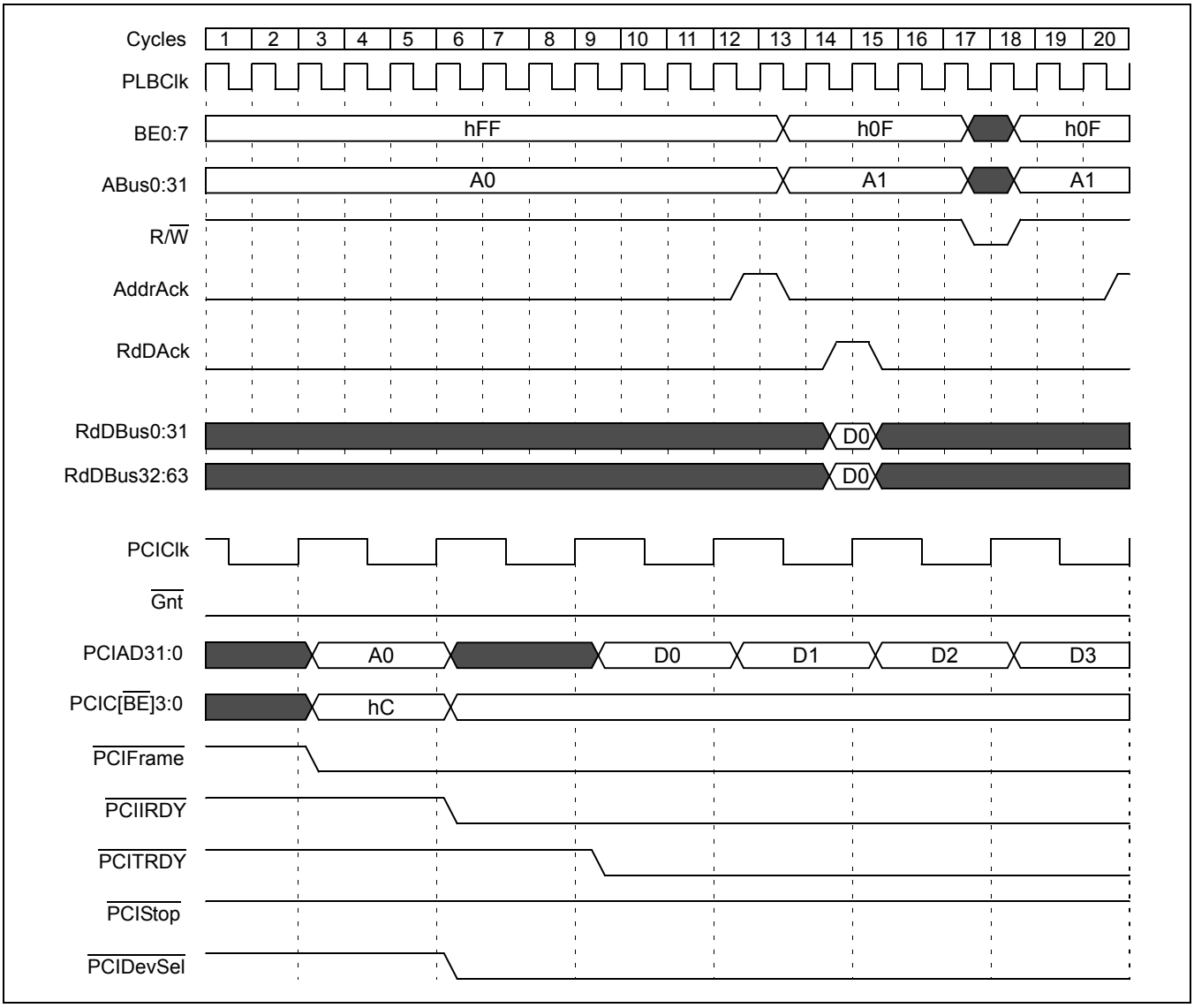


Figure 19-99. CPU Read From PCI Memory Slave, Prefetching (Continued)

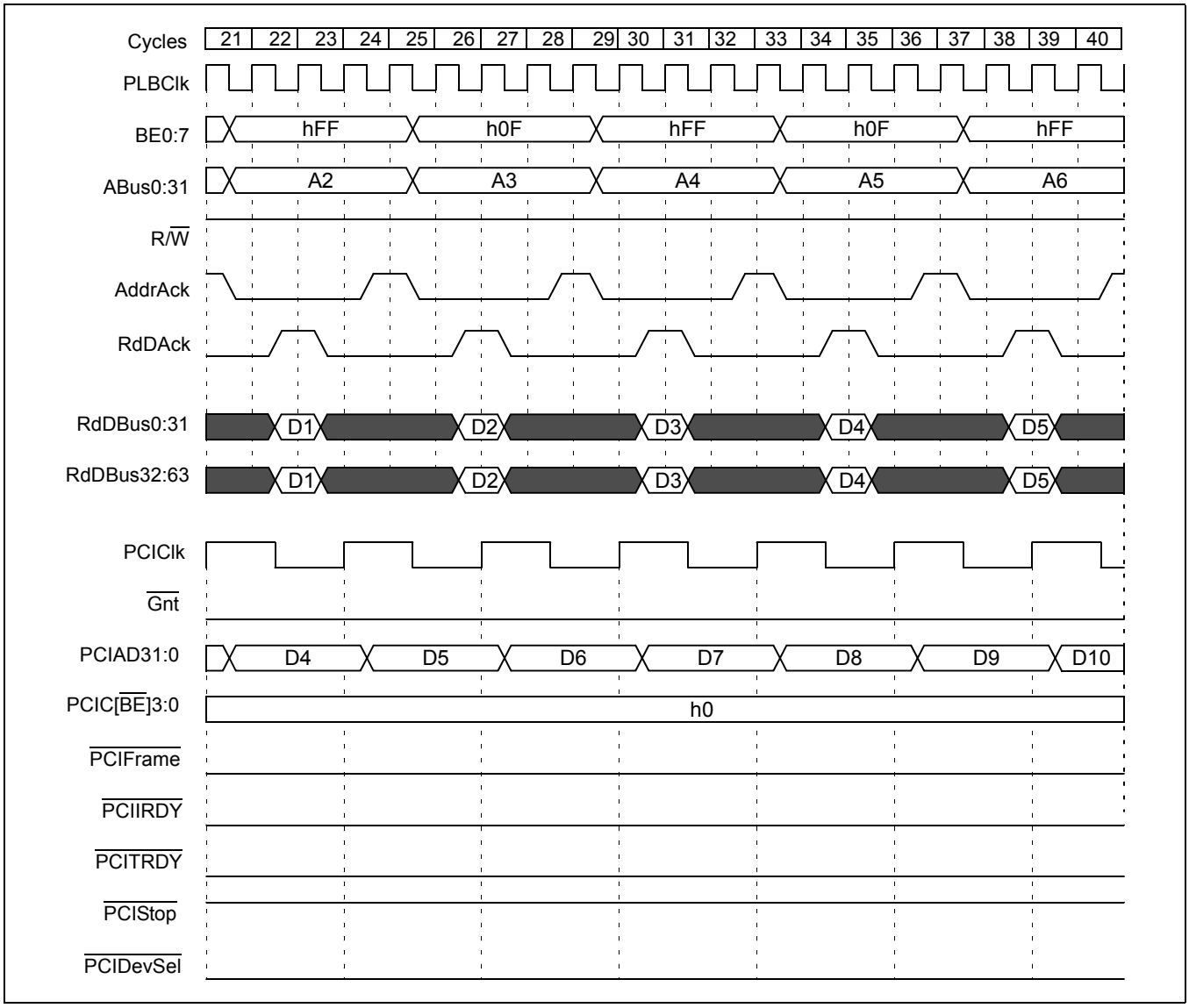


Figure 19-100. CPU Read From PCI Memory Slave, Prefetching (Continued)

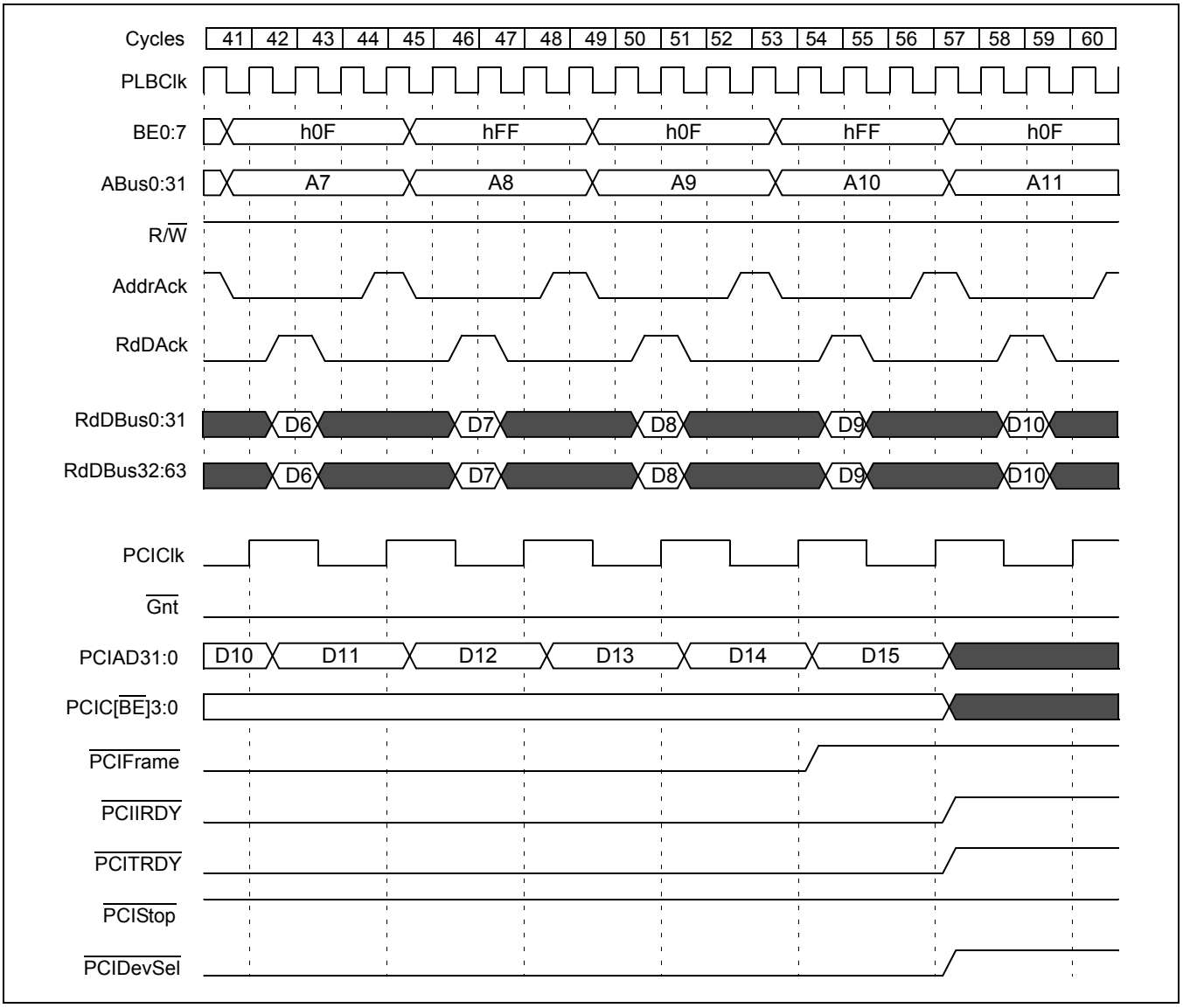


Figure 19-101. CPU Read From PCI Memory Slave, Prefetching (Continued)

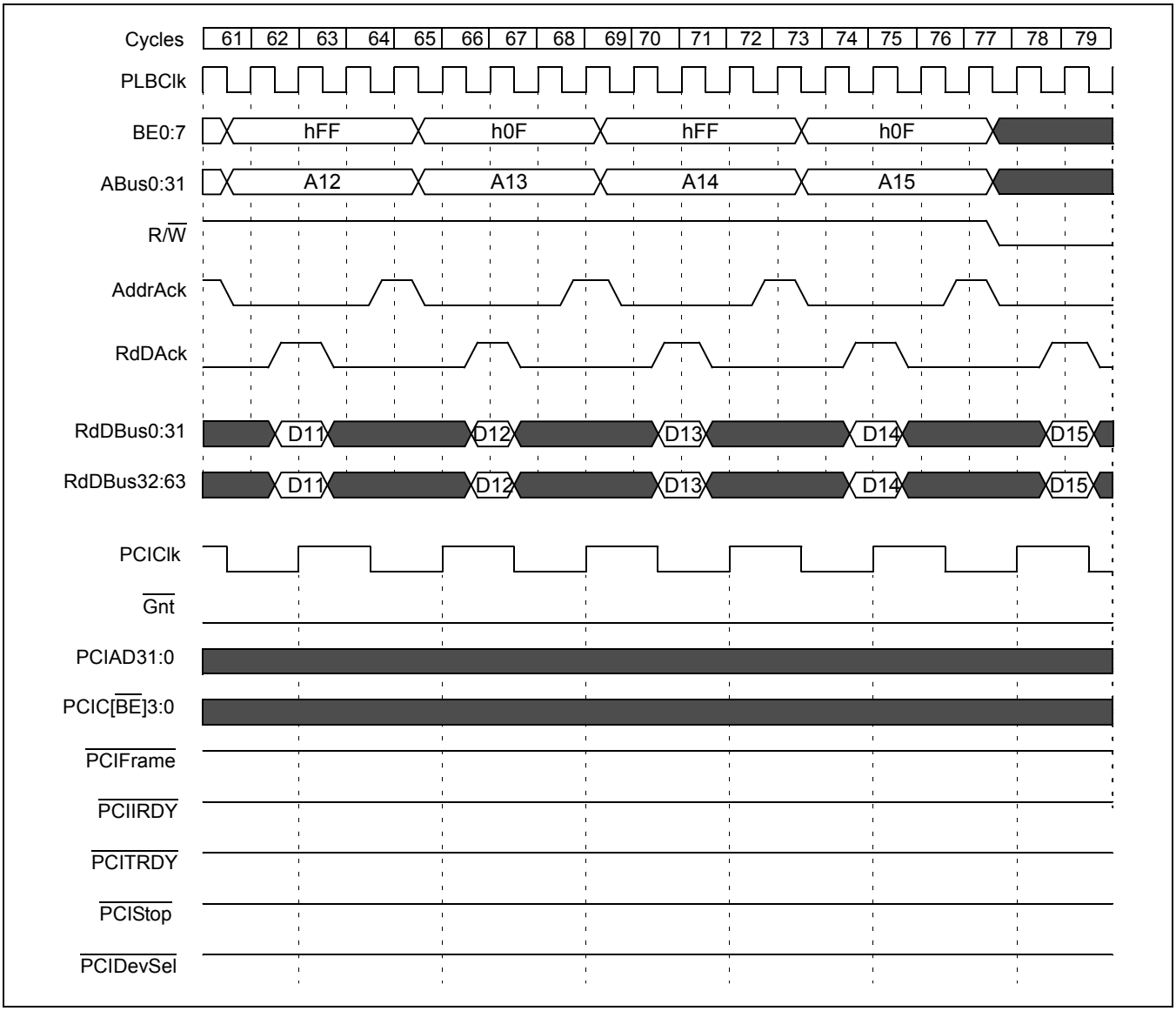


Figure 19-102. CPU Write To PCI Memory Slave

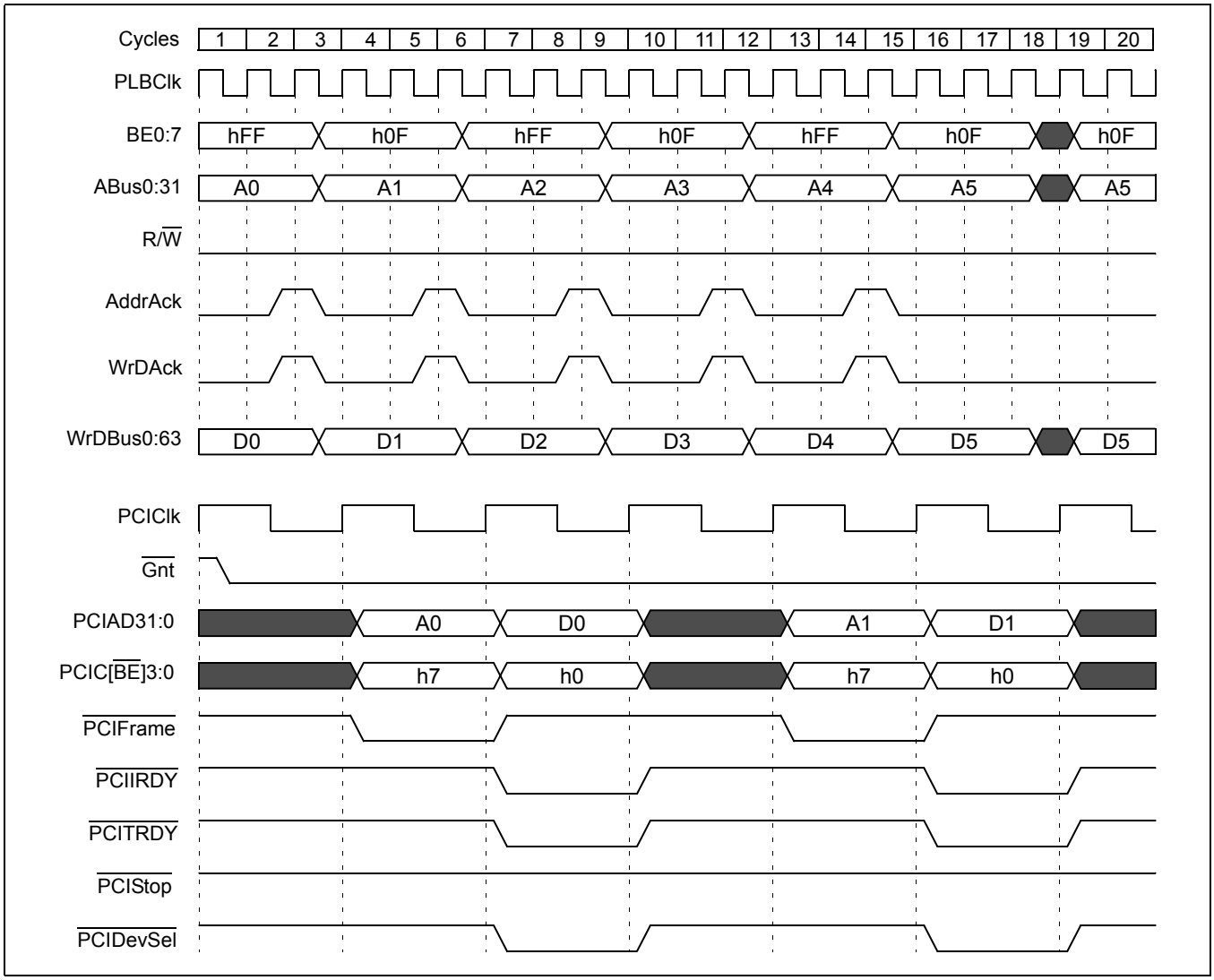


Figure 19-103. CPU Write To PCI Memory Slave (Continued)

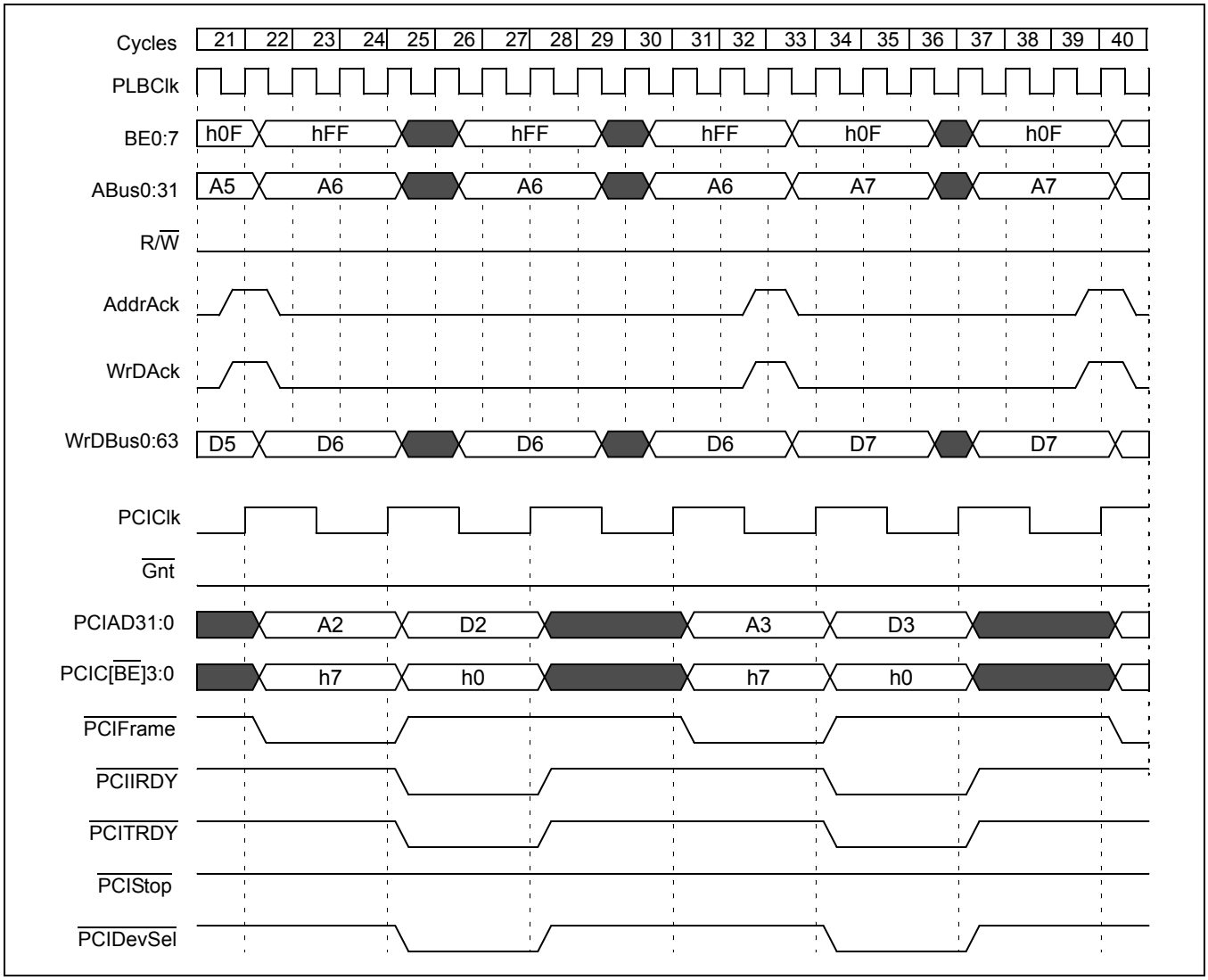


Figure 19-104. CPU Write To PCI Memory Slave (Continued)

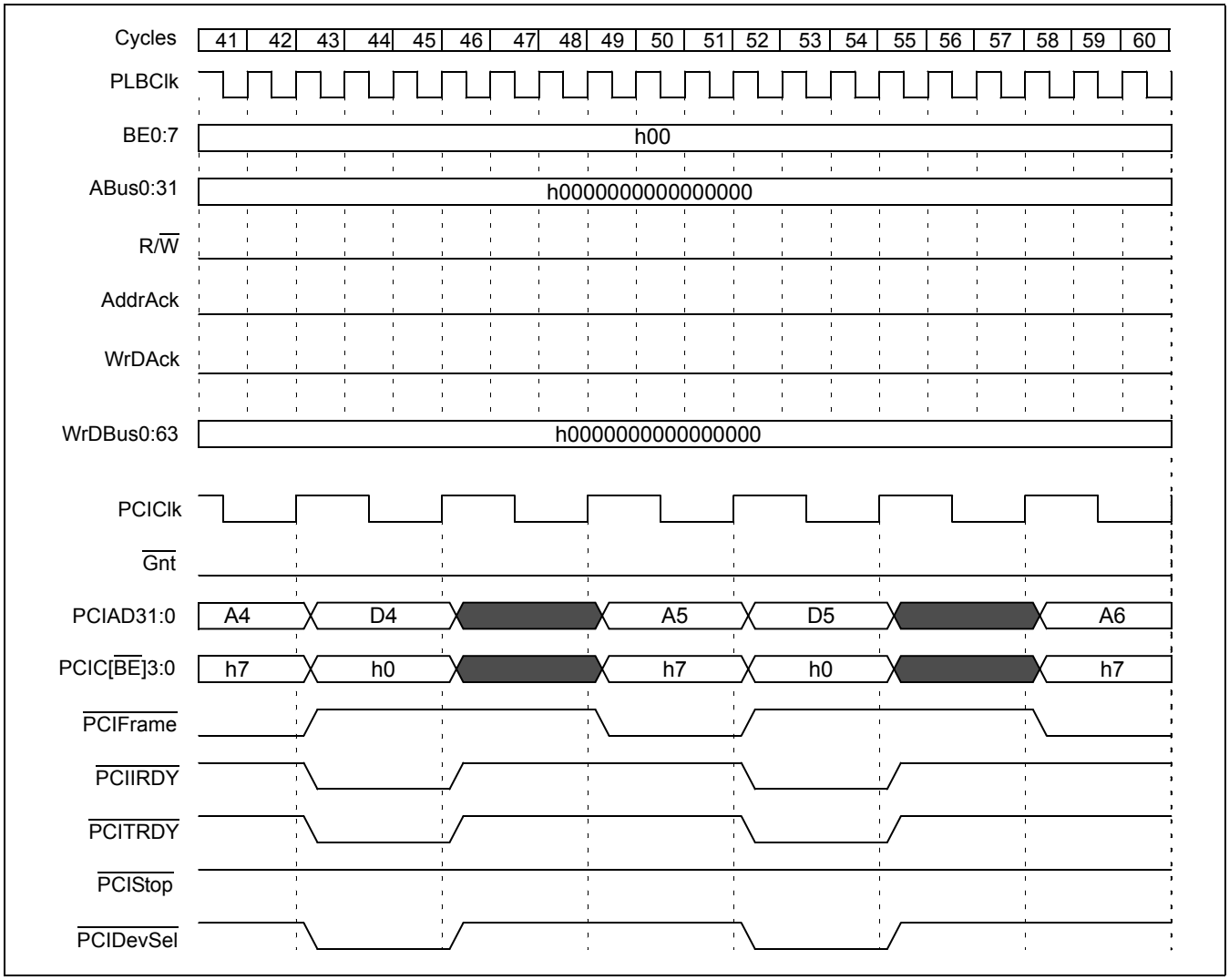


Figure 19-105. CPU Write To PCI Memory Slave (Continued)

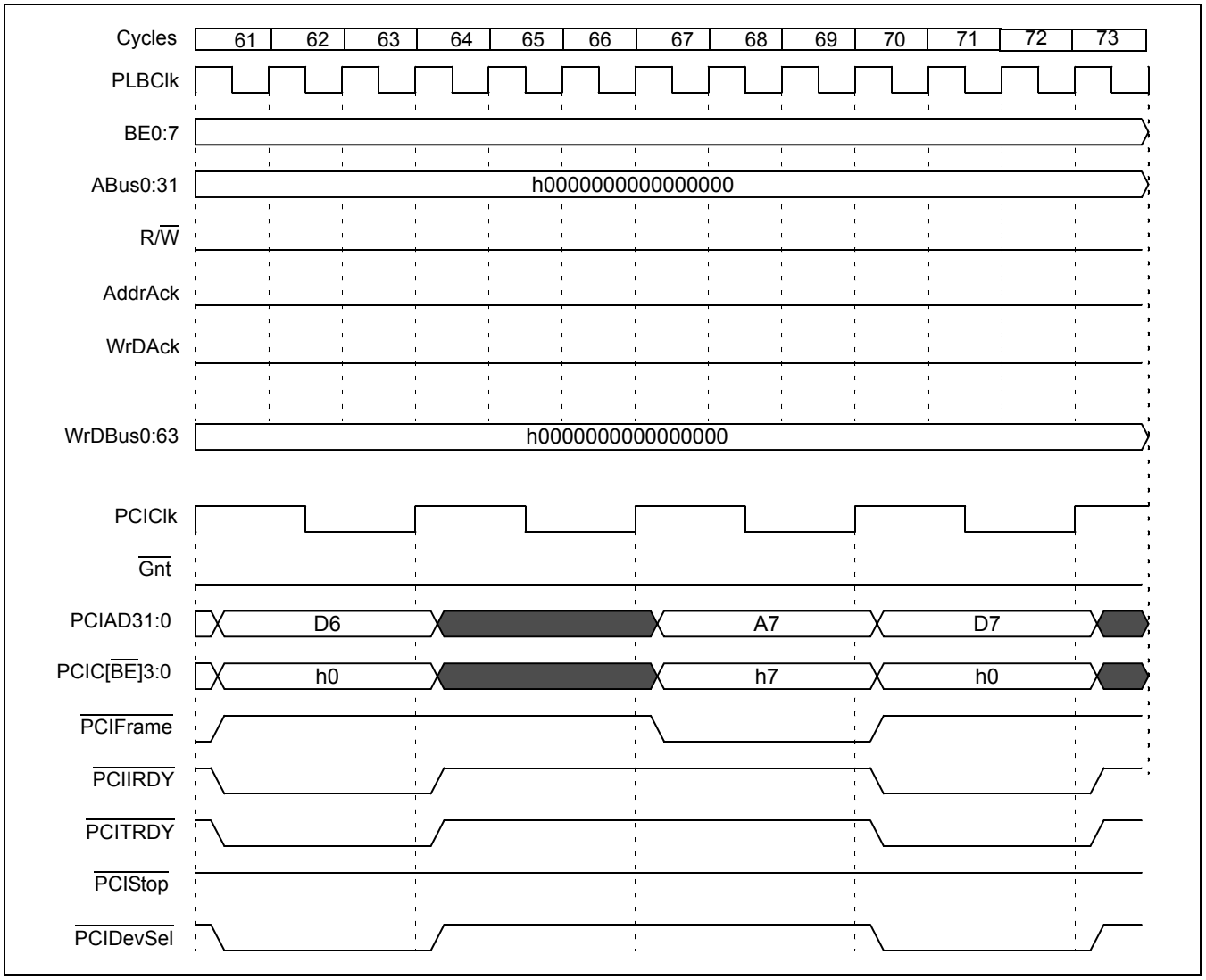


Figure 19-106. PCI Memory To SDRAM DMA Transfer

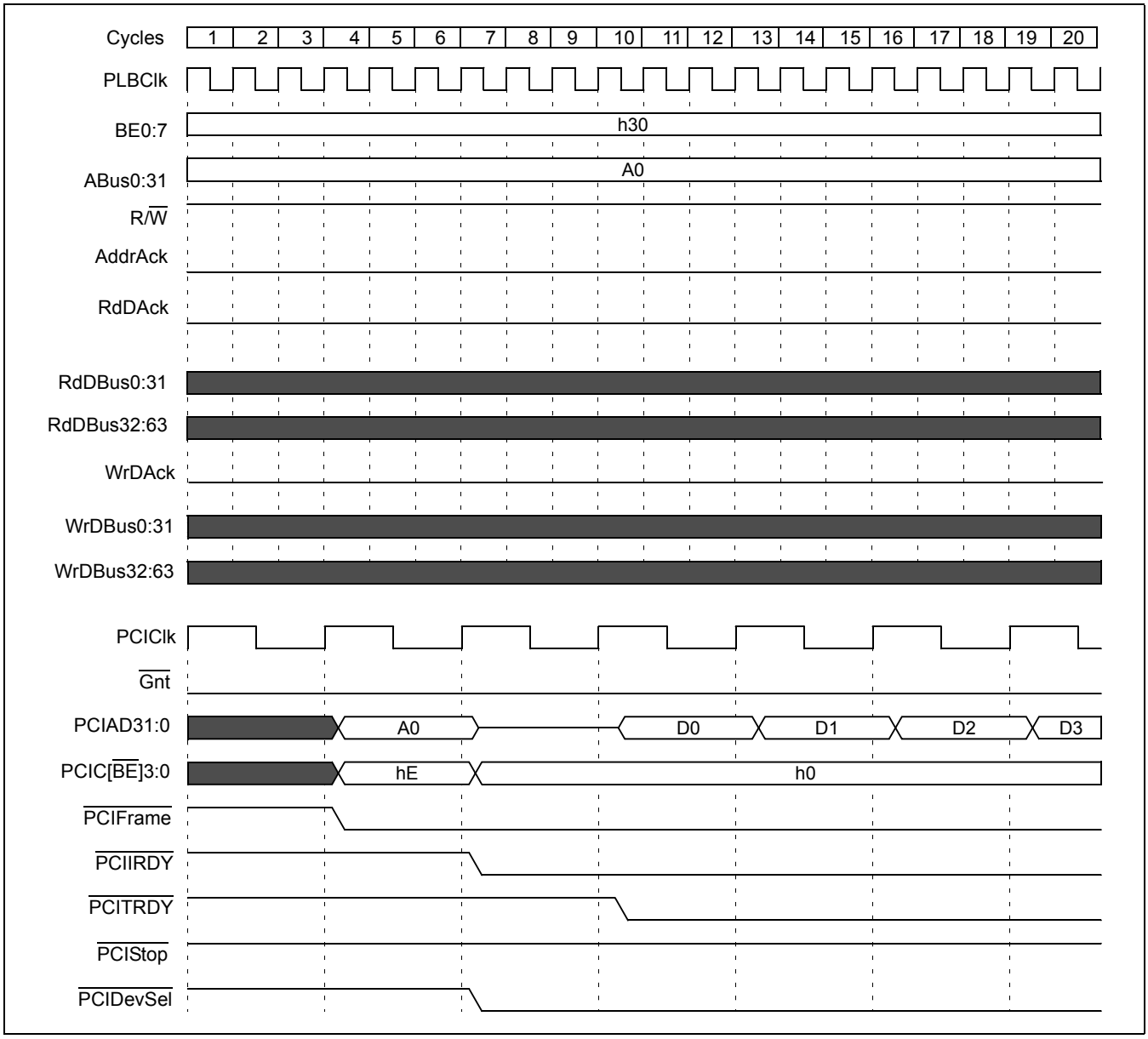


Figure 19-107. PCI Memory To SDRAM DMA Transfer (Continued)

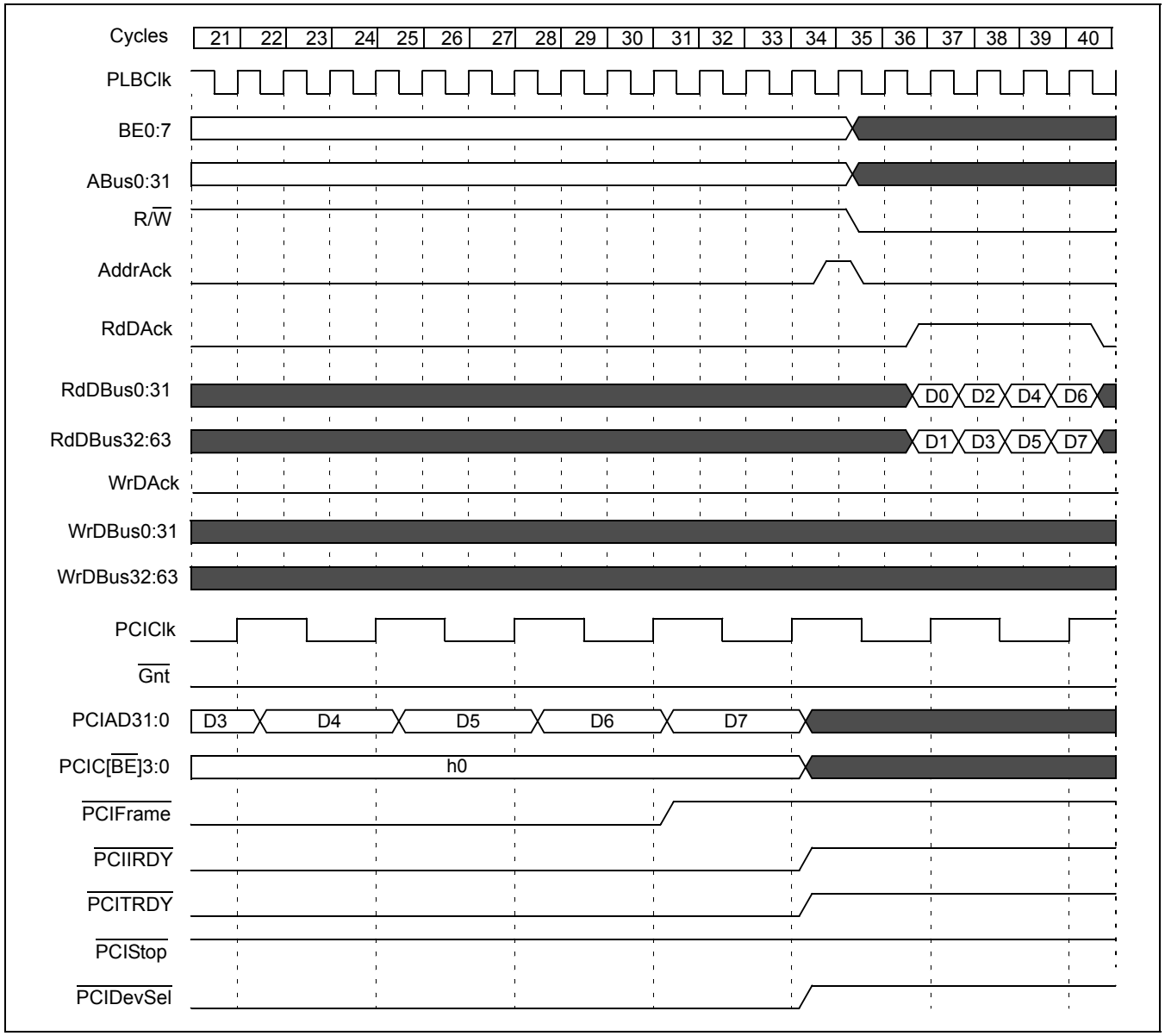


Figure 19-108. PCI Memory To SDRAM DMA Transfer (Continued)

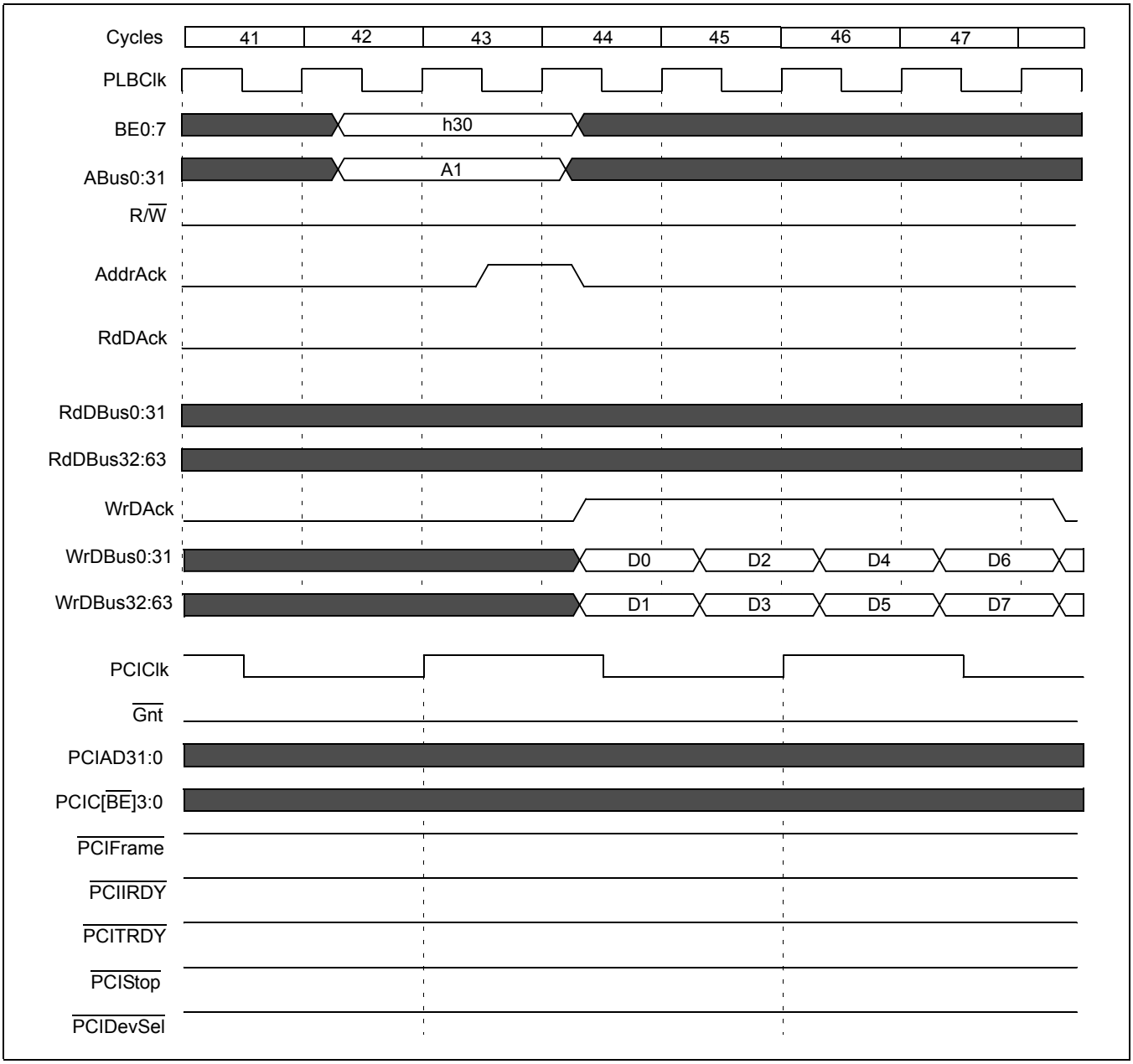


Figure 19-109. SDRAM To PCI Memory DMA Transfer

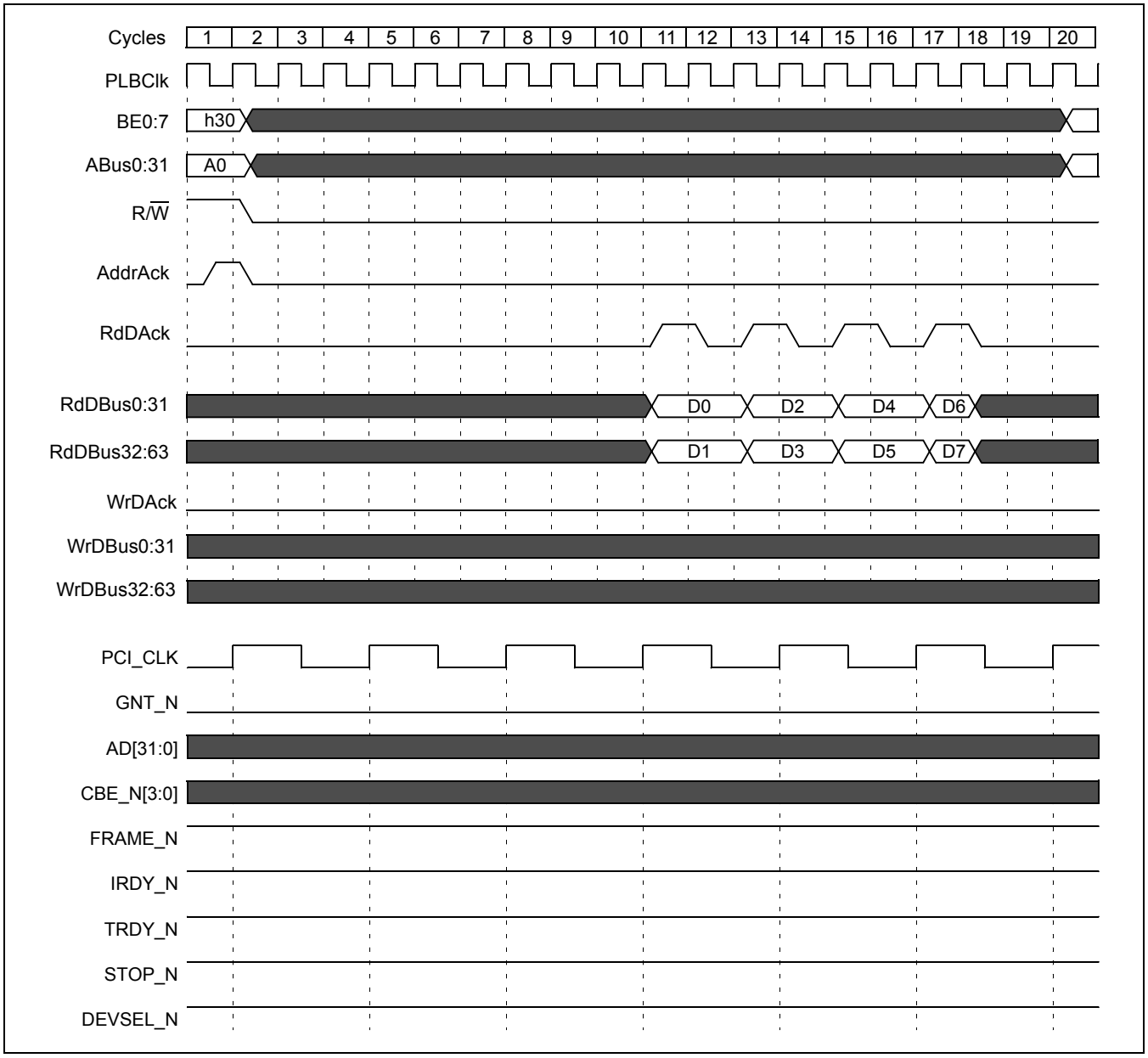


Figure 19-110. SDRAM To PCI Memory DMA Transfer (Continued)

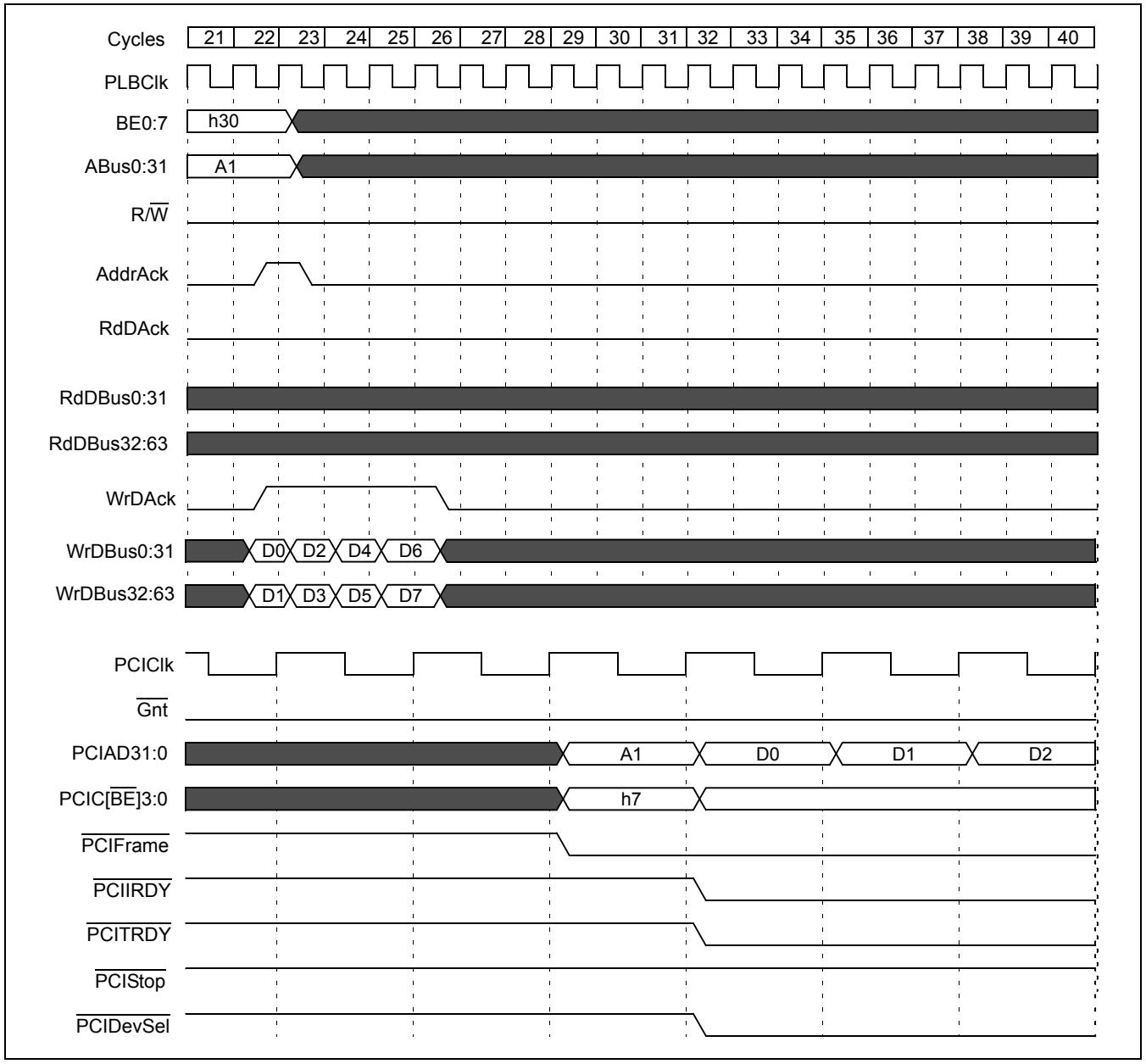
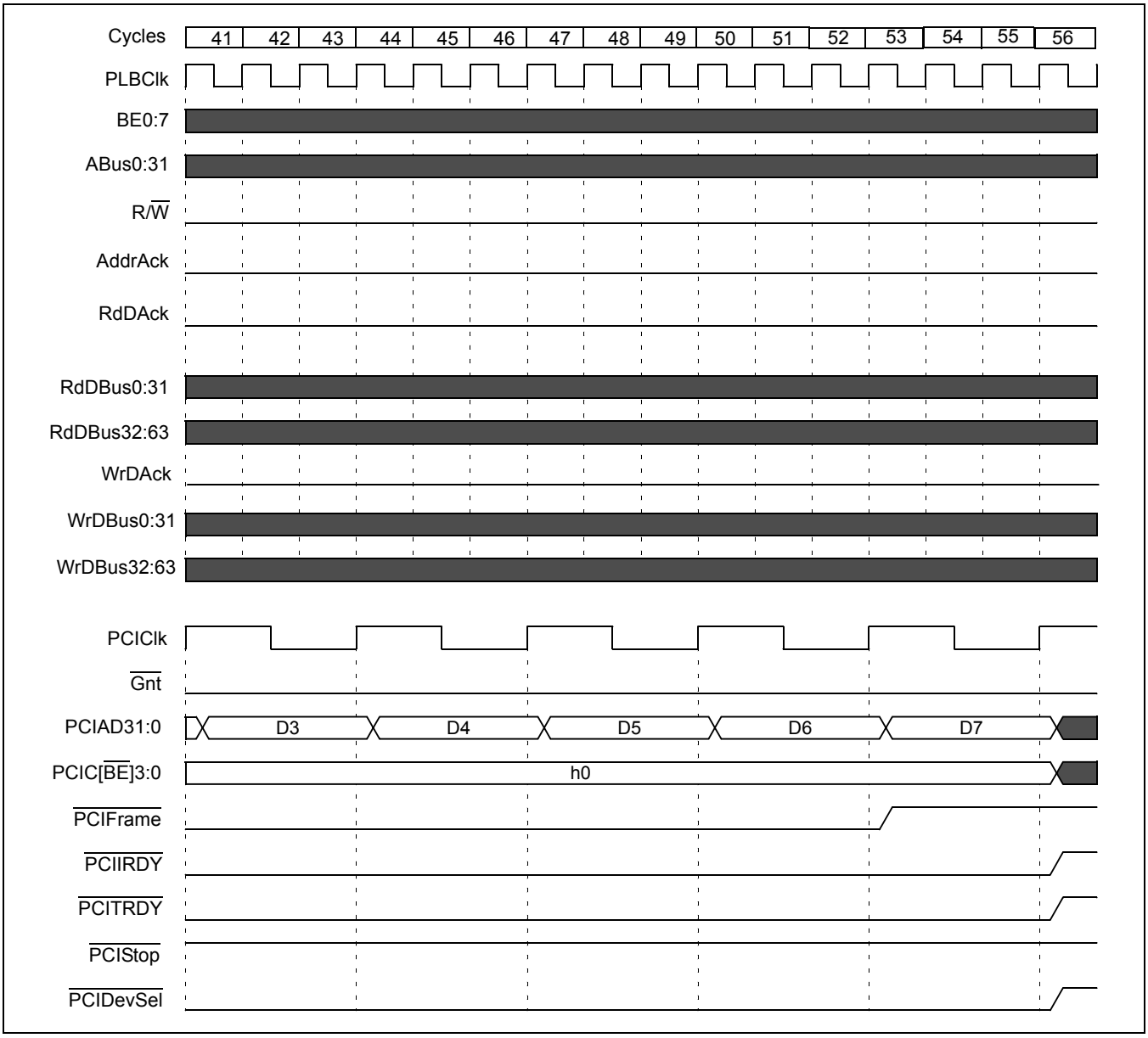


Figure 19-111. SDRAM To PCI Memory DMA Transfer (Continued)



20. External Bus Controller

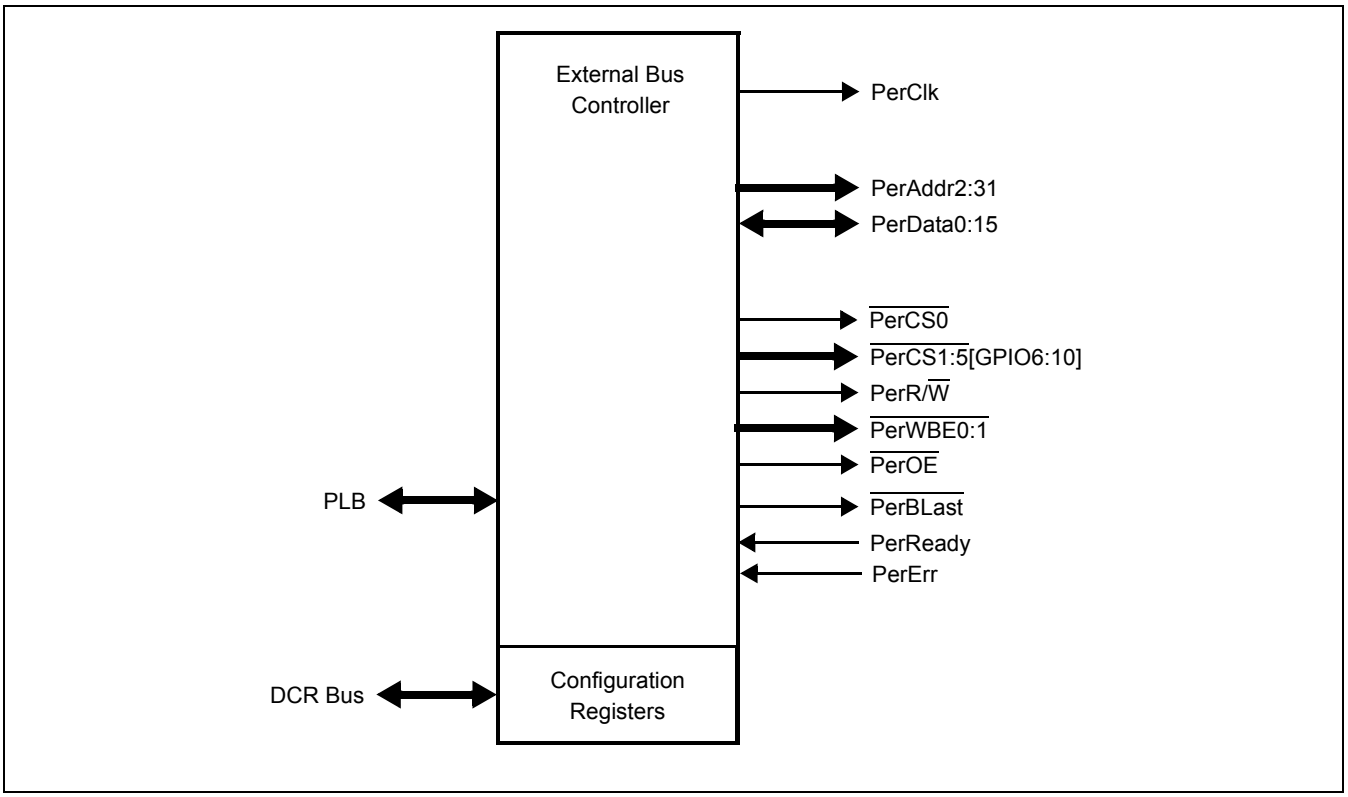
The PPC440EP External Bus Controller (EBC) provides direct attachment for most SRAM, Flash memory, and peripheral devices. The interface minimizes the amount of external glue logic needed to communicate with memory and peripheral devices, reducing embedded system device count, circuit board area, and cost.

To eliminate off-chip address decoding, the EBC provides six programmable chip selects that enable system designers to locate memory and peripherals within the PPC440EP memory map. Chip select, data bus, and associated control signal timings are programmable for both single and burst transfers. For peripherals with variable timing requirements, the EBC supports device-paced transfers with optional bus-timeout. System design is further simplified through dynamic bus sizing that supports seamlessly attaching 8- 16-bit wide memories and peripherals. Whenever a size mismatch exists between a read or write operation and the externally attached device, the EBC automatically packs or unpacks data as appropriate.

20.1 Interface Signals

The *Figure 20-1* illustrates the signal I/O between the EBC and the external peripheral bus.

Figure 20-1. EBC Signals



The following table describes signal usage and state during and after chip and system resets.

Table 20-1. EBC Signal Usage and State During and After Chip and System Resets

Signal	Usage
PerClk	Peripheral bus clock. During an EBC transfer all EBC signal transitions and data sampling occurs synchronous to PerClk.
PerAddr2:31	Peripheral address bus. PerAddr2 is the most significant bit.
PerData0:15	Peripheral data bus. PerData0 is the most significant bit.
$\overline{\text{PerCS0:5}}$	Chip selects.
PerR/$\overline{\text{W}}$	Read not write.
$\overline{\text{PerWBE0:1}}$	Write byte enables or read/write byte enables.
$\overline{\text{PerOE}}$	Output enable.
$\overline{\text{PerBLast}}$	Burst Last. Active during non-burst operations and the last transfer of a burst access.
PerReady	An input to allow external peripherals to perform device-paced transfers.
PerErr	Peripheral data error input. Sampled only during data transfers.

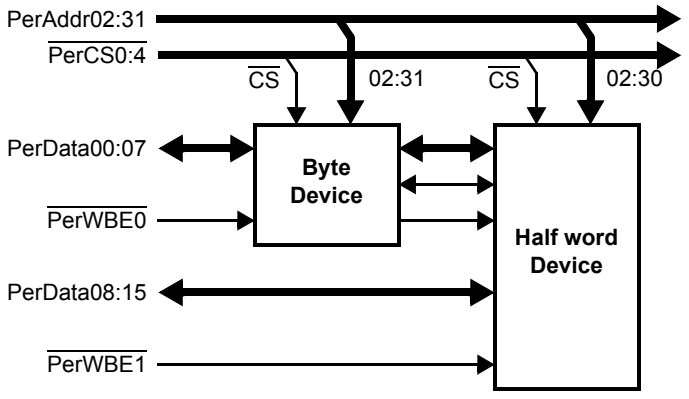
20.1.1 Interfacing to Byte and Half-Word Devices

Figure 20-2 illustrates how to interface byte and half-word devices to the peripheral data bus. When devices are connected in this way, the EBC supports burst transfers and automatically converts read and write operations to the data width of the external device. As shown in *Figure 20-2*, half word devices should not connect to PerAddr31. Instead, the active byte lanes should be inferred from PerWBE0:1, the read/write byte enables.

When a large number of byte-wide devices are attached to the peripheral data bus, the capacitive loading on byte lane 0 is much larger than the loading on byte, possibly resulting in unacceptable timing on byte lane 0.

If a bank register is configured as word-wide, then byte-wide devices may be attached to the bus in any byte lane (and accessed using byte loads and stores). External logic may be required to develop additional control signals if the data bus is used in this manner.

Figure 20-2. Attachment of Devices of Various Widths to the Peripheral Data Bus



20.1.2 Driver Enables

As shown in *Table 20-2* the output enables for the peripheral address, data and most of the EBC control signals are configurable. Setting EBC0_CFG[CSTC]=1 eliminates the need for pull-up resistors on PerCS0:5. Pull-ups are also unnecessary on the remainder of the EBC control signals when EBC0_CFG[EBTC]=1.

Both chip and system resets set EBC0_CFG[EBTC]=1 and EBC0_CFG[CSTC]=1. In most applications, clearing EBC0_CFG[CSTC] is not recommended. If EBC0_CFG[EBTC]=0, EBC control signals can change from the active state to high-Z without first being driven inactive. To prevent this, peripheral banks must be configured with at least one hold cycle (EBC0_BnAP[TH] > 0).

Table 20-2. Effect of Driver Enable Programming on EBC Signal States

EBC Operation	PerClk	PerCS0:5	PerAddr2:31 PerR/W PerWBE0:1 PerOE PerBLast	PerData0:15
Reset	High-Z	High-Z	High-Z	High-Z
Idle	Driven	EBC0_CFG[CSTC]	EBC0_CFG[EBTC]	EBC0_CFG[EBTC]
Read	Driven	Driven	Driven	High-Z
Write	Driven	Driven	Driven	Driven

If the EBC0_CFG bit is set, the signal is driven to the appropriate state during the indicated EBC operation. Otherwise, the I/O is High-Z.

20.2 Non-Burst Peripheral Bus Transactions

The timing of the PerCSn, PerOE, and PerWBE0:1 signals is programmable using the Peripheral Bank Access Parameter (EBC0_BnAP) registers. For non-burst transfers, the access parameter registers control the peripheral bus timing as follows:

- PerCSn becomes active 0–3 PerClk cycles (EBC0_BnAP[CSN]) after the address is driven.
- PerOE is driven low 0–3 PerClk cycles (EBC0_BnAP[OEN]) after PerCSn is active.
- PerBLast is active throughout the entire transfer and is driven high during the programmed hold time (EBC0_BnAP[TH]).
- PerWBE0:1 can be either write byte enables or read and write enables.

  If EBC0_BnAP[BEM]=0, PerWBE0:1 are write byte enables and:

  - PerWBE0:1 goes active 0–3 (EBC0_BnAP[WBN]) PerClk cycles after PerCSn becomes active.
  - PerWBE0:1 becomes inactive 0–3 (EBC0_BnAP[WBF]) PerClk cycles before PerCSn becomes inactive.

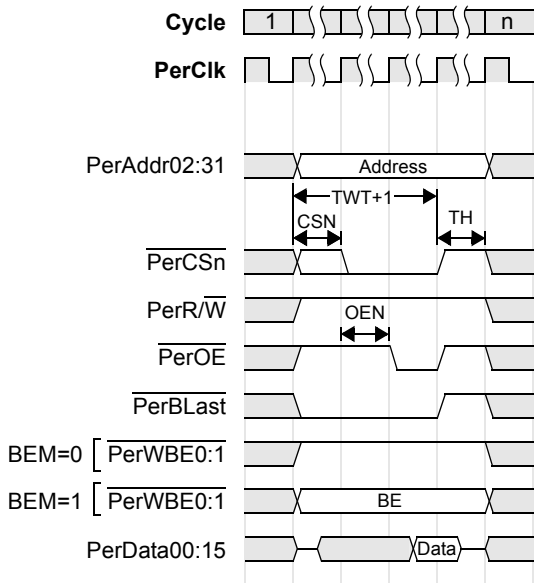
  If EBC0_BnAP[BEM]=1, PerWBE0:1 are read/write byte enables and have timing identical to the peripheral address bus. In this case the EBC0_BnAP[WBN] and EBC0_BnAP[WBF] parameters are ignored.
- 1–256 PerClk cycles (EBC0_BnAP[TWT] + 1) after the address became valid:
  - If EBC0_CFG[CSTC]=1 or EBC0_BnAP[TH]>0, PerCSn is driven high.
  - If EBC0_CFG[CSTC]=0 and EBC0_BnAP[TH]=0, PerCSn transitions directly from logic 0 to the high-impedance state.
- The parameters TWT, CSN, OEN, WBN, and WBF in EBC0_BnAP are not independent. For non-burst configured banks it is required that TWT ≥ CSN + MAX(OEN,WBN) + WBF.

- The hold time, EBC0_BnAP[TH], is programmable from 0–7 PerClk cycles. During the hold time, the peripheral address bus remains driven with the last address and all control signals are actively driven high. If the operation was a write, the peripheral data bus continues driving the last data value.
- There is no guarantee of dead cycles between transfers on the peripheral interface. If there are back-to-back transfers to the same memory bank and the number of hold cycles is programmed to zero (EBC0_BnAP[TH]=0 and EBC0_BnAP[CSN]=0), then:
  - PerCSn may not go inactive between the back-to-back transfers.
  - If EBC0_BnAP[OEN]=0, PerOE may not become inactive between the two transfers.
  - If EBC0_BnAP[WBN]=0 and EBC0_BnAP[WBF]=0, PerWBE0:1 may not go inactive between the back-to-back transfers.

20.2.1 Single Read Transfer

Figure 20-3 shows the peripheral interface timing for a single read transfer from a non-burst enabled (EBC0_BnAP[BME]=0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:1. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[TWT]+1 cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input, PerErr. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles.

Figure 20-3. Single Read Transfer



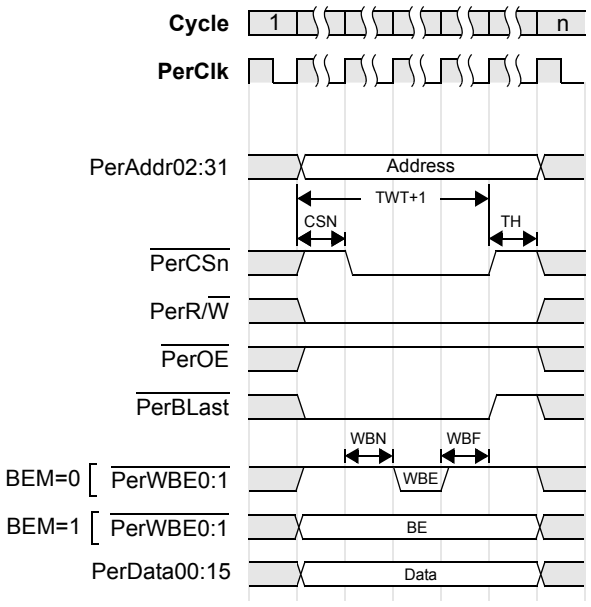
20.2.2 Single Write Transfer

Figure 20-4 shows the peripheral interface timing for a single write transfer to a non-burst enabled (EBC0_BnAP[BME]=0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

- If EBC0_BnAP[BEM]=0, byte enable mode is disabled and the PerWBE0:1 are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn. The EBC then waits until (EBC0_BnAP[TWT] – EBC0_BnAP[WBF] + 1) cycles have elapsed since the start of the transaction, then drives all the PerWBE0:1 inactive.
- If EBC0_BnAP[BEM]=1, the PerWBE0:1 lines are byte enables and have the same timing as the peripheral address bus.

After EBC0_BnAP[TWT+1] cycles elapse from the start of transfer, PerCSn and PerBLast are driven high. The EBC then waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 20-4. Single Write Transfer



20.3 Burst Transactions

Bursting is controlled on a per-bank basis by the Burst Mode Enable bit in the EBC0_BnAP registers. When enabled (EBC0_BnAP[BME]=1) this mode activates bursting for all cache line fills and flushes, PLB burst transfers to the EBC, and all packing and unpacking operations. When bursting is enabled:

- PerCSn becomes active 0–3 (EBC0_BnAP[CSN]) PerClk cycles after the address becomes valid.
- PerCSn is no longer actively driven:
  - 1–32 (EBC0_BnAP[FWT]+1) PerClk cycles after the address becomes valid when a single transfer occurs to a burst-enabled bank.

  - 1–8 (EBC0_BnAP[BWT]+1) PerClk cycles after the last address becomes valid during a burst:
    - If EBC0_CFG[CSTC]=1 or EBC0_BnAP[TH]>0, PerCSn is driven high.
    - If EBC0_CFG[CSTC]=0 and EBC0_BnAP[TH]=0, PerCSn transitions directly from logic 0 to the high-impedance state.
- During read operations PerOE is driven low 0–3 (EBC0_BnAP[OEN]) PerClk cycles after PerCSn is active. PerOE goes inactive when PerCSn goes inactive.
- For bursts, the EBC drives a new address (EBC0_BnAP[FWT]+1) + N*(EBC0_BnAP[BWT]+1) cycles after the start of the transaction, where N = 0, 1, 2, …
- Addresses during a burst may "wrap." For example, cache line fills are processed critical word first.
- During write operations, the write data is driven concurrent with each address.
- PerWBE0:1 can be either write byte enables or read and write enables.

  If EBC0_BnAP[BEM]=0, PerWBE0:1 are write byte enables and:
  - For the first transfer of a burst, or a single transfer to a burst enabled bank, the appropriate write byte enables go low 0–3 (EBC0_BnAP[WBN]) cycles after PerCSn becomes active. The EBC then waits until EBC0_BnAP[FWT] – EBC0_BnAP[WBF] + 1 cycles have elapsed since the start of the transaction and drives PerWBE0:1 inactive.
  - The remaining transfers of the burst are similar, except that PerWBE0:1 becomes active at the same time that each new address is driven on the interface. The PerWBE0:1 remain low for (EBC0_BnAP[BWT] + 1) – EBC0_BnAP[WBF] cycles.

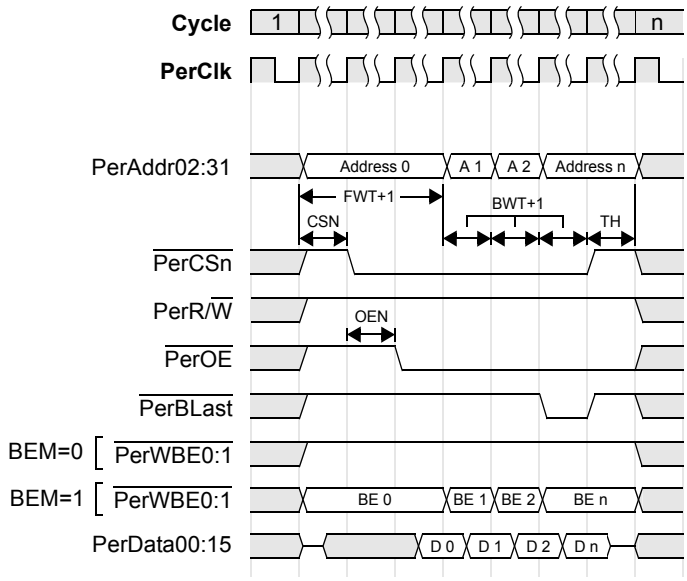
  If EBC0_BnAP[BEM]=1, PerWBE0:1 are byte enables that have timing identical to the peripheral address bus. In this case the EBC0_BnAP[WBN] and EBC0_BnAP[WBF] parameters are ignored.
- PerBLast is active throughout the entire last (or only) transfer of a burst operation and is deactivated during the programmed hold time (EBC0_BnAP[TH]).
- Access bank parameters CSN, OEN and WBN apply to the first (or only) transfer of a burst, while WBF applies to all transfers. It is required that FWT ≥ CSN + MAX(OEN,WBN) + WBF and BWT ≥ WBF.
- Hold time (EBC0_BnAP[TH]) is programmable from 0 to 7 cycles. During the hold time, the peripheral address bus remains driven and all control signals are driven inactive. If the operation was a write, the peripheral data bus continues driving the write data.
- There is no guarantee of dead cycles between transfers on the peripheral interface. If there are back-to-back transfers to the same memory bank and the number of hold cycles is programmed to zero (EBC0_BnAP[TH]=0) and EBC0_BnAP[CSN]=0, then:
  - PerCSn may not go inactive between the back-to-back transfers.
  - If EBC0_BnAP[OEN]=0, PerOE may not become inactive between the two transfers.
  - If EBC0_BnAP[WBN]=0 and EBC0_BnAP[WBF]=0, PerWBE0:1 may not go inactive between the back-to-back transfers.

20.3.1 Burst Read Transfer

Figure 20-5 shows the peripheral interface timing for a burst read transfer from a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:1. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[FWT]+1 cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input, PerErr.

The next address of the burst is then driven and after EBC0_BnAP[BWT]+1 cycles the EBC performs the next read. The remaining items in the burst are read in the same manner, except that PerBLast is active during the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 20-5. Burst Read Transfer



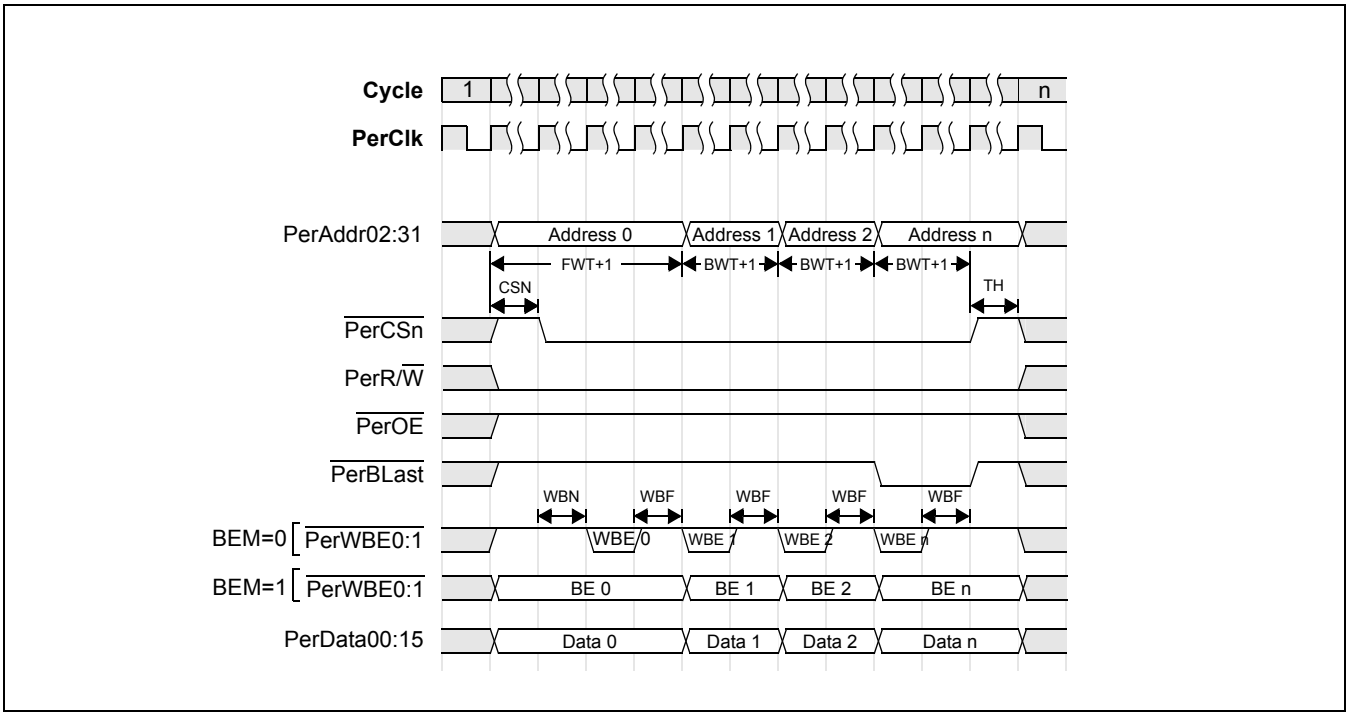
20.3.2 Burst Write Transfer

Figure 20-6 shows the peripheral interface timing for a burst write transfer to burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. At this point the signalling sequence depends on whether byte enable mode is enabled for the bank.

- If EBC0_BnAP[BEM]=0, byte enable mode is disabled and PerWBE0:1 are write byte enables. In this case, the appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn. The EBC then waits until (EBC0_BnAP[FWT] +1) - EBC0_BnAP[WBF] cycles have elapsed since the start of the transaction and drives PerWBE0:1 inactive. EBC0_BnAP[WBF] cycles are then allowed to elapse after which the address and data are output for the second element in the burst. As shown in *Figure 20-6*, the EBC transfers the subsequent data items in a similar manner.
- If EBC0_BnAP[BEM]=1, the PerWBE0:1 lines are byte enables and have the same timing as the peripheral address bus. In this configuration external logic may be necessary to latch write data at the appropriate times.

PerBLast goes low at the beginning of the last transfer to indicate that the burst is ending. The EBC then drives PerCSn and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 20-6. Burst Write Transfer



20.4 Device-Paced Transfers

For device-paced transfers, the EBC provides two distinct modes: Sample On Ready enabled and Sample On Ready disabled. The selection of these modes is controlled, for each bank, by EBC0_BnAP[SOR]. When Sample On Ready is enabled (EBC0_BnAP[SOR] = 1) data is transferred on the PerClk rising edge when PerReady is sampled active. When Sampling On Ready is disabled (EBC0_BnAP[SOR] = 0), PerReady sampled active causes the data transfer to occur in the next cycle, which results in an additional cycle of wait time.

The ready signal (PerReady) is an input which allows the insertion of externally generated (device-paced) wait states. PerReady is monitored only when EBC0_BnAP[RE]=1.

- For burst disabled banks (EBC0_BnAP[BME] = 0) sampling of the PerReady input starts EBC0_BnAP[TWT] cycles after the beginning of the transfer. Wait states are inserted and sampling continues once per cycle until either PerReady is high when sampled or a timeout occurs.
- For burst enabled banks (EBC0_BnAP[BME] = 1) sampling of the PerReady input starts EBC0_BnAP[FWT] PerClk cycles after the beginning of the first transfer of a burst, and EBC0_BnAP[BWT] cycles after the beginning of subsequent transfers of the burst. Sampling continues once per cycle until either PerReady is sampled high or a timeout occurs.
- When EBC0_BnAP[SOR] = 1 data transfer occurs in the same cycle where PerReady is sampled active. In contrast, if EBC0_BnAP[SOR]=0 the data transfer occurs in the next cycle.
- When EBC0_BnAP[SOR] = 1, if the hold time is set to zero, EBC0_BnAP[TH] = 0, the programmed hold time is ignored and the EBC performs the transaction with one hold cycle.
- When EBC0_BnAP[RE] = 1, the Write Byte Enable Off parameter must be programmed to 0, EBC0_BnAP[WBF] = 0. As a result, during device-paced burst write transfers PerWBE0:1 does not become inactive between data elements.

The EBC may be programmed to wait only a limited time for PerReady to become active, or it may be programmed for unlimited wait. If EBC0_CFG[PTD] = 1, timeouts are disabled and the EBC waits indefinitely for an active PerReady.

If EBC0_CFG[PTD] = 0, device-paced timeouts are enabled and the EBC only waits for the number of PerClk cycles programmed in EBC0_CFG[RTC]. The timeout counter is reset whenever the peripheral address changes. In this manner each data element within a device-paced burst transaction is treated separately for the purposes of determining whether a timeout error occurs. If PerReady does not become active before the timeout counter reaches the value programmed into EBC0_CFG[RTC], the transfer is aborted and an error is signalled. See *Error Reporting* on page 488 for details about how timeout errors are logged.

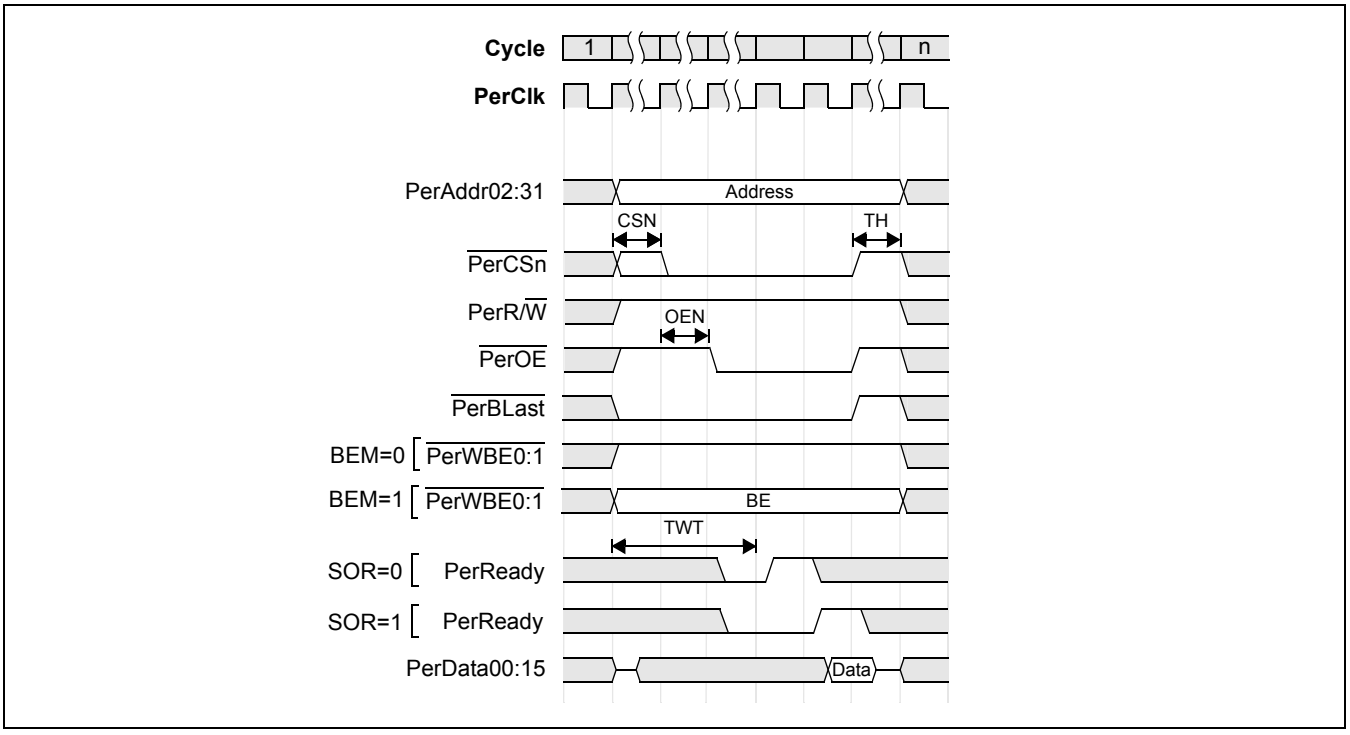
20.4.1 Device-Paced Single Read Transfer

Figure 20-7 shows the peripheral interface timing for a device-paced single read transfer from a burst disabled (EBC0_BnAP[BME]=0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:1. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

Once PerReady is sampled active if Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle. The EBC then samples the data bus and the peripheral error input, PerErr. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending EBC transfers to occur.

Figure 20-7. Device-Paced Single Read Transfer



20.4.2 Device-Paced Single Write Transfer

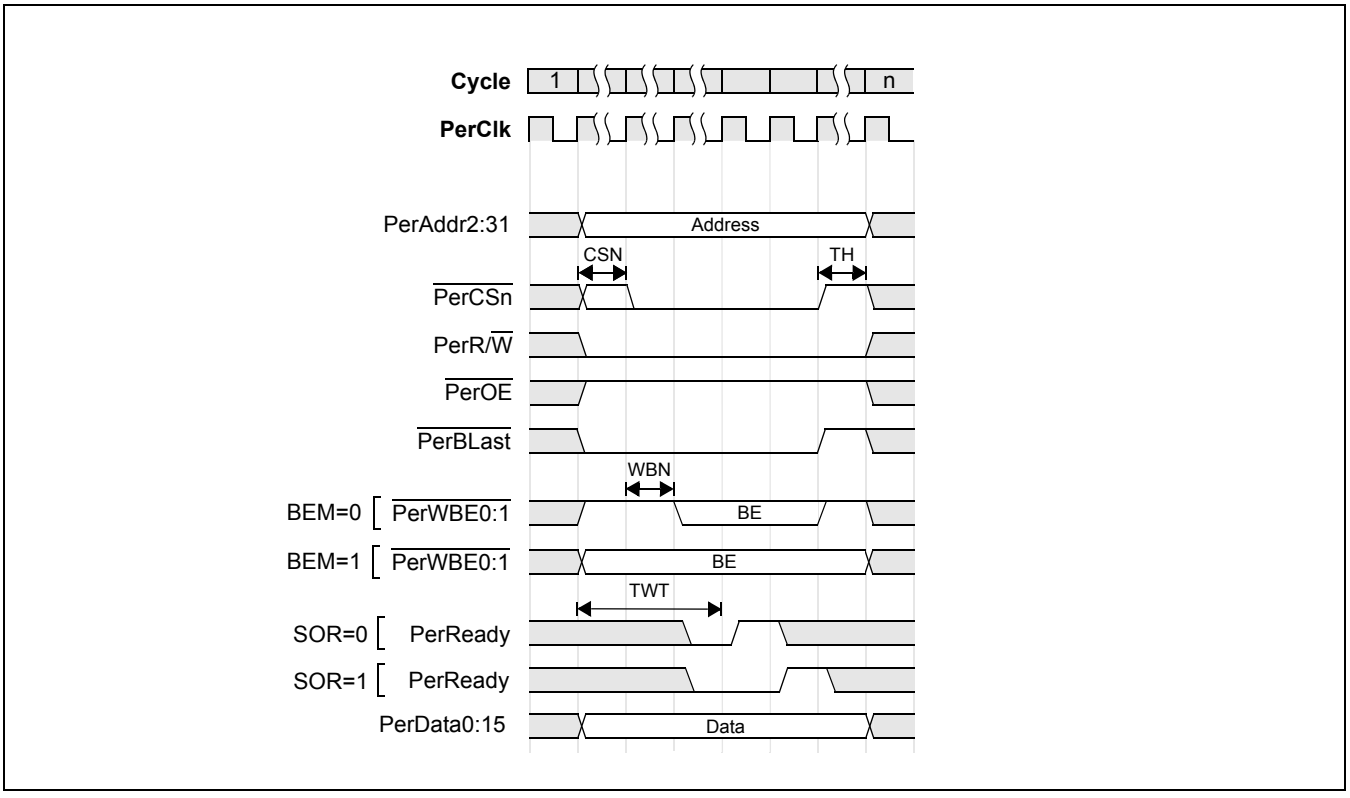
Figure 20-8 shows the peripheral interface timing for a device-paced single write transfer from a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. At this point the signalling sequence depends on whether byte enable mode is enabled for the particular bank.

- If EBC0_BnAP[BEM]=0, byte enable mode is disabled and the PerWBE0:1 are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn went low. PerWBE0:1 return high on the same PerClk edge that the write data is transferred (see below).
- If EBC0_BnAP[BEM]=1, the PerWBE0:1 lines are byte enables and have the same timing as the peripheral address bus.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sample PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle. At this point, the write transfer occurs and the EBC reads the peripheral error input, PerErr. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles.

Figure 20-8. Device-Paced Single Write Transfer



20.4.3 Device-Paced Burst Read Transfer

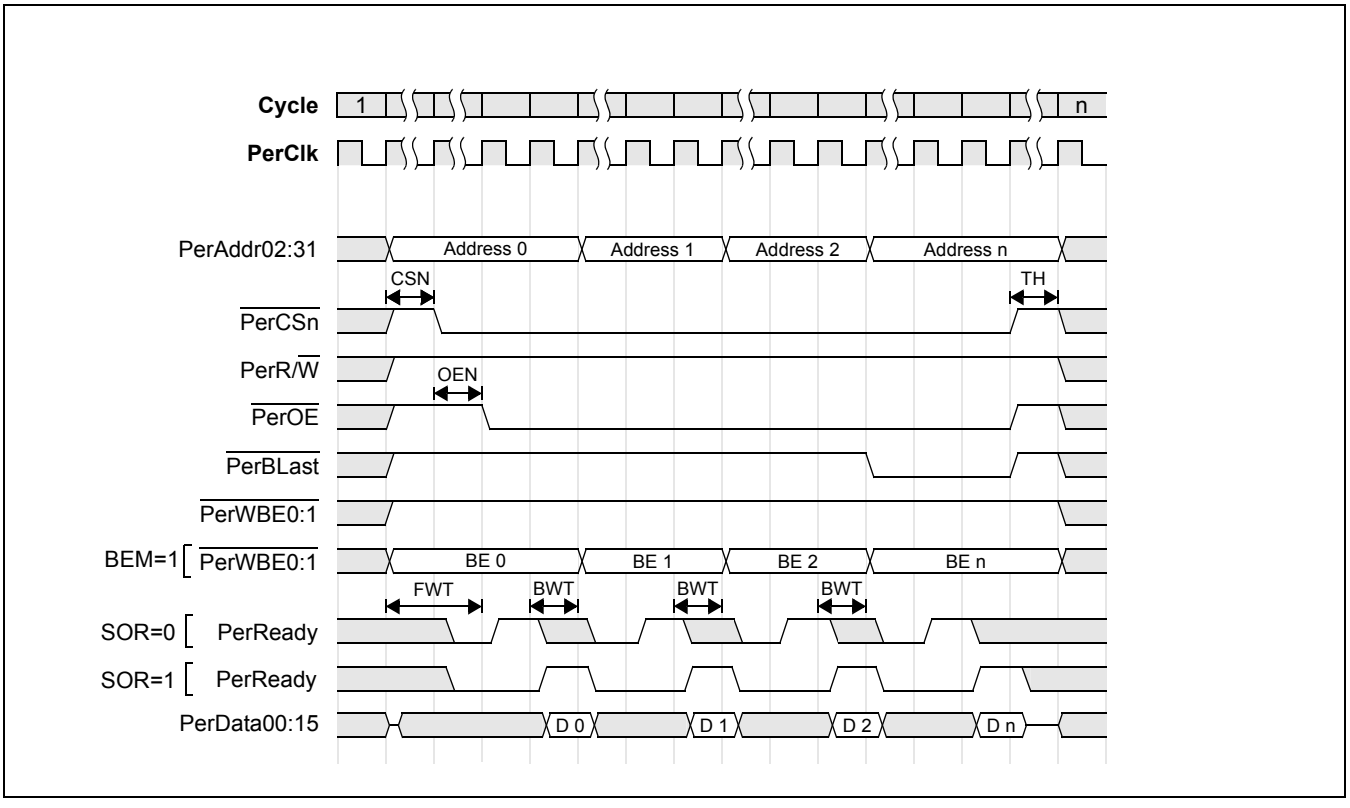
Figure 20-9 shows the peripheral interface timing for a device-paced burst read transfer from a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:1. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[FWT] cycles have elapsed since the start of the transaction and begins sampling PerReady.

If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle before sampling read data. The EBC then reads the data bus and the peripheral error input, PerErr.

The next address of the burst is then driven and after EBC0_BnAP[BWT] cycles the EBC waits for PerReady as described above. The remaining items in the burst are read in this same manner, except that PerBLast is active during the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 20-9. Device-Paced Burst Read Transfer



20.4.4 Device-Paced Burst Write Transfer

Figure 20-10 shows the peripheral interface timing for a device-paced burst write transfer to a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

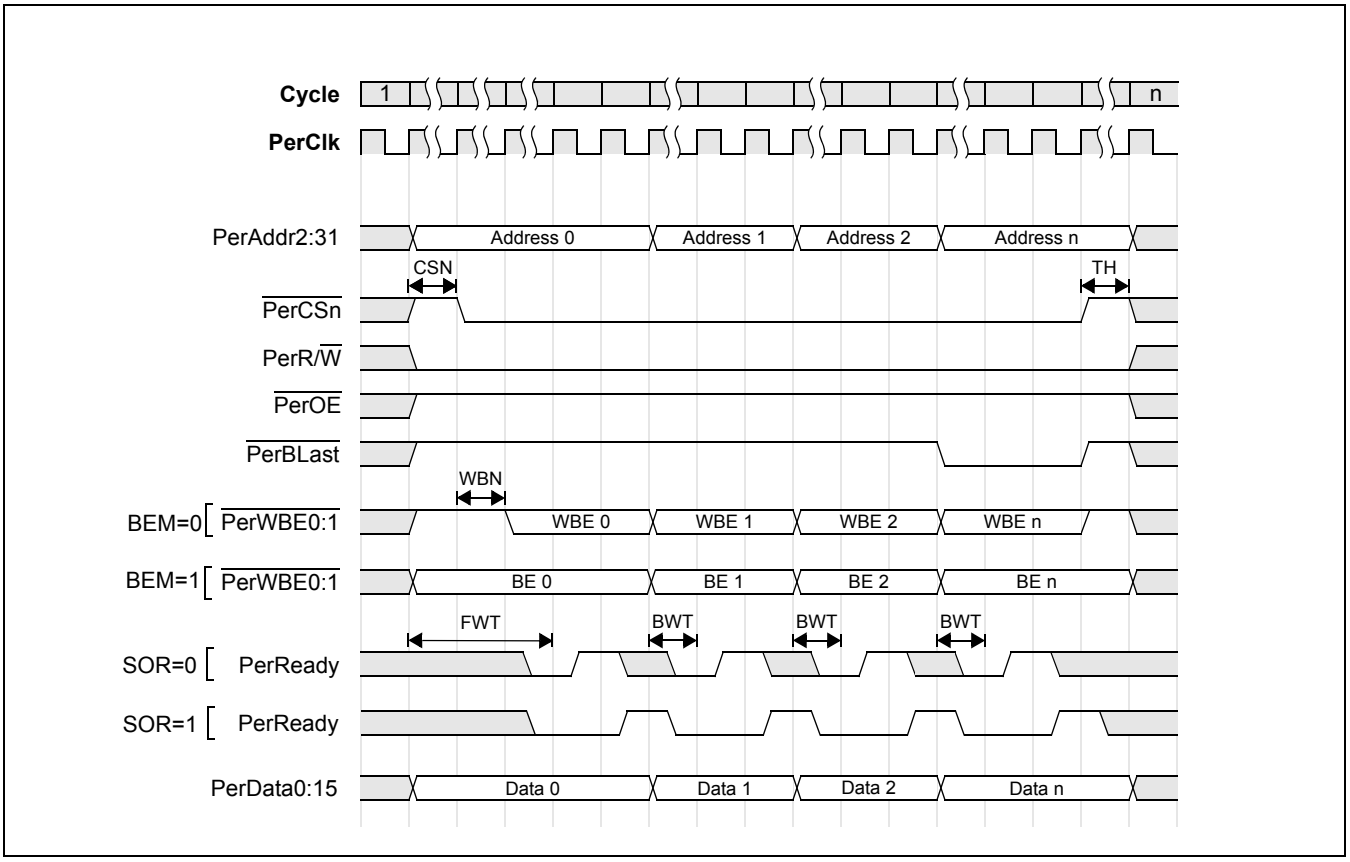
- If byte enable mode is disabled (EBC0_BnAP[BEM]=0) PerWBE0:1 are write byte enables. In this case the appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn becomes active for the first element in a burst and EBC0_BnAP[WBN] cycles after each new address for the remainder of the burst. If EBC0_BnAP[WBN]<>0, PerWBE0:1 is driven inactive on the same PerClk edge that write data is transferred (see below). Otherwise, PerWBE0:1 remains low for all data elements in the burst.
- If EBC0_BnAP[BEM]=1, PerWBE0:1 are byte enables and have the same timing as PerAddr:31.

The EBC then waits until EBC0_BnAP[FWT] cycles have elapsed since the start of the transaction and begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle. At this point the write transfer occurs and the EBC reads the peripheral error input, PerErr.

The next address of the burst is then driven and after EBC0_BnAP[BWT] cycles the EBC waits for PerReady as described above. The remaining items in the burst are transferred in this same manner, except that PerBLast is active for the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 20-10. Device-Paced Burst Write Transfer

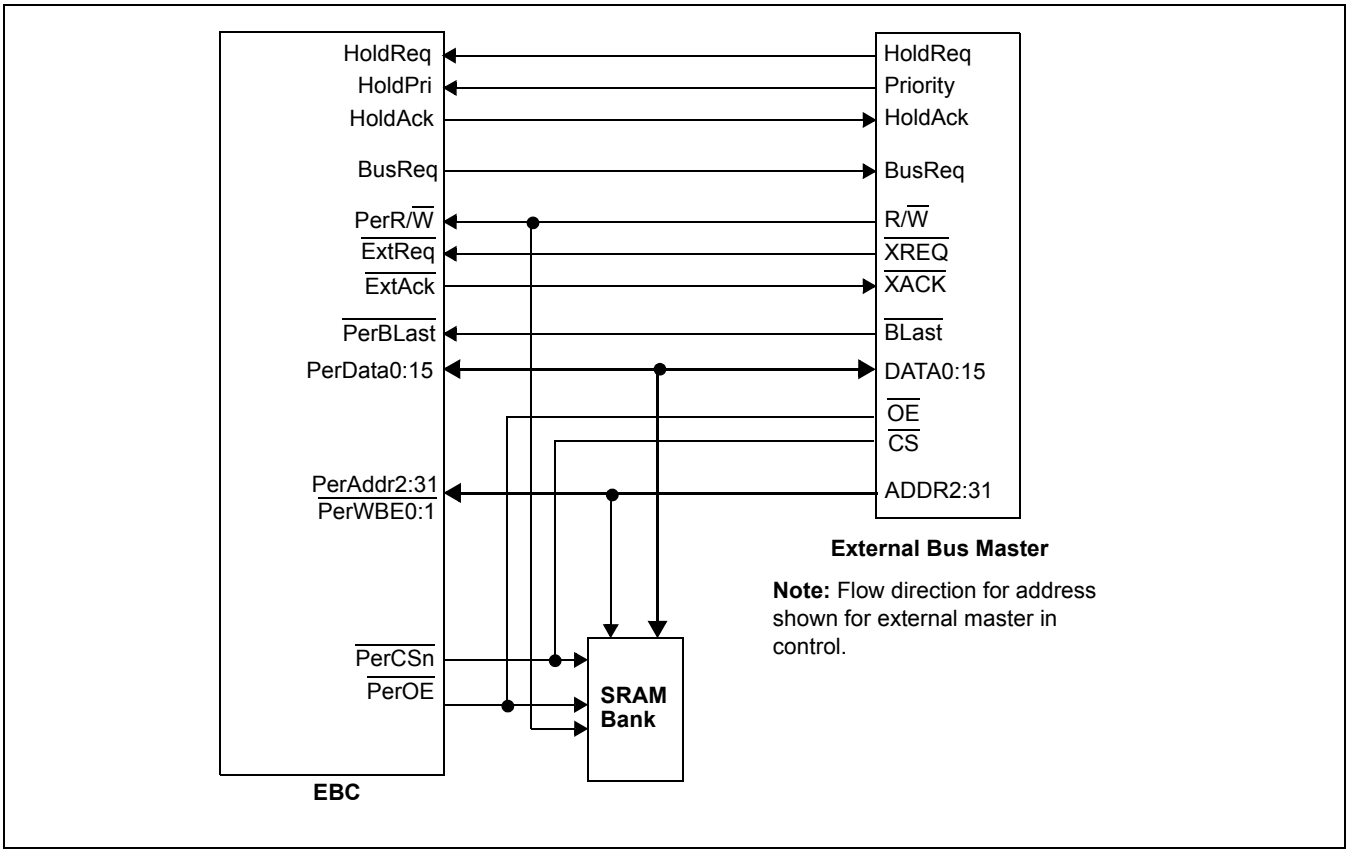


20.5 External Bus Master Interface

The EBC includes an External Bus Master (EBM) interface supporting a shared bus protocol which allows an EBM to gain control of the peripheral bus. Once an external master has been granted access to the peripheral interface it can read and write all PLB- and OPB-addressable memory, with the exception of devices controlled by the EBC. Typical destinations for EBM transactions are PCI address space and SDRAM memory. SDRAM memory is the typical destination for an EBM transaction. For EBC-attached peripherals and memory, the external master is required to directly control the target.

Figure 20-11 shows a sample interconnection of the EBC, one SRAM bank, and one external bus master. While only one SRAM bank is shown, the bus master could access all eight of the SRAM banks. Also, with the appropriate arbitration logic, multiple bus masters may be used in a system.

Figure 20-11. Sample External Bus Master System



The EBM interface includes both arbitration and data transfer functions. The arbiter grants the peripheral bus to either the EBC or an external master, while the data path logic implements an external master to PLB bridge function. This bridge function translates the EBM interface protocol to that required by the on-chip PLB bus.

20.5.1 Arbitration

To gain control of the peripheral bus, the external bus master places an active level on the HoldReq input and drives the HoldPri signal to the desired value (see *Figure 20-12*). The HoldPri input selects the priority of the external master's transactions relative to other EBC operations. Internal to the PPC440EP all reads and writes that target the EBC are assigned the priority 0b10.

If HoldPri=0 the external master priority is set to EBC0_CFG[EMPL], whereas HoldPri=1 sets the priority to EBC0_CFG[EMPH]. *Table 20-3* details the only unique ways of programming the external master priority fields in EBC0_CFG, along with the arbiter's response.

Table 20-3. External Master Arbitration.

EMPL	EMPH	HoldPri	When HoldReq=1 HoldAck Becomes Active	When HoldAck=1 BusReq Goes Active
0b11	0b11	X	After any active EBC operation completes.	Never.
0b00	0b11	0	If no EBC transfers are active or pending.	If an EBC transfer is pending.
		1	After any active EBC operation completes.	Never.
0b00	0b00	X	If no EBC transfers are active or pending.	If an EBC transfer is pending.

When an external bus master requests the bus by driving HoldAck=1, the EBC finishes any transfer in progress (except for bursts) before arbitrating between any pending PLB request and the external master request. If the external master presents a higher priority request during a PLB burst, the burst is terminated. Note that processor cache reads and writes are PLB line operations, not PLB bursts, and are never interrupted.

As detailed in *Table 20-4*, the arbitration logic drives HoldAck active either when the current EBC transaction completes or when there are no EBC transfers pending. Once the external master has been granted the bus, it may either directly control devices on the peripheral bus or read and write PLB- and OPB-addressable memory. If the EBC detects a pending PLB request when the external master owns the peripheral bus (HoldAck=1), BusReq may be asserted to signal the external master that the PPC440EP wants to regain ownership of the peripheral bus. Table 20-3 lists the cases where a PLB request causes BusReq to go active. To ensure fairness and optimize bus utilization, an external master that receives an active BusReq should complete its current transaction and then relinquish ownership of the peripheral bus by driving HoldReq inactive. This is important since the EBC cannot reclaim ownership of the external bus until the external master negates HoldReq.

Table 20-4 details the usage of the EBC I/Os when an external master has been granted ownership of the peripheral bus. The usage statements in this table for signals other than HoldPri, HoldReq, HoldAck and BusReq do not apply if an external master directly controls a device on the peripheral bus.

Table 20-4. Signal States During Hold Acknowledge (HoldAck=1)

Signal Name	State	Usage
HoldPri	Input	Requested priority for external master tenure. HoldPri must not change state when HoldReq=1.
HoldReq	Input (=1)	External master must maintain its request when it owns the bus.
HoldAck	Output (=1)	External master is receiving an active bus grant.
BusReq	Output	See *Table 20-3*
ExtReq	Input	Indicates the external master is ready to transfer data.
ExtAck	Output	Indicates to the external master that a data transfer occurred.
PerAddr2:31	Input	Requested address from external master.
$\overline{\text{PerWBE0:1}}$	Input	Selects the requested byte(s) for reads and writes.
PerR/$\overline{\text{W}}$	Input	Determines if the operation is a read or write.
$\overline{\text{PerBLast}}$	Input	Active during single transfers and the last transfer of a burst.
PerData0:15	I/O	Read and write data.
$\overline{\text{PerOE}}$	High-Z	Unused.
PerErr	High-Z	Unused.

20.5.2 Transaction Overview

The EBM interface supports direct attachment of 8-, 16-, and 32-bit masters. By programming the width of the external master into EBC0_CFG[EMS] the interface accepts write data and provides read data at the appropriate width for the master. The EBM interface includes a 32-byte data buffer, used for both read and write operations between the EBM and PLB- and OPB-mapped memory locations. While write operations only use the buffer during bursts, all reads prefetch one double word and burst reads prefetch EBC0_CFG[BPF] double words from the source memory into the buffer. This prefetched data remains in the buffer until either a write operation is performed or a read is requested to a different 32-byte block of memory.

To provide the best possible performance, the EBM interface supports both single and burst transactions. Single read transfers result in the EBM reading and buffering a 64-bit double word from the requested memory address. The requested read data is then serviced from within this double word. If the next operation on the EBM interface is a read and targets this same double word, it is serviced directly from the buffer. Burst reads are similar, except that the EBM prefetches four double words beginning with the requested word.

Single write transfers result in a separate PLB transaction for each data item. To improve performance, burst writes are gathered in the 32-byte buffer and forwarded in a single PLB transaction to the target memory.

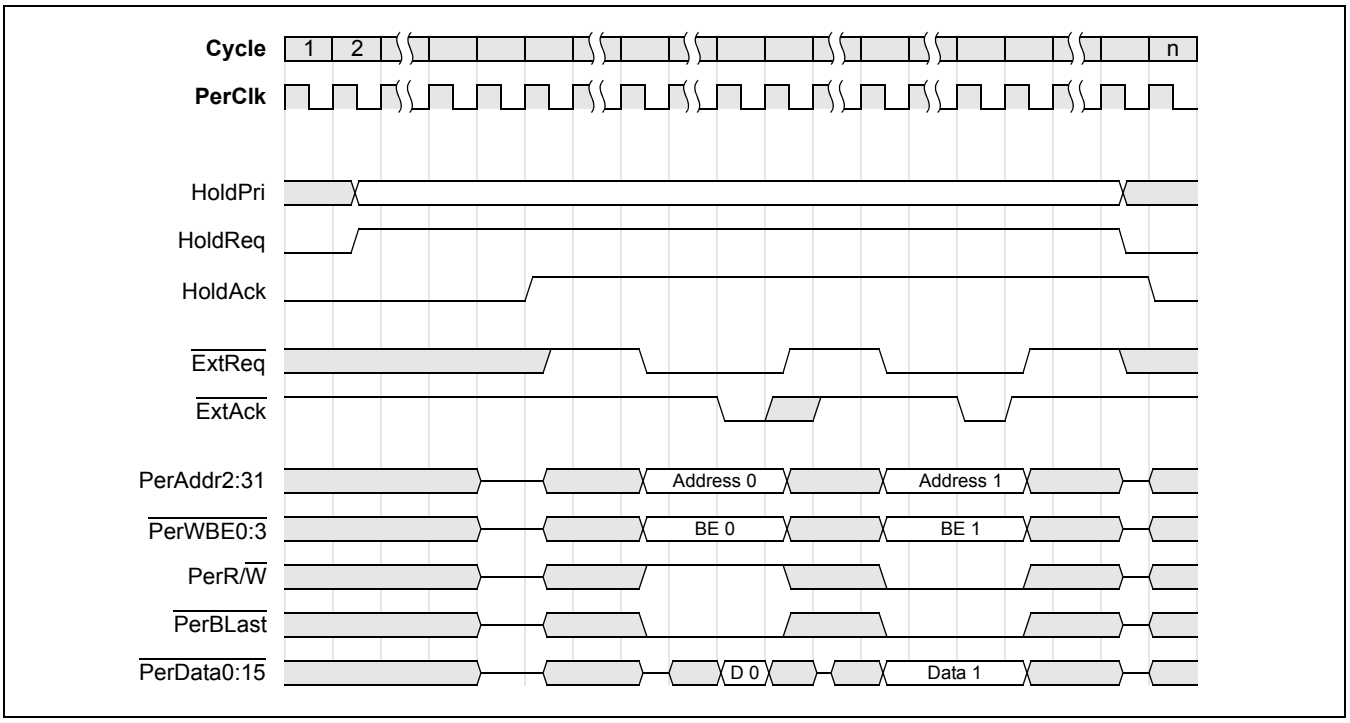
20.5.3 Single Read and Single Write Transfers

Figure 20-12 illustrates external master bus arbitration along with a single read and signal write transfer. An external master requests ownership of the peripheral bus by driving HoldReq active along with the desired priority on HoldPri. Observe that HoldPri must be held at constant value throughout the entire external master tenure. After two or more PerClk cycles the arbiter will grant the peripheral bus to the external master. The delay from when the external master asserts HoldReq to when HoldAck becomes active is variable and depends on any EBC transaction that may be in progress or pending, the level on HoldPri, and the programming in EBC0_CFG[EMPL] and EBC0_CFG[EMPH].

Once the external master is granted the peripheral bus (HoldAck=1), it may either directly control a device on the EBC or issue read and write transactions to the external master interface. This waveform and the ones that follow apply only to the later case. Additionally, cycles shown with breaks in the clock may not be present or may extend for multiple cycles.

To perform a single read operation the external master must:
- Place the desired address on PerAddr2:31.
- Indicate the requested data byte(s) on PerWBE0:1.
- Drive PerR/W high.
- Assert PerBLast to mark this as a single transfer.
- Request the transfer by driving and holding ExtReq low.

Figure 20-12. External Master Arbitration, Single Read and Single Write



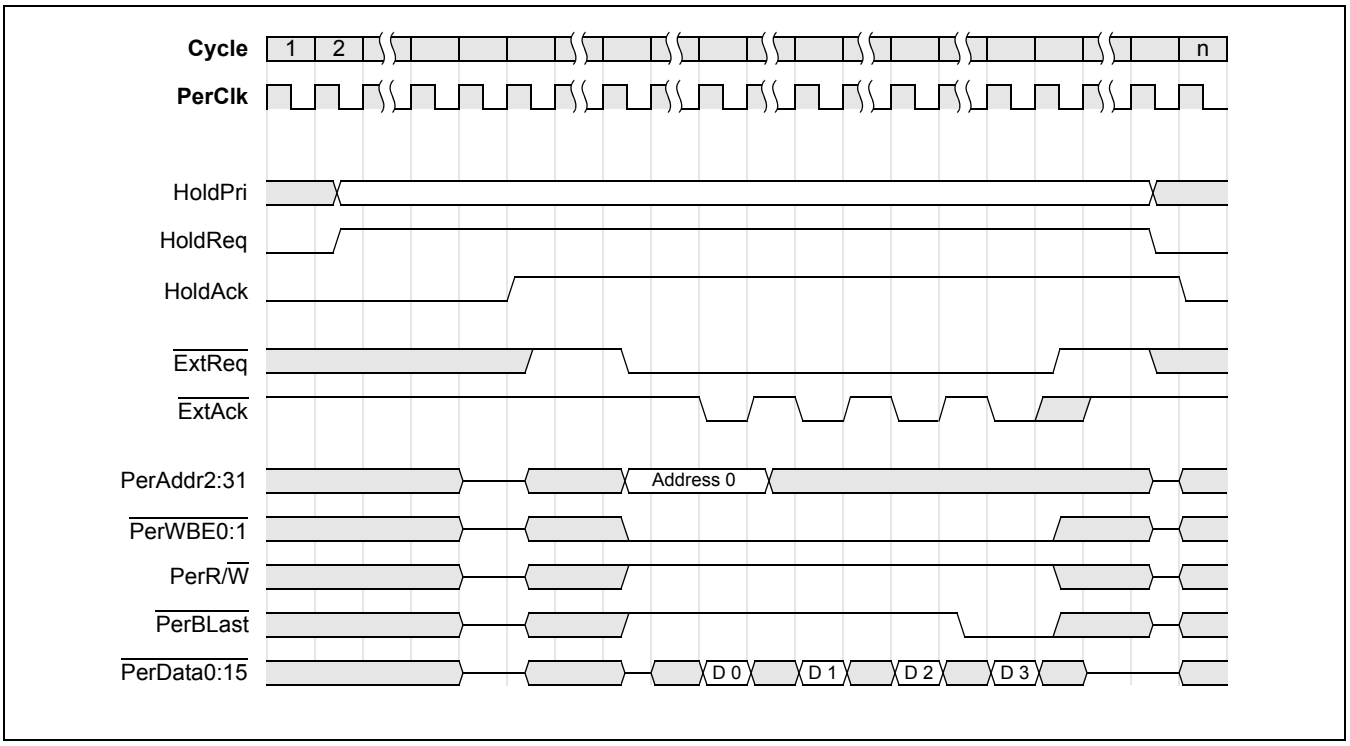
The EBM interface then converts this read request into a PLB transaction and reads the target memory location. ExtAck then goes low when read data is available on PerData. Note that the external master must not remove ExtReq until the cycle after ExtAck becomes active. In addition, ExtReq must be high for at least one cycle between all external master transactions.

Write transfers are similar to reads except that the write data must be provided along with the address and PerR/W is low to indicate a write. As with reads, ExtReq must be held until the cycle after ExtAck is received.

20.5.4 Burst Read Transfer

Burst reads are preferred when accessing sequential addresses as they provide much better performance. *Figure 20-13* illustrates an external master burst read transaction. A burst read differs from a single read in that PerBLast is held inactive for all but the last transfer of the burst. In addition, the EBM only requires the address of the first burst element. Following each ExtAck the EBM uses the size of the master configured in EBC0_CFG[EMS] to increment its internal address counter as appropriate.

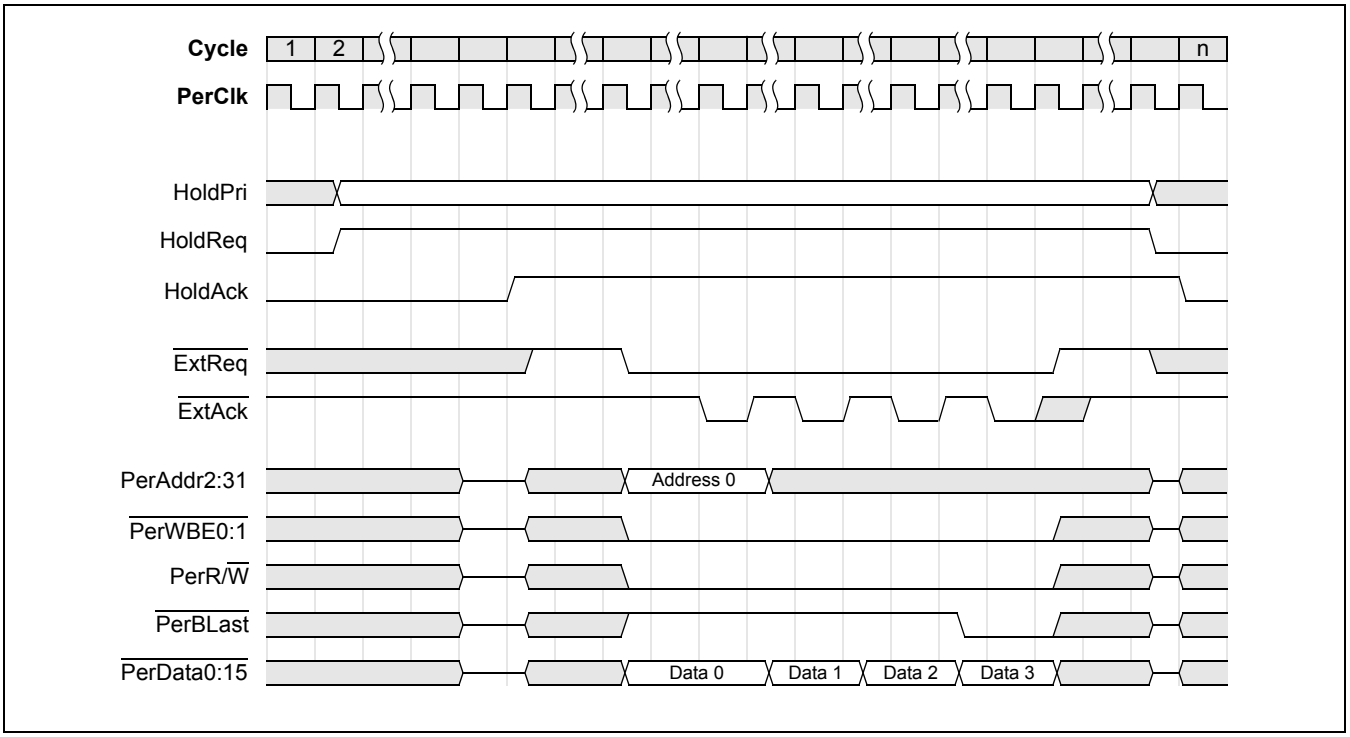
Figure 20-13. External Master Burst Read



20.5.5 Burst Write Transfer

Burst writes are preferred when accessing sequential addresses as they provide much better performance. *Figure 20-14* illustrates an external master burst write transaction. A burst write differs from a single write in that PerBLast is held inactive for all but the last transfer of the burst. In addition, the EBM only requires the address of the first burst element. Following each ExtAck the EBM uses the size of the master in EBC0_CFG[EMS] to increment its internal address counter as appropriate.

Figure 20-14. External Master Burst Write



20.5.6 External Master Error Interrupts

The EBM can generate an interrupt if a PLB error is encountered while read or writing data. As example, an DDR SDRAM uncorrectable ECC error will cause and EBM interrupt. Please refer to the UIC Chapter for additional information regarding interrupts.

20.6 EBC Registers

All EBC configuration registers are accessed with the move-to-DCR (**mtdcr**) and move-from-DCR (**mfdcr**) instructions using indirect addressing. In indirect addressing, the **mtdcr** and **mfdcr** instructions access a given EBC0 register via its address offset which is contained in the EBC0_CFGADDR register. The data for the specified register is moved to or moved from the EBC0_CFGDATA register as described below.

Table 20-5. EBC DCR Addresses

Register	DCR Address	Access	Description
EBC0_CFGADDR	0x0012	R/W	External Bus Controller Configuration Address Register
EBC0_CFGDATA	0x0013	R/W	External Bus Controller Configuration Data Register

Figure 20-15. EBC Configuration Address Register (EBC0_CFGADDR)			
0:26		Reserved	
27:31	DCRA	DCR Address Offset	

Figure 20-16. EBC Configuration Data Register (EBC0_CFGDATA)			
0:31	DCRD	DCR Data Port	

Table 20-6 lists the indirectly accessed EBC configuration and status registers.

Table 20-6. EBC Configuration and Status Registers

Mnemonic	Address Offset	Access	Description	Page
EBC0_B0CR	0x0000	R/W	Peripheral Bank Configuration Register 0	486
EBC0_B1CR	0x0001	R/W	Peripheral Bank Configuration Register 1	486
EBC0_B2CR	0x0002	R/W	Peripheral Bank Configuration Register 2	486
EBC0_B3CR	0x0003	R/W	Peripheral Bank Configuration Register 3	486
EBC0_B4CR	0x0004	R/W	Peripheral Bank Configuration Register 4	486
EBC0_B5CR	0x0005	R/W	Peripheral Bank Configuration Register 5	486
EBC0_B0AP	0x0010	R/W	Peripheral Bank Access Parameter 0	487
EBC0_B1AP	0x0011	R/W	Peripheral Bank Access Parameter 1	487
EBC0_B2AP	0x0012	R/W	Peripheral Bank Access Parameter 2	487
EBC0_B3AP	0x0013	R/W	Peripheral Bank Access Parameter 3	487
EBC0_B4AP	0x0014	R/W	Peripheral Bank Access Parameter 4	487
EBC0_B5AP	0x0015	R/W	Peripheral Bank Access Parameter 5	487
EBC0_BEAR	0x0020	R	Peripheral Bus Error Address Register	489
EBC0_BESR0	0x0021	R	Peripheral Bus Error Status Register 0	489
EBC0_BESR1	0x0022	R/W	Peripheral Bus Error Status Register 1	490
EBC0_CFG	0x0023	R/W	EBC Configuration Register	485

To access an indirectly accessed register, software must first write its address offset into the EBC0_CFGADDR register. The target register can then be read or written through the EBC0_CFGDATA DCR address. The following PowerPC code illustrates this procedure by writing the EBC0_B0CR register and then reading back the written value.

```
li        r3,EBC0_B0CR              ! address offset
lis       r4,<config upper>         ! upper 16-bits of configuration data
ori       r4,r4,<config lower>      ! lower 16-bits of configuration data
mtdcr     EBC0_CFGADDR,r3           ! set offset addr
mtdcr     EBC0_CFGDATA,r4           ! write config data
mfdcr     r5,EBC0_CFGDATA           ! read back config data
```

20.6.1 EBC Configuration Register (EBC0_CFG)

The contents of EBC0_CFG are accessed indirectly, using the EBC0_CFGADDR and EBC0_CFGDATA.

Figure 20-17. EBC Configuration Register (EBC0_CFG)

0	EBTC	External Bus Three-State Control 0 Address, data and control signals are high-Z between EBC transfers. 1 Between EBC transfers the peripheral data bus, address bus and control signals are driven.	Default after reset is EBTC = 1. Set EBTC = 0 to allow an external bus master to drive the peripheral bus.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	If PTD = 1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	If PTD = 0, the number of cycles from PerAddr02:31 changing until a timeout error occurs.
5:6	EMPH	External Master Priority High Allowed values: 00 01 10 11	This two-bit field contains the PLB priority that will be driven onto the PLB for a cycle from an External Master accessing a PLB slave when the External Master Priority input signal is high.
7:8	EMPL	External Master Priority Low Allowed values: 00 01 10 11	This two-bit field contains the PLB priority that will be driven onto the PLB for a cycle from an External Master accessing a PLB slave when the External Master Priority input signal is low.
9	CSTC	Chip Select Three-state Control 0 $\overline{\text{PerCS0:5}}$ are high-Z between EBC transfers. 1 $\overline{\text{PerCS0:5}}$ are always driven.	Default after reset is CSTC = 1. Set CSTC = 0 to allow an external bus master to drive the peripheral bus.
10:11	BPF	Burst Prefetch 00 Prefetch 1 double word 01 Prefetch 2 double words 10 Prefetch 4 double words 11 Reserved	Controls the amount of data prefetching when the EBC is servicing a PLB burst read. For most applications set this field to 0b00.
12:13	EMS	External Master Size 00 8-bit external master 01 16-bit external master 10 32-bit external master (not supported) 11 No external master	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0000–1111	The EBC makes a sleep request to the Clock and Power Management unit when PME = 1 and the EBC has been idle for 32 × PMT PerClk cycles.
20:31		Reserved	

20.6.2 Peripheral Bank Configuration Registers (EBC0_B0CR:EBC0_B5CR)

These registers must be configured to access memory in a bank. Boot ROM, if installed, must be attached to bank 0. If a boot ROM is installed, EBC0_B0CR[BAS] is loaded with a value of 0xFFE, and EBC0_B0CR[BS] is loaded with a value of 0b001 (2MB) immediately following SysReset inactive.

Figure 20-18. Peripheral Bank Configuration Registers (EBC0_B0CR–EBC0_B5CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be aligned to the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs when a write is attempted to a read-only bank or a read from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 32-bit bus (used only when NDFC is configured for this bank) 11 Reserved	The boot ROM must be attached to bank 0 as it is the only bank configured after a reset. Its bus width is controlled by strapping pins.
19:31		Reserved	

- **BAS (Base Address Select, bits 0:11)** – Sets the base address for a peripheral device. The bank starting address must be a multiple of the bank size programmed in the BS field. The BAS field is compared to bits 0:11 of the address. If the address is within the range of a BAS field, the associated bank is enabled for the transaction.

  Multiple bank registers may be inadvertently programmed with the same base address or as overlapping banks. An attempt to use such overlapping banks is recorded in EBC0_BESR0 or EBC0_BESR1 as a configuration error and no external access occurs. This error may result in a machine check exception if the requesting master is the CPU. If the error occurred during a DMA access, the DMA may signal an interrupt to the PPC440EP through the UIC.
- **BS (Bank Size, bits 12:14)** – Sets the number of bytes which the bank may access, beginning with the base address set in the BAS field.
- **BU (Bank Usage, bits 15:16)** – Protects banks of physical devices from read or write accesses.
- When a write access is attempted to an address within the range of the BAS field, and the bank is designated as read-only, a protection error occurs. Also, when a read access is attempted to an address within the range of the BAS field, and the bank is designated as write-only, a protection error occurs. The address of the attempted access is logged in EBC0_BEAR and type of error is logged in either EBC0_BESR0 or EBC0_BESR1.
- **BW (Bus Width, bits 17:18**) – Controls the width of region accesses. If the BW field is 0b00, the region is configured for an 8-bit data bus; 0b01 indicates a 16-bit data bus. If devices are attached to the data bus as shown in *Figure 20-2*, the EBC automatically packs read data and unpacks write data when a data transfer size mismatch exists.

20.6.3 Peripheral Bank Access Parameters (EBC0_B0AP:B5AP)

The following figure describes EBC0_BnAP register bits.

Figure 20-19. Peripheral Bank Access Parameters (EBC0_B0AP–EBC0_B5AP)

0	BME	Burst Mode Enable 0 Bursting is disabled 1 Bursting is enabled	
1:8	TWT	Transfer Wait 0–255 PerClk cycles	Wait states on all transfers when BME=0.
9:11		Reserved	
12:13	CSN	Chip Select On Timing 0–3 PerClk cycles	Number of cycles from peripheral address driven to PerCSn low.
14:15	OEN	Output Enable On Timing 0–3 PerClk cycles	Number of cycles from PerCSn low to PerOE low.
16:17	WBN	Write Byte Enable On Timing 0–3 PerClk cycles	If BEM=0, number of cycles from PerCSn low to PerWBE0:1 active.
18:19	WBF	Write Byte Enable Off Timing 0–3 PerClk cycles	If BEM=0 and RE=0, number of cycles PerWBEn becomes inactive prior to PerCSn inactive.
20:22	TH	Transfer Hold 0–7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer.
23	RE	Ready Enable 0 PerReady is disabled 1 PerReady is enabled	
24	SOR	Sample on Ready 0 Data transfer occurs one PerClk cycle after PerReady is sampled active 1 Data transfer occurs in the same PerClk cycle that PerReady becomes active	
25	BEM	Byte Enable Mode 0 PerWBE0:1 are only active for write cycles 1 PerWBE0:1 are active for read and write cycles	If BEM=0, PerWBE0:1 timing is controlled by WBN and WBF. If BEM=1, PerWBE0:1 has the same timing as PerAddr02:31.
26:31		Reserved	

- **BME (Burst Mode Enable, bit 0)** – Controls bursting for cache line fills and flushes, PLB burst transfers and all packing and unpacking operations. If BME=1, bursting is enabled. When bursting is enabled the parameters Chip Select On (CSN), Output Enable On (OEN), and First Wait (FWT) apply only to the first transfer, while Burst Wait (BWT) and Write Byte Enable On (WBN) apply during all remaining transfers of the burst.
- **TWT (Transfer Wait, bits 1:8**) – Specifies the number of wait states taken by each transfer to the bank. The number of cycles from address valid to the deassertion of PerCSn is (1 + TWT), where $0 \leq TWT \leq 255$. This field is used for non-burst transfers (field BME = 0).
- **FWT (First Wait, bits 1:5)** – Specifies the number of wait states to be taken by the first access to the bank during a burst transfer (field BME = 1). During a burst the number of cycles from the first address valid to the second address is (1 + FWT), where $0 \leq FWT \leq 31$.
- **BWT (Burst Wait, bits 6:8)** – Specifies the number of wait states to be taken by accesses beyond the first during a burst transfer (field BME = 1). On burst accesses except for the last, the number of cycles from address valid to the next valid address on each burst access is (1 + BWT), where $0 \leq BWT \leq 7$. On the last

burst access, the number of cycles from address valid to the deassertion of PerCSn is (1 + BWT), where $0 \leq BWT \leq 7$.

- **CSN (Chip Select On Timing, bits 12:13**) – Specifies the chip select turn on delay relative to the address. PerCSn may turn on coincident with the address or be delayed by 1, 2, or 3 PerClk cycles.
- **OEN (Output Enable On Timing, bits 14:15)** – Specifies when the output enable signal, PerOE, is asserted for read operations relative to the chip select signal. If 0, PerOE is asserted coincident with the chip select. If 1, 2 or 3, PerOE is delayed by 1, 2, or 3 PerClk cycles.
- **WBN (Write Byte Enable On Timing, bits 16:17**) – Specifies when the write byte enables, PerWBE0:, are asserted relative to the chip select signal. If 0, then PerWBE0: turns on coincident with the chip select. If 1, 2, or 3, PerWBE0: is delayed 1, 2, or 3 PerClk cycles from the chip select.
- **WBF (Write Byte Enable Off Timing, bits 18:19)** – Specifies when the write byte enables are deasserted, relative to the deassertion of the chip select signal. If WBF=0, PerWBE0: goes high coincident with the chip select signal. If WBF is 1, 2, or 3, PerWBE0: turns off 1, 2, or 3 PerClk cycles before the turn-off of the chip select signal.

  Note: It is an error to set WBF > BWT. Moreover, for device-paced transfers (EBC0_BnAP[RE]=1) WBF must be set to 0.
- **TH (Transfer Hold, bits 20:22)** – Specifies the number of PerClk cycles (0 through 7) that the peripheral bus is held idle after the deassertion of PerCSn. During these cycles, the address bus and data bus are active and PerR/W is valid. During the hold time, chip select, output enable, and write byte enables are inactive. If Ready Mode is used (RE=1) along with Sample on Ready (SOR=1) TH must be set to at least 1.
- **RE (Ready Enable, bit 23)** – Controls the use of the PerReady input signal. If RE=0, the PerReady input is ignored and no additional wait states are inserted into bus transactions. If RE=1, the PerReady input is examined after the wait period expires; additional wait states are inserted if the PerReady input is 0. The maximum number of wait states in each transaction is determined by the settings in the Device-Paced Timeout Disable (PTD) and Ready Timeout Counter (RTC) fields in EBC0_CFG. If EBC0_CFG[PTD] = 0, the PPC440EP waits the number of cycles indicated by EBC0_CFG[RTC] for PerReady to become active. If EBC0_CFG[PTD] = 1, the ready timeout function is disabled and the PPC440EP waits indefinitely until PerReady=1. If PerReady does not become active in the allotted time, the address of the error is logged in EBC0_BEAR and the type of error is captured in either EBC0_BESR0 or EBC0_BESR1.
- **SOR (Sample Ready, bit 24)** – Controls the location of the data transfer cycle with respect to the PerReady input. If SOR=1 the data transfer occurs on the same PerClk edge that PerReady is sampled active, whereas if SOR=0 the data transfer occurs one cycle later.
- **BEM (Byte Enable Mode, bit 25)** – Controls whether PerWBE0: is active during writes or for both reads and writes.

20.7 Error Reporting

The EBC monitors three kinds of the following errors when performing read and write transfers. Of these four, bank protect and external bus errors are always checked, while timeout error checking must be enabled using DCR-mapped configuration registers.

- **Protect Error** – Requested read or write operation violates the bank usage programmed in EBC0_BnCR[BU]. For example, write attempt to read-only bank. In all cases, no external bus activity occurs.
- **Timeout Error** – This error is possible during memory operations when both PerReady sampling is enabled, EBC0_BnAP[RE]=1, and device paced timeouts are enabled, EBC0_CFG[PTD]=0. Whenever the peripheral address bus changes the EBC begins counting PerClk cycles. If the count reaches the value represented by EBC0_CFG[RTC] a timeout error occurs. Note that timeout errors are not possible during the peripheral portion of DMA transfers.

When the EBC slave detects one of the above errors it reports the error condition to the PLB master that initiated the transfer. The EBC also logs the type of error into EBC0_BESR0 or EBC0_BESR1 and the address of the error in EBC0_BEAR.

20.7.1 Peripheral Bus Error Address Register (EBC0_BEAR)

The Peripheral Bus Error Address Register (EBC0_BEAR) is a 32-bit register containing the address of the access where a data bus error occurred. The contents of the EBC0_BEAR are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the mfdcr and mtdcr instructions.

Figure 20-20. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error (asynchronous)	

20.7.2 Peripheral Bus Error Status Register 0 (EBC0_BESR0)

The Peripheral Bus Error Status Register 0 (EBC0_BESR0) records the occurrence and type of errors for transactions attempted on behalf of the PLB4 to PLB3 bridge, MAL, PCI bridge, DMA controller, and OPB to PLB3 bridge. The contents of EBC0_BESR0 are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

Figure 20-21. Peripheral Bus Error Status Register 0 (EBC0_BESR0)

0:2	EET0	Error type for master 0 000 No error 001 Reserved 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 0 - PLB4 to PLB3 bridge
3	RWS0	Read/write status for master 0 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL0	Field lock for master 0 0 EET0 and RWS0 fields are unlocked 1 EET0 and RWS0 fields are locked	
5	AL0	EBC0_BEAR address lock for master 0 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
6:8	EET1	Error type for master 1 000 No error 001 Reserved 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 1 - Media Access Layer (MAL)
9	RWS1	Read/write status for master 1 0 Error operation was a write operation 1 Error operation was a read operation	

10	FL1	Field lock for master 1 0 EET1 and RWS1 fields are unlocked 1 EET1 and RWS1 fields are locked	
11	AL1	EBC0_BEAR address lock for master 1 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
12:14	EET2	Error type for master 2 000 No error 001 Reserved 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 2 - PCI bridge
15	RWS2	Read/write status for master 2 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL2	Field lock for master 2 0 EET2 and RWS2 fields are unlocked 1 EET2 and RWS2 fields are locked	
17	AL2	EBC0_BEAR address lock for master 2 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
12:14	EET3	Error type for master 3 000 No error 001 Reserved 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 3 - DMA controller
15	RWS3	Read/write status for master 3 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL3	Field lock for master 3 0 EET3 and RWS3 fields are unlocked 1 EET3 and RWS3 fields are locked	
17	AL3	EBC0_BEAR address lock for master 3 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
24:31		Reserved	

20.7.3 Peripheral Bus Error Status Register 1 (EBC0_BESR1)

EBC0_BESR1 records the occurrence and type of errors for transactions attempted on behalf of the OPB to PLB3 bridge. The contents of EBC0_BESR1 are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

Figure 20-22. Peripheral Bus Error Status Register 1 (EBC0_BESR1)			
0:2	EET4	Error type for master 4 000 No error 001 Reserved 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 4 - OPB to PLB3 bridge
3	RWS4	Read/write status for master 4 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL4	Field lock for master 4 0 EET4 and RWS4 fields are unlocked 1 EET4 and RWS4 fields are locked	
5	AL4	EBC0_BEAR address lock for master 4 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
6:31			Reserved

21. NAND Flash Controller

PPC440EP implements the NAND Flash controller (NDFC) as an interface between the External Bus Controller (EBC) and various NAND Flash-based storage devices.

NDFC has the following features:

- Direct interfacing to discrete NAND Flash devices (up to 4 devices), SmartMedia card socket (22-pins)
- One to four banks of NAND Flash supported.
- Supports 4MB to 256MB (32Mb to 2Gb) device sizes.
- Supports 512B + 16B or 2KB + 64B device page sizes.
- ECC generation—hamming code, single bit correction, double bit detection (SEC/DED)
- x8 wide command write, x8 wide address write, x8 and x16 bit wide data read/write
- Automatically generates RE# and WE# strobes with configurable strobe pulse width parameter
- Supports DMA in memory-to-memory copying mode to allow direct, no-processor-intervention block copy from NAND Flash out to SDRAM (after initial command/address setup performed)
- Interrupt on device becoming ready (after long page write or block erase operations).
- Boot-from-NAND: execute up to 4KB of boot code out of first block. Automatic page read accesses performed based on device configuration and read address.
- Operates from EBC peripheral clock.
- Frequency range for operation: Up to 66MHz
- Class 2 power management

21.1 Overview

The NDFC interface allows easy communication with up to four separate external NAND Flash devices. It provides both direct command, address, and data access to the external device as well as a memory mapped linear region that generates data accesses. The linear region can be used as the source or destination of a **memcpy()** command or DMA memory-to-memory transfer function to simplify and speed the software device driver. NDFC also provides hardware support for the Hamming code ECC algorithm that provides 1-bit error correction and 2-bit error detection (per 256B of data).

NDFC supports a mechanism that allows the chip to boot directly from code stored in the external NAND Flash device when the low-level boot loader follows a specific algorithm.

21.2 NDFC Signal Multiplexing

NDFC shares I/O package pins (balls) with the EBC and GPIO. In order to use NDFC, SDR0_CUST0[MEN] must be set to 0x2. This selects NDFC and causes the NDFC signals to be available on the shared balls. Additionally, SDR0_CUST0[NGC] must be set to configure each of the four external chip selects according to whether or not a NAND Flash module is installed in each of the four possible locations.

Note: Remember that because of signal multiplexing, using NDFC will reduce the functionality of the other units that share balls with NDFC.

21.3 Resetting the Controller

Software should use SDR0_SRST1[NDFC] to reset the controller.

21.4 Configuring EBC to Support the NAND Flash Controller

In order to use the NDFC, both NDFC and EBC must be properly configured. To communicate with the NDFC controller, some EBC registers must be configured differently from their power-on default settings as indicated in the following sections.

21.4.1 EBC0_BxAP

For the EBC bank or banks that are used to communicate with the NDFC, the EBC0_BxAP registers should be set to 0x018003C0.

After a power-on reset, EBC0_ PB0AP can be set to the value 0x7F8FFE80, which corresponds to maximum transfer timings for EBC bank 0 (TWT = 255, CSN = OEN = WBN = WBF = 3, TH=7). The RE bit is defaulted to 0. When EBC is configured this way, it waits 255 wait states (peripheral clock cycles) after starting a cycle and then captures whatever data is present at the end of the wait states. However, because 255 clock cycles is not usually enough wait time for an external NAND flash page read access when the NDFC is configured for AutoRead mode (see *Section 21.6.1*), the NDFC asserts an SLNDFC_GATE_EBC_CLK output that is used to gate the signal that is used to count the wait states. This forces the EBC into suspend mode where it cannot process any new requests coming from the PLB or from the External bus mastering interface. This gating should only be applied while the chip is using the NDFC to load initial boot code during power-on reset.

After the EBC core is properly configured to wait for the SLNDFC_READY assertion from the NDFC, the NDFC0_CR[EBCC] bit can be set to 1 to indicate that the SLNDFC_GATE_EBC_CLK signal is no longer needed to suspend the wait state timer in the EBC.

21.4.2 EBC0_CFG

Set EBC0_CFG[PTD] = 0b1. Paced timeout is disabled, allowing the EBC to wait indefinitely for NDFC to assert SLNDFC_READY output.

Set EBC0_CFG[RTC] = 0b111. The Ready Timeout Counter is set to the maximum of 2048 cycles. This prevents the EBC from ending normal NAND Flash cycles prematurely.

Settings of other EBC0_CFG fields do not affect how EBC communicates with NDFC.

21.5 Enabling the NAND Flash Controller

In order to use NDFC in any configuration, boot or not, the controller must be enabled by setting SDR0_CUST0[NE] to 1.

21.6 Booting from NDFC

When configured for boot, the NDFC's 4KB linear access region is the target for the PowerPC boot vector access at 0xFFFFFFFC. The stages of boot from NDFC are:

1. NDFC issues commands to NAND Flash to put device into sequential read mode.

2. The processor executes the Initial Program Loader (IPL) which loads the Secondary Program Loader (SPL) directly from the NAND Flash into the processor's Instruction Cache or other on-chip code storage memory.
3. The processor executes the SPL from cache and loads the OS Image from NAND Flash into main memory (usually DDR SDRAM).

In order to boot from the NAND flash on chip select 0, certain parameters must be properly configured. These parameters are programmed by means of bits in SDR0_CUST0, SDR0_CUST1, SDR0_CP440, and SDR0_EBC0. Coming out of reset, these registers are initialized with default values from SDR0_SDSTP1, SDR0_SDSTP2, and SDR0_SDSTP3, which are in turn initialized by the IIC Bootstrap Controller. Boot configuration C properly sets all the parameters to boot from NAND flash. Otherwise, custom boot configurations can be created using configurations G or H with a serial boot ROM.

SDR0_CUST0[MEN, NCG, NE] must be set as previously discussed.

SDR0_CUST0[NDRSC] is a 16-bit value that specifies the maximum number of clock cycles that the NDFC waits for NFRdyBusy (1 = ready) after a device reset command, before proceeding. The setting for boot configuration C is recommended.

SDR0_CUST1[NDRDC] is a 16-bit value that specifies the maximum number of clock cycles that the NDFC waits for NFRdyBusy (1 = ready) after a page read command, before proceeding. The worst case for the page read access time in many NAND flash devices is 50μs (specified in NAND flash device data sheets as tR). The value in boot configuration C is recommended.

Set SDR0_CP440[RL] = 0x2. This tells the EBC and NDFC that NDFC is the boot source.

Set SDR0_EBC0[RW] = 0x2. This tells the EBC that the width of the boot device is 32 bits. Since the external NAND Flash device is hidden from the EBC, 32 bits indicates the width of the controller and not the width of the device itself. Note that EBC0_PB0CR[BW] must stray set to 0x2 once code is running.

Some registers in the NDFC controller itself must be configured coming out of reset. By virtue of having selected NDFC as a boot device, the PPC440EP will automatically set some bits in NDFC0_B0CR and NDFC0_CR in a certain way. In particular, the timing values CRW, RWP, RWH, and RR are set to their maximum. See descriptions of those registers for more details.

Other bits in NDFC0_B0CR and NDFC0_CR must be set according to system architecture and/or choice of device before the first access to the NAND flash can take place. Coming out of reset, these bits are automatically set by NDFC from the corresponding bits in SDR0_CUST0, which in turn is initialized with settings provided by SDR0_SDSTP2. *Table 21-1* summarizes the correspondence between SDR0_CUST0 fields and NDFC0_B0CR and NDFC0_CR fields.

Table 21-1. Correspondence between SDR0_CUST0 Fields and NDFC Register Fields

SDR0_CUST0[NE]	NDFC0_B0CR[EN]
SDR0_CUST0[NBW]	NDFC0_B0CR[SZ]
SDR0_CUST0[NBP]	NDFC0_CR[RPG]
SDR0_CUST0[NBAC]	NDFC0_CR[ABAC]
SDR0_CUST0[NARE]	NDFC0_CR[ARE]
SDR0_CUST0[NRB]	NDFC0_CR[REN]

21.6.1 AutoRead Mode

The strap input set in SDR0_CUST0[NARE] is used to load the initial value for NDFC0_CCR[ARE]. If this bit is set, then the NDFC only issues a reset command to the NAND Flash device and then deasserts SDR register SDR0_CUST0 to allow the rest of the system to come out of reset.

In the AutoRead mode (NDFC0_CCR[ARE]=1), the NDFC will automatically generate the page read command and page read address based on the read cycle address that is driven by the EBC. If the EBC read cycle's address is exactly +4 from the previous read cycle then the NDFC will generally not issue a new read command to the NAND Flash device, but rather will just read and return the next sequential piece of data. The exception to this '+4' rule is when an access would cross the NAND Flash device's data page boundary. To prevent reading data in the device's spare area and returning this as valid data, the NDFC will detect when an access crosses the device's data page size boundary and will also treat this access as non-sequential.

This sequential read address detection allows the NDFC to avoid unnecessary read commands, and their corresponding page read access delay (tR), for code that is being fetched sequentially from memory. EBC write cycles to the NDFC will not affect the sequential read address detection, so for example, the NDFC configuration registers can be written to update the configuration of NDFC0_CR to speed up external NAND Flash access timings. Also read cycles that are not inside the linear access region will not affect the sequential read address detection, so reads of the NDFC configuration or status registers are possible as well.

21.7 ECC

The NDFC implements a Hamming Code Error Correction Code (ECC) that is directly compatible with SmartMedia format. This ECC is based on parity calculations of the read (incoming) or write (outgoing) data and consists of 3 Bytes (22-bits) per 256 Byte (2048-bits) data segment and provides single-bit error correction, double-bit error detection (SEC/DED) on the 256 Byte block. The 22-bit ECC code is composed of 16-bit line parity and 6 bit column parity.

- Hamming ECC Algorithm is directly compatible with SmartMedia specification.
- ECC accumulated during page write and would be available to software for programming after either 256 or 512 Bytes of page data write cycles are completed.
- ECC calculated during page read on incoming data and is available for comparison with ECC stored in device.
- Eight ECC registers accumulate calculated ECC data for each 256 Byte block in an entire 2KB page-sized device.
- ECC hardware assist is mainly useful when reading entire pages.
- Software is responsible for making the stored versus calculated ECC comparison and performing any required bit error correction to the data block.
- The hardware ECC may be supplemented with stronger software ECC algorithm to improve coverage for Multi-Level-Cell (MLC) NAND Flash devices, where the failure of a single transistor can cause the loss of 2-bits of data.

For the SmartMedia format, the ECC code stored in the external NAND Flash device is always stored in the format shown in *Table 21-2*.

Table 21-2. ECC Code Assignment Table

I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
P64	P64'	**P32**	P32'	**P16**	P16'	**P8**	P8'
P1024	P1024'	**P512**	P512'	**P256**	P256'	**P128**	P128'
P4	P4'	**P2**	P2'	**P1**	P1'	1	1
P8 to P1024 = Line/Row Parity P1 to P4 = Column Parity							

To ease the comparison of stored ECC with the NDFC's calculated ECC, the calculated result stored in NDFC0_ECC0-NDFC0_ECC7 is in the same SmartMedia format shown in *Table 21-2*.

21.8 NDFC Registers

The NAND Flash controller provides one dedicated configuration register for each NAND Flash device selected. There are also 15 other common registers that are shared across all devices and are used to configure or monitor status of the common aspects of the NDFC.

All of the NDFC registers are memory mapped.

All NDFC registers can be written with a transfer size of byte, half word or full word, based on the value of the EBC_WBE_N(0:3). The table below describes the valid transfer types. A byte size transfer is performed if only one bit of EBC_WBE_N(0:3) is asserted (active low). A half word size transfer is performed if either EBC_WBE_N(0:1) or EBC_WBE_N(2:3) are asserted while the other bits are not asserted. A full word transfer is performed when EBC_WBE_N(0:3) are all asserted, or when a read cycle is detected (EBC_OE_N asserted) and none of the EBC_WBE_N(0:3) are asserted.

Table 21-3. EBC Transfer Sizes

EBC_WBE_N(0:3)	EBC_ABUS(30:31)	EBC_RNW	EBC_OE_N	Transfer Type	Transfer Size	Valid EBC_DBUS bytes
0000	00	1	0	Read	Full word (4 bytes)	(0:31)
		0	1	Write		
1111	00	1	0	Read	Full word (4 bytes)	(0:31)
0011	00	1	0	Read	Half word (2 bytes)	(0:15)
		0	1	Write		
1100	10	1	0	Read	Half word (2 bytes)	(16:31)
		0	1	Write		
0111	00	1	0	Read	Byte (1 byte)	(0:7)
		0	1	Write		
1011	01	1	0	Read	Byte (1 byte)	(8:15)
		0	1	Write		
1101	10	1	0	Read	Byte (1 byte)	(16:23)
		0	1	Write		

Table 21-3. EBC Transfer Sizes (continued)

EBC_WBE_N(0:3)	EBC_ABUS(30:31)	EBC_RNW	EBC_OE_N	Transfer Type	Transfer Size	Valid EBC_DBUS bytes
1110	11	1	0	Read	Byte (1 byte)	(24:31)
		0	1	Write		
All other combinations are considered invalid and the NDFC will return SLNDFC_BUS_ERROR if they are attempted.						

The EBC_WBE_N(0:3) byte enables are used to gate the writes to NDFC registers. For the NDFC0_CMD and NDFC0_ADDR register, since the only valid location is at byte 0, the EBC_WBE_N(0) must be active for writes to these registers in order for an external hardware cycle to be generated.

21.8.1 Memory Map

This section lists all of the NDFC registers and describes how they are accessed.

The NDFC is relocatable within the EBC address space. See *External Bus Controller* on page 465 for details on how to program the base address and region size and assign the NDFC to one of the EBC banks. Addresses in the following table are offsets from the programmed base address.

All NDFC configuration, control and status registers are typically accessed on word (32-b/4-B) boundaries to simplify software design. All NDFC registers and the NDFC Linear Data Access region can be accessed as individual bytes if the appropriate byte enables are asserted by the EBC for both reads and writes.

Table 21-4. NAND Flash Controller Memory Map

Mnemonic	Register	Address Offset	Access	Page
NDFC0_CMD	NDFC Command Register	0x0000	R/W	499
NDFC0_ADDR	NDFC Address Register	0x0004	R/W	499
NDFC0_DATA	NDFC Data Register	0x0008	R/W	500
NDFC0_ECC0	NDFC ECC Register 0	0x0010	R	501
NDFC0_ECC1	NDFC ECC Register 1	0x0014	R	501
NDFC0_ECC2	NDFC ECC Register 2	0x0018	R	501
NDFC0_ECC3	NDFC ECC Register 3	0x001C	R	501
NDFC0_ECC4	NDFC ECC Register 4	0x0020	R	501
NDFC0_ECC5	NDFC ECC Register 5	0x0024	R	501
NDFC0_ECC6	NDFC ECC Register 6	0x0028	R	501
NDFC0_ECC7	NDFC ECC Register 7	0x002C	R	501
NDFC0_B0CR	NDFC Bank Configuration Register 0	0x0030	R/W	502
NDFC0_B1CR	NDFC Bank Configuration Register 1	0x0034	R/W	502
NDFC0_B2CR	NDFC Bank Configuration Register 2	0x0038	R/W	502
NDFC0_B3CR	NDFC Bank Configuration Register 3	0x003C	R/W	502
NDFC0_CR	NDFC Configuration Register	0x0040	R/W	503
NDFC0_SR	NDFC Status Register	0x0044	R	505

Table 21-4. NAND Flash Controller Memory Map (continued)

Mnemonic	Register	Address Offset	Access	Page
NDFC0_HWCTL	NDFC Direct Hardware Control Register	0x0048	R/W	505
NDFC0_REVID	NDFC Revision ID Register	0x0050	R	506
na	Linear Data Access Region (when NDFC0_CR[4] = 0)	0x1000 - 0x1FFF	R/W	na

21.8.2 NDFC Address Register (NDFC0_ADDR)

The NAND Flash Address register (NDFC0_ADDR) is an external hardware access register. When a write is made to this register, the ALE signal is asserted and a write cycle is performed to the external device.

External device cycles are only performed when the currently selected bank, as programmed in the NDFC0_CR[BS] field, is enabled in its bank configuration register, NDFC0_CR[EN]. When the currently selected bank is not enabled, writes and reads to this register simply store and return stored data, respectively.

This register can be written with a transfer size of byte, half word or full word, but in all cases, only the data on EBC_DBUS(0:7) is used and output to the external device and only when EBC_WBE_N(0) is asserted ('0') and the address indicates that the byte 0 was accessed (EBC_ABUS(30:31)==0b00).

Reads of this register only return the data stored in the NDFC0_ADDR register. No external device cycle is performed for a read of this register. Unused bits return a value of '0'

Figure 21-1. NAND Flash Address Register (NDFC0_ADDR)

0:7	ADDR	Nand Flash Command	
8:31		Reserved	

21.8.3 NDFC Command Register (NDFC0_CMD)

The NAND Flash Command register (NDFC0_CMD) is an external hardware access register. When a write is made to this register, the CLE signal is asserted and a write cycle is performed to the external device.

External device cycles are only performed when the currently selected bank (as programmed in the NDFC0_CR[BS] field) is enabled in its bank configuration register, NDFC0_BnCR[EN]. When the currently selected bank is not enabled, writes and reads to this register simply store and return stored data, respectively.

This register can be written with a transfer size of byte, half word or full word, but in all cases, only the data on EBC_DBUS(0:7) is used and output to the external device and only when EBC_WBE_N(0) is asserted (0) and the address indicates that the byte 0 was accessed (EBC_ABUS(30:31)==0b00).

Reads of this register only return the data stored in the NDFC0_CMD register. No external device cycle is performed for a read of this register. Unused bits return a value of 0.

Figure 21-2. NAND Flash Command Register (NDFC0_CMD)

0:7	CMD	Nand Flash Command	
8:31		Reserved	

21.8.4 NDFC Data Register (NDFC0_DATA)

The NAND Flash Data register (NDFC0_DATA) is an external hardware access register. When a write is made to this register, one or more write cycles are performed to the external device. When a read is made to this register, one or more read cycles are performed to the external device.

External device cycles are only performed when the currently selected bank, as programmed in the NDFC0_CR[BS] field, is enabled in its bank configuration register, NDFC0_CR[EN]. When the currently selected bank is not enabled, writes and reads to this register simply store and return stored data, respectively.

This register can be written with a transfer size of byte, half word or full word, based on the value of the EBC_WBE_N(0:3).

Note that a byte size read transfer to a 16-bit NAND Flash device (NDFC0_CR[SZ]=1) will result in the loss of the upper half of the data returned from the device. This condition is detected and causes the NDFC0_SR[RDL] bit to be set.

The total number of external accesses performed for each transfer size is dependent on the selected bank's device width.

Figure 21-3. NAND Flash Data Register (NDFC0_DATA)

0:31	DATA	Nand Flash Data	

Table 21-5 and *Table 21-6* describe the data input and out ordering, respectively.

Table 21-5. Data Input Ordering to Read Data on EBC

Device Size	Read size[a]	External Cycle #	Data Input 0:7	Data Input 8:15	EBC_DBUS(0:31)[b]
16	FW (4)	1 2	<aa> <cc>	<bb> <dd>	<**aabbccdd**>
	HW (2)	1	<aa>	<bb>	<**aabb**aabb>
	BY (1)[c]	1	<aa>	<bb>	<**aa**aaaaaa>
8	FW (4)	1 2 3 4	<aa> <bb> <cc> <dd>	<XX> <XX> <XX> <XX>	<**aabbccdd**>
	HW (2)	1 2	<aa> <bb>	<XX> <XX>	<**aabb**aabb>
	BY (1)	1	<aa>	<XX>	<**aa**aaaaaa>

a. For transfer size descriptions, refer to Table 21-3 on page 497
b. Reads of less than a full word will return data mirrored onto all possible byte lanes.
c. A byte size read from a 16-bit device results in loss of data returned from the device. This will set the NDFC0_STAT[RDL] flag.

Table 21-6. Write Data to Output Ordering

Device Size	EBC_DBUS(0:31)	Write size [1]	External Cycle #	Data Output 0:7	Data Output 8:15
16	<aabbccdd>	FW (4)	1 2	<aa> <cc>	<bb> <dd>
	<aabb>	HW (2)	1	<aa>	<bb>
	<aa>	BY (1)	1	<aa>	<XX>
8	<aabbccdd>	FW (4)	1 2 3 4	<aa> <bb> <cc> <dd>	<ZZ> <ZZ> <ZZ> <ZZ>
	<aabb>	HW (2)	1 2	<aa> <bb>	<ZZ> <ZZ>
	<aa>	BY (1)	1	<aa>	<ZZ>
Note 1: For transfer size descriptions, refer to Table 21-3 on page 497					

21.8.5 NAND Flash ECC Calculation Registers (NDFC0_ECC0:NDFC0_ECC7)

The NAND Flash ECC Calculation Registers (NDFC0_ECC0:NDFC0_ECC7) are for reading the hardware generated ECC values for each 256-Bytes of page read or page write data.

These registers cannot be written. They can be cleared using the NDFC0_CR[ECR] (ECC Reset) bit. Reads of this register only return the data stored in the register.

Figure 21-4. NAND Flash ECC Calculation Register (NDFC0_ECC0:NDFC0_ECC7)

0:7	BC	Byte Counter	
8:15	LP0	Line/Row Byte 0 ECC Calculated Result 8 - P64 9 - P64 10 - P32 11 - P32 12 - P16 13 - P16 14 - P8 15 - P8	
16:23	LP1	Line/Row Byte 1 ECC Calculated Result 16 - P1024 17 - P1024 18 - P512 19 - P512 20 - P256 21 - P256 22 - P128 23 - P128	

24;31	CP	Column (bit offset) Calculated Result 24 - P4 25 - P4 26 - P2 27 - P2 28 - P1 29 - P1 30 - 1 31 - 1	

21.8.6 NDFC Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR)

The NAND Flash Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR) are for enabling and setting access parameters for each of the external banks of NAND Flash.

This register can be written with a transfer size of byte, half word or full word. Reads of this register only return the data stored in the register. Unused bits return a value of 0.

Figure 21-5. NAND Flash Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR)

0	EN	Bank Enable 0 Bank access disabled 1 Bank access enabled	When the NDFC0_ADDR, NDFC0_CMD, NDFC0_DATA, or Linear Data regions are accessed and the currently selected bank is disabled, the NDFC will not perform an external bus cycle, but rather just store the data into the appropriate register. NDFC0_CR reset default comes from PGM_BOOTNAND_EN.
1	CED	CE# style 0 CE# will remain asserted/active until ECC Reset or switching bank selects via NDCF0_CR 1 CE# is 'Don't Care' style and will deassert when external access cycle is completed	When cleared, the selected bank's CE# signal will be asserted when the first access is made to any of the NDFC0_CMD, NDFC0_ADDR, NDFC0_DATA registers or the Linear Data regions and will remain asserted until a CE# reset is performed (NDFC0_CR[CER]) or the active bank is switched (NDFC0_CR[BS]). NDFC0_CR reset default is always 0.
2:3		Reserved	
4	SZ	Bank Size 0 External Device is 8-bit, attached to NDFC0_DATA[0:7] 1 External Device is 16-bit, attached to NDFC0_DATA[0:15]	If PGM_BOOTNAND_EN=1, NDFC0_CR reset default comes from PGM_BOOTNAND_WIDTH.
5:16		Reserved	
17:19	CRW	CE# low to RE#/WE# starting (0-7)	Setting is based on EMI_CLK core clock cycles If PGM_BOOTNAND_EN=1, NDFC0_CR reset default=0b111.
20		Reserved	
21:23	RWP	RE#/WE# low pulse time (1-8)	Setting is based on EMI_CLK core clock cycles If PGM_BOOTNAND_EN=1, NDFC0_CR reset default=0b111.
24		Reserved	
25:27	RWH	RE#/WE# high pulse time (1-8)	Setting is based on EMI_CLK core clock cycles If PGM_BOOTNAND_EN=1, NDFC0_CR reset default=0b111.
28		Reserved	

29:31	RR	RDY_BSY# high to RE# starting (1-8)	Setting is based on EMI_CLK core clock cycles If PGM_BOOTNAND_EN=1, NDFC0_CR reset default=0b111. Note: This value only has an effect for read accesses in the linear region and only when NDFC0_CR[REN] is set.

21.8.7 NDFC Configuration Register (NDFC0_CR)

The NAND Flash Configuration Register (NDFC0_CR) is for enabling and setting access parameters for the entire NDFC.

This register can be written with a transfer size of byte, half word or full word. Reads of this register only return the data stored in the register.

Figure 21-6. NAND Flash Configuration Register (NDFC0_CR)

0	CER	CE# Reset 0 no action 1 Reset any active Chip Enable output	Setting this bit to 1 will deassert an active CE# output. This is only necessary when a bank is configured with NDFC0_CR[CER]='0' and the CE# signal would otherwise stay asserted.
1	ERC	ECC Reset 0 no action 1 Reset ECC	Setting this bit to 1 will reset the ECC counter(s)/calculation(s). Note: Writes to this bit are not actually stored in the register, but rather only serve to perform the reset function.
2	RIE	Interrupt Enable on Device Ready 0 No Interrupt output 1 Interrupt pulse generated when RDY/BSY# goes from 0 -> 1, indicating device has just completed some internal operation.	The Interrupt pulse will be generated on the NDFC_IRQ core output and will be 4 EMI_CLKs wide. Typically this means that the interrupt controller must detect this signal with an edge-sensitive input.
3	REN	Enable wait for Ready in reads from Linear Region 0 NDFC ignores value of I_EXT_NDFC_RDYBSY when accessing the linear data region 1 NDFC waits for I_EXT_NDFC_RDYBSY == 1 (ready) before allowing read cycles to occur in the linear data region.	Setting this bit will also allow the NDFC0_CR_L2[`CR_RR] timer to be activated when a read is performed in the linear region and the current, latched state of I_EXT_NDFC_RDYBSY == 0 (busy). The CR_RR counter will count when I_EXT_NDFC_RDYBSY goes from 0 (busy) to 1 (ready) before allowing a read cycle to the external device. Clearing this bit means that software must have insured that the device is ready prior to accessing the external device. Clearing this bit is required if the I_EXT_NDFC_RDYBSY signal does not come from chip i/o pad (due to pin multiplexing) or if the external NAND flash device does not provide a ready/busy# indicator output. This bit can be strapped at reset time via PGM_RDYBSY_EN. Note: This setting also affects AutoRead mode because accesses at reset time for AutoRead mode are sent to the linear access region.
4	ROMEN	Enable ROM in Linear Region 0 Normal Linear Region behavior -- access ext. NAND flash for reads from 0x1000-0x1FFF 1 Return ROM data for reads from 0x1000-0x1FFF	When set, the ROM address decode resides in the Linear region that is normally use to access data on the external NAND Flash device. This bit's reset default is 1 when PGM_BOOTNAND_EN is 0 and vice-versa.

5	ARE	Auto-Read Enable 0 Reads to Linear Region only generate read cycle to the external NAND Flash. 1 Reads to Linear Region can generate automatic page read commands with the address determined from the EBC read address and then perform a read cycle to the external NAND flash.	When set, the address of reads to the Linear region will be compared to the previous read address and if the new address is 'non-sequential' (i.e., it is not <prev_addr>+4), then an automatic 'page read' command will be performed. The address of the page read command will be determined from the new address following the rules outlined in *AutoRead Mode* on page 496.
6:7	BS	Bank Select 00 Select Bank on CE[0] 01 Select Bank on CE[1] 10 Select Bank on CE[2] 11 Select Bank on CE[3]	Changing the Setting from 0 to 3; setting determines which bank is 'selected' for subsequent hardware accesses. Changing the Bank Select setting will also deassert any active chip enable (the same as writing NDFC0_CR[CER]=1).
8:17		Reserved	
18:19	ARAC	Auto-Read mode Address Cycles 00 3 Addr. Cycles, 1 Col. + 2 Row (512 page size) 01 4 Addr. Cycles, 1 Col. + 3 Row (512 page size) 10 4 Addr. Cycles, 2 Col. + 2 Row (2k page size) 11 5 Addr. Cycles, 2 Col. + 3 Row (2k page size)	Used for the AutoRead mode logic that is used when booting from NAND Flash. This value determines how many address cycles are performed during the setup to configure the external device for reading the IPL from bank 0. This value is preloaded from the PGM_BOOTNAND_ADDR_CYCLES(0:1) strapping input during reset. See *AutoRead Mode* on page 496.
20:23	RPG	Auto-Read mode base Page value	Used for the AutoRead mode logic that is used when booting from NAND Flash. This value determines the base page address that an automatically generated page read command will output to the NAND Flash during AutoRead. This value is preloaded from the PGM_BOOTNAND_PAGE(0:3) strapping input during reset. If this value is changed from it's strapped, default value, the software must take care to insure that the next AutoRead mode access made to the NDFC linear region is non-sequential so that the new page address value will be sent to the external NAND Flash device. This is because writing to the NDFC0_CR register by itself will not cause the NDFC to assume that the ARPG value has changed. See *AutoRead Mode* on page 496.
24:28		Reserved	
29	EBCC	EBC Configuration Completed 0 EBC Core configuration not completed yet, continue to drive SLNDFC_GATE_EBC_CLK when necessary during AutoRead mode 1 EBC Core configuration completed, the SLNDFC_GATE_EBC_CLK signal will never be asserted, even during AutoRead mode.	Once the EBC core is properly configured to wait for the READY signal for the bank(s) that are used to communicate to the NDFC, then this configuration bit can be set to prevent further gating of the EBC's SAMPLE_CYCLE signal.
30	DHC	Direct Hardware Control Enable 0 Normal 1 Use NDFC0_HWCTL register to directly control and read external interface inputs and outputs	This bit enables the NDFC0_HWCTL register to control the NDFC external interface directly. The automatic control from the NDFC0_CMD, NDFC0_ADDR, NDFC0_DATA registers and the linear address region(s) are disabled when DHC is set. In addition, setting DHC will cause the O_NDFC_EXT_CTL_EN & O_NDFC_EXT_CE_EN enable outputs to go active (=1). The O_NDFC_EXT_DBUS_EN[0:1] outputs are controlled directly by the NDFC0_HWCTL[DENL] and NFHWCTL[DENH] registers when DHC is set. This should be used for debug purposes only.
31		Reserved	

21.8.8 NDFC Status Register (NDFC0_SR)

The NAND Flash Status Register (NDFC0_SR) is for monitoring the external Ready/Busy# indicator input that comes from the NAND Flash devices, indicating the currently active ECC accumulation register, and a status bit to indicate that read data loss may have occurred in accesses to an external NAND Flash device.

This register cannot be written. Reads of this register only return the data stored in the register. Unused bits return a value of '0'.

Figure 21-7. NAND Flash Status Register (NDFC0_SR)

0		Reserved	
1:3	ECCP	Active NDFC0_ECC register identifier 0-7 = NDFC0_ECC0:NDFC0_ECC7	This field is reset to '000' when the NDFC0_CR[ERC] bit is written with a '1' to generate an ECC Reset. This field is automatically incremented after every 256-Bytes of data are transferred to the external interface. Once ECCP reaches '111' (NDFC0_ECC7) and 256 bytes of data are transferred, the field will not be incremented any further. This allows a full 2kB page to be transferred followed by its 64B spare data area without the spare data affecting the ECC value stored in any of the NDFC0_ECC registers.
4:5		Reserved	
6	RDL	Read Data Loss 0 No Read Data Loss has been detected 1 Possible Read data has been lost	This bit is only set when a EBC single byte-wide read access is performed to an external NAND Flash bank that is configured to be 16-bit wide, indicating that the upper 8-bits of read data from that device access were not returned. This may occur if byte read cycles are performed by the EBC to a 16-bit NAND flash bank. This bit is only cleared when an CE# Reset via NDFC0_CR[CER] is performed. Note: This status bit is not cleared by changing the NDFC0_CR[BS] bank select.
7	RDY	Ready/Busy# Input indicator 0 External RDY/BSY# input is low -> Busy 1 External RDY/BSY# input is high -> Ready	
8:31		Reserved	

21.8.9 NAND Flash Direct Hardware Control Register (NDFC0_HWCTL)

The NAND Flash Direct Hardware Control Register (NDFC0_HWCTL) enables direct control of the external interface inputs and outputs that go to the NAND Flash devices. The settings in this register have no control over the external interface unless the NDFC0_CR[DHC] bit is set.

When NDFC0_CR[DHC] is set, the O_NDFC_EXT_CTL_EN and O_NDFC_EXT_CE_EN enable outputs go active (=1) immediately, but the O_NDFC_EXT_DBUS_EN[0:1] outputs are controlled directly by the NDFC0_HWCTL[DENL] and NDFC0_HWCTL[DENH] registers.

This register is Read/Write with several conditions.

1. The contents of this register are not writable and reads will not return data in the DATAL and DATAH fields unless the NDFC0_CR[DHC] bit is already set.
2. The DATAL and DATAH fields are read-only when the DENL and DENH fields respectively are cleared (=0).

Unused bits return a value of 0

Figure 21-8. NAND Flash Direct Hardware Control Register (NDFC0_HWCTL)			
0:3	CE0-CE3	Chip Enable (O_NDFC_EXT_CE_N[0:3])	This 4-bit field contains bits CE0:CE3. The setting of these bits is inverted before going to the external output, so a setting of '1' here gives an active output (low) to the NAND Flash device's CE# input.
4	CLE	O_NDFC_EXT_CLE direct control 0 O_NDFC_EXT_CLE = 0 1 O_NDFC_EXT_CLE = 1	The value of this bit directly correlates to the external output signal
5	ALE	O_NDFC_EXT_ALE direct control 0 O_NDFC_EXT_ALE = 0 1 O_NDFC_EXT_ALE = 1	The value of this bit directly correlates to the external output signal
6	RE	O_NDFC_EXT_RE_N direct control (inverted) 0 O_NDFC_EXT_RE_N = 1 1 O_NDFC_EXT_RE_N = 0	The value of this bit is inverted before driving the external output signal.
7	WE	O_NDFC_EXT_WE_N direct control (inverted) 0 O_NDFC_EXT_WE_N = 1 1 O_NDFC_EXT_WE_N = 0	The value of this bit is inverted before driving the external output signal.
8	DENL	O_NDFC_EXT_DBUS_EN[0] direct control 0 O_NDFC_EXT_DBUS_EN[0] = 0 (Read I_EXT_NDFC_DATA[0:7]) 1 O_NDFC_EXT_DBUS_EN[0] = 1 (Write O_NDFC_EXT_DATA[0:7])	Data Direction (0=Read, 1=Write) for 8 bits of data on NDFC0_DATA[0:7]
9	DENH	O_NDFC_EXT_DBUS_EN[1] direct control 0 O_NDFC_EXT_DBUS_EN[1] = 0 (Read I_EXT_NDFC_DATA[8:15]) 1 O_NDFC_EXT_DBUS_EN[1] = 1 (Write O_NDFC_EXT_DATA[8:15])	Data Direction (0=Read, 1=Write) for 8 bits of data on NDFC0_DATA[8:15]
10:15		Reserved	
16:23	DATAL	O_NDFC_EXT_DATA[0:7] direct control (DENL = 1) Read/Write I_EXT_NDFC_DATA[0:7] latched input value (DENL = 0) Read-only	When DENL is 0, this field returns the value on the I_EXT_NDFC_DATA[0:7] inputs. When DENL is 1, this field drives its value out to the O_NDFC_EXT_DATA[0:7] outputs.
24:31	DATAH	O_NDFC_EXT_DATA[8:15] direct control (DENH=1) Read/Write I_EXT_NDFC_DATA[8:15] latched input value (DENH = 0) Read-only	When DENH is 0, this field returns the value on the I_EXT_NDFC_DATA[8:15] inputs. When DENH is 1, this field drives its value out to the O_NDFC_EXT_DATA[8:15] outputs.

21.8.10 NDFC Revision ID Register (NDFC0_REVID)

NDFC0_REVID is a read-only register that allows software to read the version information.

Figure 21-9. NAND Flash Revision ID Register (NDFC0_REVID)			
0:11		Reserved	
12:23	RN	Revision Number	Corresponds to the RCS major revision number of the source RTL
24:31	BRN	Branch Revision Number	Corresponds to the RCS minor (branch) revision number of the source RTL

21.8.11 Linear Data Access Region

The linear data access regions are not registers at all, but rather is a contiguous region of the NDFC memory map that, when accessed, causes a NAND Flash device Read or Write cycle to be performed.

Each access that is performed to one of these regions is handled the same as if it were performed to the NDFC0_DATA register. Since NAND Flash is a sequential access type device, keep in mind that every access to this region will just return the next sequential piece of data from the NAND Flash device, regardless of the address within the access.

One advantage of the linear access region is that the CPU or DMA can perform multi-word burst or line transfers to this region with an incrementing address on each transfer. The EBC will convert these burst or line type transfers into individual full word transfers to the NDFC and the NDFC will then make the appropriate access to the external NAND Flash device.

22. Direct Memory Access Controllers

PPC440EP implements two direct memory access controllers: one attached to 128-bit processor local bus (DMA2P40) and the other attached to 64-bit processor local bus (DMA2P30). Because of functional differences this chapter is organized under two separate sections as follows:

- *DMA to PLB4 Controller (DMA2P40)*
- *DMA to PLB3 Controller (DMA2P30)*

The Direct Memory Access (DMA) controller is a Processor Local Bus (PLB) and On-chip Peripheral Bus (OPB) master which supports the autonomous transfer of data between memory and peripherals and from memory-to-memory. The controller provides four DMA channels, each of which has an independent set of configuration registers. Each channel has its own control, count and control, source address, destination address, and scatter/gather address registers. Once these registers are programmed by the PPC440EP processor, the DMA controller performs the requested data transfer without the need for processor intervention.

The four DMA channels also support scatter/gather transfers. During a scatter/gather transfer, the configuration registers for a particular DMA channel are automatically loaded from a data structure in memory instead of being individually programmed. Since the scatter/gather address register is updated in this process, the channel can optionally reconfigure itself for another transfer when the current one completes.

As PLB and OPB master, the DMA can read and write any address accessible by the PPC440EP processor. This includes memory and memory-mapped peripherals on the external bus controller (EBC) interface and SDRAM memory and PCI addresses that have been mapped into PLB address space. The DMA controller can also service DMA peripherals attached to the EBC via the DMAReqn, DMAAckn and EOTn[TCn] I/Os, along with the OPB-attached UART0 and UART1.

22.1 DMA to PLB4 Controller (DMA2P40)

DMA to PLB4 is attached to the 128-bit processor local bus. Neither sub-channels nor external sources are used by this controller.

Table 22-1 lists the DMA to PLB4 channel assignments in the PPC440EP.

Table 22-1. DMA to PLB 4 Channel Assignments

Channel	DMA Internal Source
Channel 0	USB2.0 Device Endpoint 1 In
Channel 1	USB2.0 Device Endpoint 2 In
Channel 2	USB2.0 Device Endpoint 1 Out
Channel 3	USB2.0 Device Endpoint 2 Out

22.1.1 DMA Transfers

As a specialized controller, the DMA provides system designers and programmers with a highly efficient method of moving data. During any DMA transfer the controller always buffers data read from the source prior to writing the data to the destination. Since many buses, including the internal PLB, provide substantially better performance

when bursting data, the DMA controller includes a 128-byte (8 quad word) buffer. This buffer is enabled on a per-channel basis by setting DMA2P40_CRn[BEN] and serves to minimize the number of discrete memory transactions. Each of the four DMA channels is configurable for either peripheral or memory-to-memory transfers.

22.1.1.1 Memory-to-Memory Transfers

The DMA controller can perform either device-paced (hardware-initiated) or software-initiated memory-to-memory transfers

Device-Paced Memory Mode Transfers (Hardware Initiated)

Device-paced memory (DPM) corresponds to the USB 2.0 device channels listed in *Table 22-1*. Memory is any address accessible from the PLB or OPB, including PLB-mapped PCI address space, SDRAM memory, and memory connected to the external bus.

During a DPM mode transfer, the DPM device requests a DMA transfer by asserting a DMA request line. For internal DPMs, this signal is internal to the chip. When the requesting channel has the highest priority of any active channel, the DMA core executes a DPM transfer.

There are two types of DPM transfers: DPM-to-memory and memory-to-DPM. A DPM-to-memory transfer reads data from a DMA DPM device, while a memory-to-DPM transfer writes data. In both cases, the DPM interface transfers data at the programmed width of the peripheral (DMA2P40_CRn[PW]). When the DMA buffer is disabled for the active channel (DMA2P40_CRn[BEN]=0), each DPM transfer causes a corresponding memory operation. When buffering is enabled during a DPM-to-memory transfer, data is collected until the 128-byte buffer is full, the DPM deasserts DMAReqn, or the channel completes. The buffer contents are then written to the target memory as efficiently as possible.

If the Destination address is located on PLB space, the data is written to memory with one burst operation if the destination address is quad word aligned and the buffer is full; or, it is written with a combination of single beat and burst transfers, depending on the destination address alignment and the amount of data stored in the DMA buffer. If the destination address is located on OPB memory space, the DMA controller writes out the content of the DMA buffer with a series of single beat or burst transfers, depending on whether bursting is enabled.

Memory-to-DPM transfers differ since the amount of data that will be requested by the DPM is unknown. If the DMA buffer is disabled (DMA2P40_CRn[BEN]=0), a discrete source memory read occurs for each element in the DMA transfer. Since this is inefficient, the buffer should only be disabled for low data rate transfers. When the 128-byte buffer is enabled, the controller uses the setting in DMA2P40_CRn[PF] to prefetch 1, 2, 4, or 8 128-bit quad words from the source memory. The DMA controller provides data from the buffer until the DPM deasserts its request or the transfer completes. Whenever any of these conditions occurs, any unused data in the DMA buffer is discarded.

For both DPM-to-memory and memory-to-DPM transfers, the DMA controller supports fixed length bursts of 2, 4, 8, or 16 data elements on the peripheral side of the transfer. When bursting is enabled (DMA2P40_CTn[BTEN]=1), an active DMA request (DMAReqn) causes the DMA controller to transfer the programmed number of data items (DMA2P40_CTn[BSIZ]) in one atomic operation.

Software-Initiated Memory-to-Memory Transfers

Software-initiated memory-to-memory transfers between memories with fixed timings provide the best overall performance. During a software-initiated transfer, the DMA controller knows the exact amount of data to be transferred. As a result, when the 128-byte DMA buffer is enabled (DMA2P40_CRn[BEN]=1) the controller uses bursts as much as possible if the memory is located on PLB space. For memory located on OPB space, bursting has to be enabled (DMA2P40_CTn[BTEN]=1) for the DMA controller to burst. To ensure the highest bandwidth with memory locate on PLB space, source and destination addresses should be aligned on 128-byte boundaries.

There are two special cases of software-initiated memory-to-memory transfers.

- Source address increment bit is reset (DMA2P40_CRn[SAI]=0).

  In this special case, the source memory device responds to a unique address. If the source memory device (FIFO-like device) is located on PLB space, bursting is not possible. If the source memory device is located on OPB space, bursting is possible if enabled (DMA2P40_CTn[BTEN]=1).

- Destination address increment bit is reset (DMA2P40_CRn[DAI]=0).

  In this case, the destination memory device responds to a unique address. If the destination memory device (FIFO-like device) is located on PLB space, bursting is not possible. If the destination memory device is located on OPB space, bursting is possible if enabled (DMA2P40_CTn[BTEN]=1).

22.1.1.2 Scatter/Gather Transfers

Each of the four DMA channels supports scatter/gather transfers. This scatter/gather capability allows the chaining of multiple DMA controller operations within a channel. During a normal DMA operation software must program the control, count and control, source address, and destination address registers for each transfer. Scatter/gather transfers differ in that these registers are automatically loaded from a linked list data structure in system memory. When a channel completes one transfer, the DMA controller loads the next set of configuration values into the channel registers and the channel continues with the new programming.

22.1.2 Configuration and Status Registers

Table 22-2 lists the DMA configuration and status registers, which are accessed using the **mtdcr** and **mfdcr** instructions. As example, the following PowerPC assembly code writes the control register for DMA channel 0 and then reads the DMA status register:

```
#define DMA2P40_CR0  0x100
#define DMA2P40_SR   0x120

    mtdcr     DMA2P40_CR0,r3            ! write r3 to channel 0 control register
    mfdcr     r4,DMA2P40_SR             ! read contents of status register into r4
```

The DMA configuration and status registers are readable at any time; however, since each register read requires a separate operation, it is not possible to guarantee that the values read from multiple registers correspond to a state that ever existed in the DMA controller. To illustrate, consider software that reads the destination address registers for channel 0 (DMA2P40_DAH0 and DMA2P40_DAL0) and the count for channel 0 (DMA2P40_CTC0). If the DMA controller updates the destination address or count between these operations, the values read differ from what is expected.

While reads can occur at any time, software must not write the configuration registers for any channel that is currently enabled (DMA2P40_CRn[CE]=1). The only exception is that a channel may be disabled by reading the channel control register, clearing the channel enable bit, and then writing the new value to the control register. Note that before enabling the channel enable bit, the status has to be queried until the channel busy status bit is cleared. Once a channel is disabled, all of its configuration registers may be reprogrammed as desired.

Table 22-2. DMA to PLB4 Controller Configuration and Status Registers

Mnemonic	Name	DCR Number	Access	Page
DMA2P40_CR0	DMA to PLB4 Channel Control Register 0	0x0300	R/W	513
DMA2P40_CTC0	DMA to PLB 4 Count and Control Register 0	0x0301	R/W	514
DMA2P40_SAH0	DMA to PLB 4 Source Address High Register 0	0x0302	R/W	515
DMA2P40_SAL0	DMA to PLB 4 Source Address Low Register 0	0x0303	R/W	515

Table 22-2. DMA to PLB4 Controller Configuration and Status Registers (continued)

Mnemonic	Name	DCR Number	Access	Page
DMA2P40_DAH0	DMA to PLB 4 Destination Address High Register 0	0x0304	R/W	516
DMA2P40_DAL0	DMA to PLB 4 Destination Address Low Register 0	0x0305	R/W	516
DMA2P40_SGH0	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 0	0x0306	R/W	516
DMA2P40_SGL0	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 0	0x0307	R/W	516
DMA2P40_CR1	DMA to PLB 4 Channel Control Register 1	0x0308	R/W	513
DMA2P40_CTC1	DMA to PLB 4 Count and Control Register 1	0x0309	R/W	514
DMA2P40_SAH1	DMA to PLB 4 Source Address High Register 1	0x030A	R/W	515
DMA2P40_SAL1	DMA to PLB 4 Source Address Low Register 1	0x030B	R/W	515
DMA2P40_DAH1	DMA to PLB 4 Destination Address High Register 1	0x030C	R/W	516
DMA2P40_DAL1	DMA to PLB 4 Destination Address Low Register 1	0x030D	R/W	516
DMA2P40_SGH1	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 1	0x030E	R/W	516
DMA2P40_SGL1	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 1	0x030F	R/W	516
DMA2P40_CR2	DMA to PLB 4 Channel Control Register 2	0x0310	R/W	513
DMA2P40_CTC2	DMA to PLB 4 Count and Control Register 2	0x0311	R/W	514
DMA2P40_SAH2	DMA to PLB 4 Source Address High Register 2	0x0312	R/W	515
DMA2P40_SAL2	DMA to PLB 4 Source Address Low Register 2	0x0313	R/W	515
DMA2P40_DAH2	DMA to PLB 4 Destination Address High Register 2	0x0314	R/W	516
DMA2P40_DAL2	DMA to PLB 4 Destination Address Low Register 2	0x0315	R/W	516
DMA2P40_SGH2	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 2	0x0316	R/W	516
DMA2P40_SGL2	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 2	0x0317	R/W	516
DMA2P40_CR3	DMA to PLB 4 Channel Control Register 3	0x0318	R/W	513
DMA2P40_CTC3	DMA to PLB 4 Count and Control Register 3	0x0319	R/W	514
DMA2P40_SAH3	DMA to PLB 4 Source Address High Register 3	0x031A	R/W	515
DMA2P40_SAL3	DMA to PLB 4 Source Address Low Register 3	0x031B	R/W	515
DMA2P40_DAH3	DMA to PLB 4 Destination Address High Register 3	0x031C	R/W	516
DMA2P40_DAL3	DMA to PLB 4 Destination Address Low Register 3	0x031D	R/W	516
DMA2P40_SGH3	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 3	0x031E	R/W	516
DMA2P40_SGL3	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 3	0x031F	R/W	516
DMA2P40_SR	DMA to PLB 4 Status Register	0x0320	R/Clear	517
DMA2P40_SGC	DMA to PLB 4 Scatter/Gather Command Register	0x0323	R/W	518
DMA2P40_SLP	DMA to PLB 4 Sleep Mode Register	0x0325	R/W	518
DMA2P40_POL	DMA to PLB 4 Polarity Configuration Register	0x0326	R/W	519

22.1.2.1 DMA to PLB 4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3)

DMA channel control registers (DMA2P40_CR0-DMA2P40_CR3) are used to configure and enable their respective DMA channels. Before a DMA channel can transfer data, the channel control, count and control, source address and destination address registers must be programmed. If a DMA channel is set up for scatter/gather transfers (DMA2P40_SGC[SSGn]=1), the DMA channel control register is automatically loaded from memory (see *Scatter/Gather Transfers* on page 522).

Figure 22-1. DMA to PLB4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA2P40_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Not allowed 1 Device located on the OPB.	For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers, the device is always located on the OPB.
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Half word (16 bits) 010 Word (32 bits) 011 Double word (64 bits) 100 Quad word (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, half word, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quad word size. Use 010 (32 bits) for USB 2.0 Device controller.
7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: • 1, for byte (8-bit) transfers • 2, for half word (16-bit) transfers • 4, for word (32-bit) transfers • 8, for double word (64-bit) transfers • 16, for quad word (128-bit) transfers	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.
8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: • 1, for byte (8-bit) transfers • 2, for half word (16-bit) transfers • 4, for word (32-bit) transfers • 8, for double word (64-bit) transfers • 16, for quad word (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.

9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	Must specify 11 for USB.
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers. Select 0 cycles for USB.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers. Select 0 cycles for USB.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers. Select 0 cycles for USB.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quad word (128-bits) 01 Prefetch 2 quad words 10 Prefetch 4 quad words 11 Prefetch 8 quad words	Used only during memory-to-peripheral and memory to device-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller. Must be 0 for USB.
31		Reserved	

22.1.2.2 DMA to PLB 4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3))

The DMA count and control registers (DMA2P40_CT0-DMA2P40_CT3) contain the number of transfers remaining in the DMA transaction for their respective channels when EOTn[TCn] is programmed as a terminal count output, along with additional channel control information. In addition to the transfer count, the count and control registers

contain fields used to enable interrupts for terminal count, end of transfer and error conditions. Other fields enable parity checking during the peripheral portion of peripheral type (DMA2P40_CRn[TM]=00) transfers, define sub channel ID information, and enable bursting during peripheral and device-paced memory transfers. For additional details on bursting, see *Peripheral and Device-Paced Memory Bursts* on page 519.

When a DMA channel is setup for scatter/gather transfers (DMA2P40_SGC[SSGn]=1) the count and control register is automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 522.

The value in the Transfer Count (TC) field of DMA count and control register is interpreted as the number of transfers of the width specified in DMA2P40_CRn[PW], not the total number of bytes. The maximum number of transfers is 1024K, and each transfer can be either 1, 2, 4, 8, or 16 bytes as programmed in DMA2P40_CRn[PW]. The maximum count of 1024K transfers is programmed by writing zero to DMA2P40_CTn[TC].

The sub channel ID bits can be used to allow one DMA channel to support up to eight different peripherals. Bits 5:7 control the external logic that selects the current peripheral.

When EOTn[TCn] is programmed as an end-of-transfer input (DMA2P40_CRn[ETD]=0), the channel will not stop when the count reaches zero. Instead, DMA2P40_CTn[TC] continues to count down past zero until EOTn is asserted. For DMA2P40_CRn[ETD]=0, DMA2P40_CTn[TCE] must be set to zero.

Figure 22-2. DMA to PLB4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA2P40_CR0-DMA2P40_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Disable end of transfer interrupts 1 Enable end of transfer interrupts	EOT interrupts are further qualified by DMA2P40_CR0-DMA2P40_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Disable error interrupts 1 Enable error interrupts	Error interrupts are further qualified by DMA2P40_CR0-DMA2P40_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Device-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRSTSIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

22.1.2.3 DMA to PLB 4 Source Address Registers

The DMA Source Address Registers (DMA2P40_SAHn and DMA2P40_SALn) contain the source address for memory-to-memory and memory-to-peripheral transfers. If a DMA channel is setup for scatter/gather transfers (DMA2P40_SGC[SSGn]=1) the source address registers are automatically loaded from memory. For additional details *Scatter/Gather Transfers* on page 522

The source address must be aligned at the transfer width programmed in DMA2P40_CRn[TW], otherwise the error bit (DMA2P40_SR[RIn]) is set for the channel and no transfer occurs. If the source address increment bit in the channel's control register is set (DMA2P40_CRn[SAI]) the address is incremented by the transfer width after each data transfer. In contrast, if the channel is performing a memory-to-peripheral transfer and the address decrement bit is set (DMA2P40_CRn[DEC]=1), the address is decremented by the transfer width after each transfer.

Figure 22-3. DMA to PLB4 Source Address High Registers (DMA2P40-SAH0-DMA2P40-SAH3)

0:27		Reserved	
28:31	USA	Upper 4-bits of source address for memory-to-memory and memory-to-peripheral transfers.	

Figure 22-4. DMA to PLB4 Source Address Low Registers (DMA2P40-SAL0-DMA2P40-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.	

22.1.2.4 DMA to PLB 4 Destination Address Registers

DMA destination address registers (DMA2P40_DAH0-DMA2P40_DAH3 and DMA2P40_DAL0-DMA2P40_DAL3) contain the destination address for memory-to-memory and peripheral-to-memory transfers. When a DMA channel is configured for scatter/gather transfers (DMA_SGC[SSGn]=1) the destination address registers are automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 522

The destination address must be aligned at the transfer width programmed in DMA2P40_CRn[TW], otherwise the error bit (DMA2P40_SR[RIn]) is set for the channel and no transfer occurs. If the destination address increment bit in the channel's control register is set (DMA2P40_CRn[DAI]) the address is incremented by the transfer width after each data transfer. However, if the channel is performing a peripheral-to-memory transfer and the address decrement bit is set (DMA2P40_CRn[DEC]=1), the destination address is decremented by the transfer width after each transfer.

Figure 22-5. DMA to PLB4 Destination Address High Registers (DMA2P40_DAH0-DMA2P40_DAH3)

0:27		Reserved	
28:31	UDA	Upper 4-bits of destination address for memory-to-memory and peripheral-to-memory transfers.	

Figure 22-6. DMA to PLB4 Destination Address Low Registers (DMA2P40_DAL0-DMA2P40_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.	

22.1.2.5 DMA to PLB 4 Scatter/Gather Descriptor Address Registers

When a DMA channel is setup for scatter/gather transfers (DMA2P40_SGC[SSGn]=1), the Scatter/Gather Descriptor Address Registers (DMA2P40_SGH0-DMA2P40_SGH3 and DMA2P40_SGL0-DMA2P40_SGL3) contain the memory address of the next scatter/gather descriptor table. Prior to starting a scatter/gather transfer,

software must write the address of the channel's descriptor table to DMA2P40_SGHn and DMA2P40_SGLn. Once the scatter/gather transfer starts, DMA2P40_SGHn and DMA2P40_SGLn are automatically updated from the descriptor table. For additional details see *Scatter/Gather Transfers* on page 522.

Figure 22-7. DMA to PLB4 Scatter/Gather Desc Adr High Reg (DMA2P40_SGH0-DMA2P40_SGH3)

0:27		Reserved	
28:31	USGD	Upper 4-bits of address of next scatter/gather descriptor table.	

Figure 22-8. DMA to PLB4 Scatter/Gather Desc Adr Low Reg (DMA2P40_SGL0-DMA2P40_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.	

22.1.2.6 DMA to PLB 4 Status Register (DMA2P40_SR)

DMA Status Register (DMA2P40_SR) provides status information for each of the DMA channels. Bits in DMA2P40_SR are set in hardware, and can be either read or cleared by software. Clearing is performed by writing a word to DMA2P40_SR containing a 1 in any bit position to be cleared and 0 in all other bit positions.

The terminal count status (DMA2P40_SR[CSn]), end of transfer status (DMA2P40_SR[TSn]) and error status (DMA2P40_SR[RIn]) must be cleared for a DMA channel to operate. If a scatter/gather operation generates an interrupt for any of the above conditions, the channel pauses until software clears the associated status field(s) in DMA2P40_SR.

Figure 22-9. DMA to PLB4 Status Register (DMA2P40_SR)

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA2P40_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA2P40_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During scatter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept active until the last link is completed.
28:31		Reserved	

22.1.2.7 DMA to PLB 4 Scatter/Gather Command Register (DMA2P40_SGC)

The DMA Scatter/Gather Command Register (DMA2P40_SGC) is a 32-bit register, of which 12-bits are implemented. Bits 0:3 are the Start Scatter/Gather Enable bits for channels 0 to 3, bits 4:7 determine whether a given channel's Scatter/Gather Descriptor Table resides in PLB or OPB address space, and bits 16:19 are the corresponding Enable Mask bits for the Start Scatter/Gather Enable bits. Setting a Start Scatter/Gather Enable bit causes the selected channel to begin a scatter/gather operation, while writing a 0 stops the Scatter/Gather operation. To start or stop a specific Scatter/Gather channel, the corresponding Enable Mask bit must be set to 1; otherwise, the register holds the previous value. Note that halting a scatter/gather transfer prevents the channel from continuing to the next descriptor, but does not stop the transfer currently in progress.

Upon completion of a scatter/gather sequence of transfers the DMA controller clears DMA2P40_SGC[SSGn]. If an error occurs when the DMA controller is reading the Scatter/Gather descriptor table, DMA2P40_SGC[SSGn] is cleared for the affected channel, and the channel's error status bit (DMA2P40_SR[RIn]) is set. For additional details see *Scatter/Gather Transfers* on page 522.

Figure 22-10. DMA to PLB4 Scatter/Gather Command Register (DMA2P40_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See *Scatter/Gather Transfers* on page 522.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

22.1.2.8 DMA to PLB 4 Sleep Mode Register (DMA2P40_SLP)

The Sleep Mode Register (DMA2P40_SLP) enables the DMA controller to enter sleep (low-power) mode and programs the number of PLB clock cycles to wait when the controller is idle before going to sleep. The DMA controller only goes to sleep when no DMA channels are enabled, no scatter/gather operation is pending, and no configuration or status (DCR) register operations are in progress. Reading or writing any of the DMA DCRs awakens the DMA controller.

To enable sleep mode set DMA2P40_SLP[SME] and CPM0_ER[DMA]. When sleep mode is enabled and the DMA controller becomes idle the 10-bit idle timer begins counting down from the programmed value. Only the upper 5 bits of the idle counter are programmable; the lower 5 bits are hard coded to 5'0b11111; therefore, the minimum granularity of the idle timer is 32 PLB clock cycles. When the counter reaches zero, the controller is placed in sleep mode.

Figure 22-11. DMA to PLB4 Sleep Mode Register (DMA2P40_SLP)			
0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hard coded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

22.1.2.9 DMA to PLB 4 Polarity Configuration Register (DMA2P40_POL)

The Polarity Configuration Register (DMA2P40_POL) is present but is reserved since external DMA is not supported.

22.1.3 Channel Priorities

The priority of DMA transfers is controlled on a per-channel basis by the channel priority field in the channel control register. *Table 22-3* shows the different priority settings for DMA2P40_CRn[PW].

Table 22-3. DMA to PLB 4 Transfer Priorities

DMA2P40_CRx[CP]	Priority Level
0b00	Low
0b01	Medium Low
0b10	Medium High
0b11	High

These priorities serve two purposes. First, the DMA controller arbitrates among all actively requesting channels and selects the highest priority one for service. If multiple channels request at the same priority, the arbiter selects the lowest numbered channel for service. Secondly, DMA2P40_CRn[CP] determines the priority of the internal PLB transactions that the DMA controller uses to read and write data. During a Scatter/Gather transfer, the next descriptor is read using the priority current programmed in DMA2P40_CRn[CP].

22.1.4 Data Parity During DMA Peripheral Transfers

During memory-to-memory transfers any data error checking and/or correction is dependent on the configuration of the relevant memory controller. For example, if ECC is enabled on the SDRAM controller, only uncorrectable errors are reported to the DMA controller.

22.1.5 Peripheral and Device-Paced Memory Bursts

Normally, device-paced (hardware initiated) DMA transfers move one data item at a time.

To improve performance, the DMA controller supports fixed-length bursts on the device-paced side of device-paced memory-to-memory transfers. When bursting is enabled (DMA2P40_CTn[BTEN]=1), the DMA controller responds to an active DMAReqn by reading or writing DMA2P40_CTn[BSIZ] data items in one atomic, non-interruptible operation.

When a DMA channel has bursting enabled (DMA2P40_CTn[BTEN]=1) it is required that the prefetch count (DMA2P40_CRn[PF]) be sufficient to complete at least one fixed-length burst transfer. In addition, the transfer count (DMA2P40_CTn[TC]) must be a multiple of the burst size (DMA2P40_CTn[BSIZ]). If either of these requirements is not met, an error results, no data transfer occurs, and the channel is disabled. For additional details see *Errors* on page 520

22.1.6 Errors

The DMA controller detects and reports five types of errors: address alignment, burst count, burst prefetch, PLB/OPB time-out, and slave errors. The DMA controller reports errors through the channel error status bit in the DMA status register (DMA2P40_SR[RIn]). Whenever the error status bit for a channel is set, the channel enable bit (DMA2P40_CRn[CE]) is cleared, disabling the channel. An interrupt signal is also presented to the interrupt controller if DMA2P40_CRn[CIE] and DMA2P40_CTn[EIE] are set. For more information on interrupt processing, see *DMA to PLB4 Interrupts* on page 521

When an error is detected during the execution of an external cycle, the DMA terminates the transfer and deasserts all external signals.

When the DMA controller has multiple channels active, an error may be reported on the current channel which was in actuality caused by a previously active channel. This causes the current channel to have its error status bit set; therefore, for deterministic error analysis with multiple DMA channels active, the slave bus controller's error status registers (the bus error address register in particular) must be queried to isolate the actual channel that encountered the error. In any case, the channel causing the errors will eventually cause all active channels, including itself, to be disabled.

22.1.6.1 Address Alignment Error

The source address (DMA2P40_SAHn || DMA2P40_SALn) and destination address (DMA2P40_DAHn || DMA2P40_DALn) registers must be aligned to the programmed transfer width (DMA2P40_CRn[PW]).The address alignment rules are outlined in *Table 22-4*. In addition, when a channel is configured for scatter/gather transfers, the scatter/gather table must be quad word-aligned. If the source, destination, and scatter/gather address registers are not appropriately aligned, an error occurs immediately after the channel is enabled.

Table 22-4. Address Alignment Requirements

DMA2P40_CRx[PW] Setting	Required Alignment for: Source Address (DMA2P40_SAHn || DMA2P40_SALn) Destination Address (DMA2P40_DAHn || DMA2P40_DALn)
0b000	Byte (8-bit)
0b001	Halfword (16-bit)
0b010	Word (32-bit)
0b011	Doubleword (64-bit)
0b100	Quadword (128-bit)

22.1.6.2 Burst Count Error

If peripheral/device-paced bursting is enabled for a particular DMA channel (DMA2P40_CTn[BTEN]=1), the programmed transfer count (DMA2P40_CTn[TC]) must be a multiple of the burst size (DMA2P40_CTn[BSIZ]).

22.1.6.3 Burst Prefetch Error

If peripheral/device-paced bursting is enabled for a particular DMA channel (DMA2P40_CTn[BTEN]=1), the programmed prefetch count (DMA2P40_CRn[PF]) must be sufficient to perform at least one fixed-length burst. More precisely, DMA2P40_CRn[PF] must prefetch at least DMA2P40_CTn[BSIZ] * DMA2P40_CRn[PW] bytes.

22.1.6.4 PLB or OPB Timeout

The DMA controller uses PLB and OPB operations to read and write memory. A PLB or OPB timeout results if the DMA controller attempts to access a non-existent memory location. This will occur if the source, destination or scatter/gather address registers do not map to valid memory locations.

22.1.6.5 Slave Transfer Errors

If the DMA controller detects an error from a slave, it stops the execution of the current transfer, disables the channel, and then reports an error. If a signal is asserted by the PLB slave as a result of such error, the DMA finishes any active read/write pair transfer before disabling the channel. An SDRAM uncorrectable ECC error and an EBC bank protection error are examples of slave errors.

22.1.7 DMA to PLB4 Interrupts

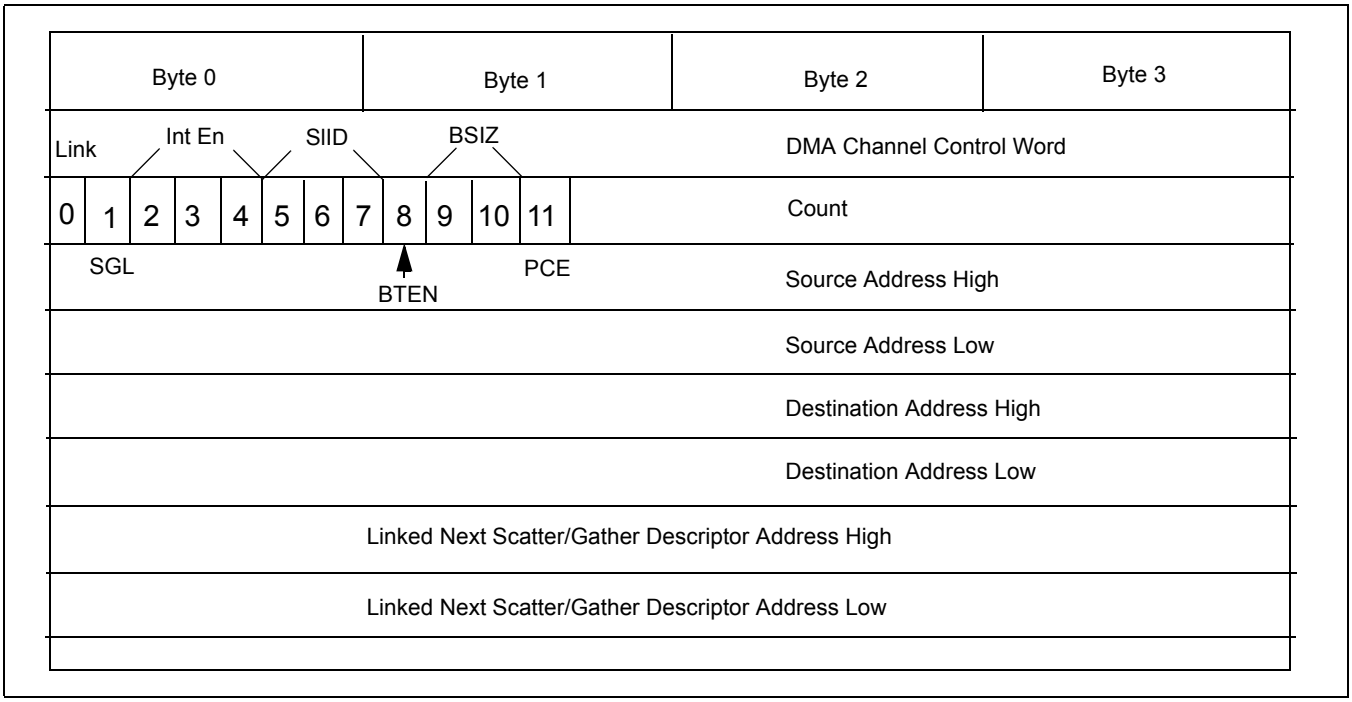
Each DMA channel can generate interrupts for end of transfer, terminal count and error conditions. All interrupts from a particular DMA channel are enabled by setting the channel enable bit in the channel's control register (DMA2P40_CRn[CIE]=1) and enabling the specific type of interrupt in the count and control register: end of transfer (DMA2P40_CTn[ETIE]), terminal count (DMA2P40_CTn[TCIE]), and error (DMA2P40_CTn[EIE]). When an interrupt occurs for a given channel, the DMA controller sends a signal to the Universal Interrupt Controller. For the PPC440GP's CPU to take an exception, interrupts from the particular DMA channel must be enabled in the interrupt controller's interrupt enable register (UIC0_ER[DMA2P40]=1). Also, the CPU's machine state register's interrupt enable bit must be enabled for the appropriate interrupt type (critical or non-critical), MSR[EE,CE]. See *Universal Interrupt Controller* on page 223 for more information on interrupt controller processing.

When the DMA controller generates an interrupt, the interrupt remains active until the appropriate bits are cleared in the DMA Status Register (DMA2P40_SR). In addition, interrupts from a channel performing a scatter/gather transfer cause the channel to pause until the interrupt is cleared.

22.1.8 Scatter/Gather Transfers

With a normal DMA transfer it is necessary to program a channel's control, count and control, source, and destination registers for each transfer. The scatter/gather capability of the DMA controller provides a more efficient solution for applications that require multiple transactions on a single DMA channel. Instead of individually programming a channel's registers, software creates a set (linked list) of descriptor tables in system memory. *Figure 22-12* illustrates the required table format.

Figure 22-12. Scatter/Gather Descriptor Table



The most significant bit of word 2, the transfer count and configuration register value, is the link (LK) bit. Bit number 1 (SGL) of the same word is the scatter/gather locator bit. It determines the location of the next scatter/gather table (0=PLB space, 1=OPB space). Other than these bits, the contents of the descriptor table are identical to the associated register definitions.

To configure a channel for a scatter/gather transfer the DMA Scatter/Gather Descriptor Address Registers (DMA2P40_SGHn and DMA2P40_SGLn) for the channel are set to the address of the first descriptor table, which must be quad word (16 byte) aligned. The linked list of descriptor tables for a given programming of the DMA controller must not cross between PLB or OPB address space. The location of the linked list is programmed into DMA2P40_SGC[SGLn].

To begin a scatter/gather transfer, software writes DMA2P40_SGC with the enable mask (DMA2P40_SGC[EMn]) set, the start scatter/gather bit (DMA2P40_SGC[SSGn]) set and indicates the location (PLB or OPB) of the descriptor tables via DMA2P40_SGC[SGLn]. The DMA controller then reads the descriptor table at address (DMA2P40_SGHn || DMA2P40_SGLn) and updates the DMA controller registers as shown in *Table 22-5*. Upon receiving the data from the scatter/gather descriptor table, the channel's terminal count status bit (DMA2P40_SR[TCn]) and end of transfer status bit (DMA2P40_SR[EOTn]) are automatically cleared.

Table 22-5. DMA to PLB 4 Registers Loaded from Scatter/Gather Descriptor

Descriptor Table Entry	Register Loaded
Channel Control Word	DMA2P40_CRn
Source Address	DMA2P40_SAHx and DMA2P40_SALn

Table 22-5. DMA to PLB 4 Registers Loaded from Scatter/Gather Descriptor (continued)

Descriptor Table Entry	Register Loaded
Destination Address	DMA2P40_DAHn and DMA2P40_DALn
Count and Control	DMA2P40_CTn
Next Descriptor Address	DMA2P40_SGHn and DMA2P40_SGLn

After loading the channel's registers from the descriptor table, the transfer functions as a normal non-scatter/gather operation.

If the LK (link) bit was not set the scatter/gather process stops when the current transfer completes. Otherwise, the DMA controller reads the descriptor table at address (DMA2P40_SGHn || DMA2P40_SGLn) and the process repeats.

22.1.9 Programming the DMA2P40 Controller

Before the DMA controller can transfer data it must be configured, both globally and on a per-channel basis. Global settings include the DMA Polarity Register (DMA2P40_POL) and DMA Sleep Mode Register (DMA2P40_SLP). For most applications, these registers should be configured when the DMA controller is first initialized. To prevent spurious activity resulting from changing the active level for DMAReqn, DMAAckn, or EOTn[TCn], a channel's configuration in the Polarity Register should not be altered when the channel is enabled (DMA2P40_CRn[CE]=1).

Each channel has Control (DMA2P40_CRn), Count and Control (DMA2P40_CTn), Source Address (DMA2P40_SAHn and DMA2P40_SALn), Destination Address (DMA2P40_DAHn and DMA2P40_DALn), and Scatter/Gather Descriptor Address (DMA2P40_SGHn and DMA2P40_SGLn) registers. The type of DMA transfer determines which of these registers must be programmed and what causes the channel to start. In all cases, the terminal count (CSn), end of transfer (TSn) and error status (RIn) bits in the DMA Status Register (DMA2P40_SR) must be cleared or the channel will not start.

The programming information that follows assumes that the DMA controller is operating in non-scatter/gather mode. To use scatter/gather transfers the channel configuration data must be written into a set of descriptor tables in system memory. See *Scatter/Gather Transfers* on page 522 for additional details.

22.1.9.1 Memory-to-Memory Transfers

Memory-to-memory transfers can be initiated either by software or by an external device. If initiated via software, the transfer begins as soon as the channel is configured and enabled. When initiated by hardware (also known as a device-paced memory-to-memory transfer), software configures the channel for a memory-to-memory move and transfers begin when an external device places an active request on the channel request line, DMAReqn.

Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers

To perform a device-paced memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P40_CRn[PW]) to the width of the device-paced memory.
2. Set the source (DMA2P40_SAHn and DMA2P40_SALn) and destination (DMA2P40_DAHx and DMA2P40_DALn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P40_CRn[PW]), otherwise an alignment error will occur.
3. In the count and control register:

Caution: Program the desired count in the transfer count, TC, field.

Caution: Optionally enable bursting, BRSTEN=1, and set the burst size field, BRSTSIZ, as desired.

4. Clear the channel's status bits in the DMA status register (DMA2P40_SR).
5. In the channel control register (DMA2P40_CRn):
   a. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
   b. If the device-paced memory is at the source memory location set PL=1.
   c. Set the source address increment, SAI, and destination address increment, DAI, as desired.
   d. Set the transfer mode to device-paced memory-to-memory, TM=0b11.
   e. If the memory controller is on the OPB set the source location bit, SL=1. If the controller is on the PLB, clear SL.
   f. Enable the channel, CE=1.

Once the DMA channel is configured for this mode, the external device initiates a transfer by activating the DMAReqn input. The PPC440EP chip then reads the source memory, buffers the data in the DMA controller and then outputs the data to the destination memory address. Transfers continue as long as the controlling device maintains an active signal on DMAReqn and the channel count register (DMA2P40_CTn) is non-zero. To pause a device paced memory-to-memory transfer, the controlling device must deassert DMAReqn one PerClk cycle before the last cycle in the device-paced memory access.

Software-Initiated Memory-to-Memory Transfers (Non-Device Paced)

To perform a software-initiated memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P40_CRn[PW]) as desired.
2. Set the source (DMA2P40_SAHn and DMA2P40_SALn) and destination (DMA2P40_DAHn and DMA2P40_DALn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P40_CRn[PW]), otherwise an alignment error will occur.
3. Program the transfer count field in the control and count register (DMA2P40_CTn[TC]) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA2P40_SR).
5. In the channel control register (DMA2P40_CRn):
   a. Optionally enable the DMA buffer, BEN=1.
   b. Set the source address increment, SAI, and destination address increment, DAI, as desired.
   c. Set the transfer mode to software-initiated memory-to-memory, TM=0b10.
   d. If the source memory controller is on the OPB set the source location bit, SL=1. If the controller is on the PLB, clear SL.
   e. If the destination memory controller is on the OPB set the PL=1. If the memory controller is on the PLB clear the PL bit.
   f. Enable the channel, CE=1.

Once the channel is enable the DMA controller transfers data from source to destination until the channel count reaches zero. Note that memory-to-memory transfers initiated by software do not use DMAReqn or DMAAckn.

22.2 DMA to PLB3 Controller (DMA2P30)

DMA to PLB3 is attached to the 64-bit processor local bus. *Table 22-6* lists the DMA2P30 channel assignments in PPC440EP.

Table 22-6. DMA to PLB 3 Channel Assignments

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 0	Sub-Channel 0	UART 0 Receive	
	Sub-Channel 1	UART 2 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 1	Sub-Channel 0	UART 0 Transmit	
	Sub-Channel 1	UART 2 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 2	Sub-Channel 0	UART 1 Receive	
	Sub-Channel 1	UART 3 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

Table 22-6. DMA to PLB 3 Channel Assignments (continued)

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 3	Sub-Channel 0	UART 1 Transmit	
	Sub-Channel 1	UART 3 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

22.2.1 External Interface Signals

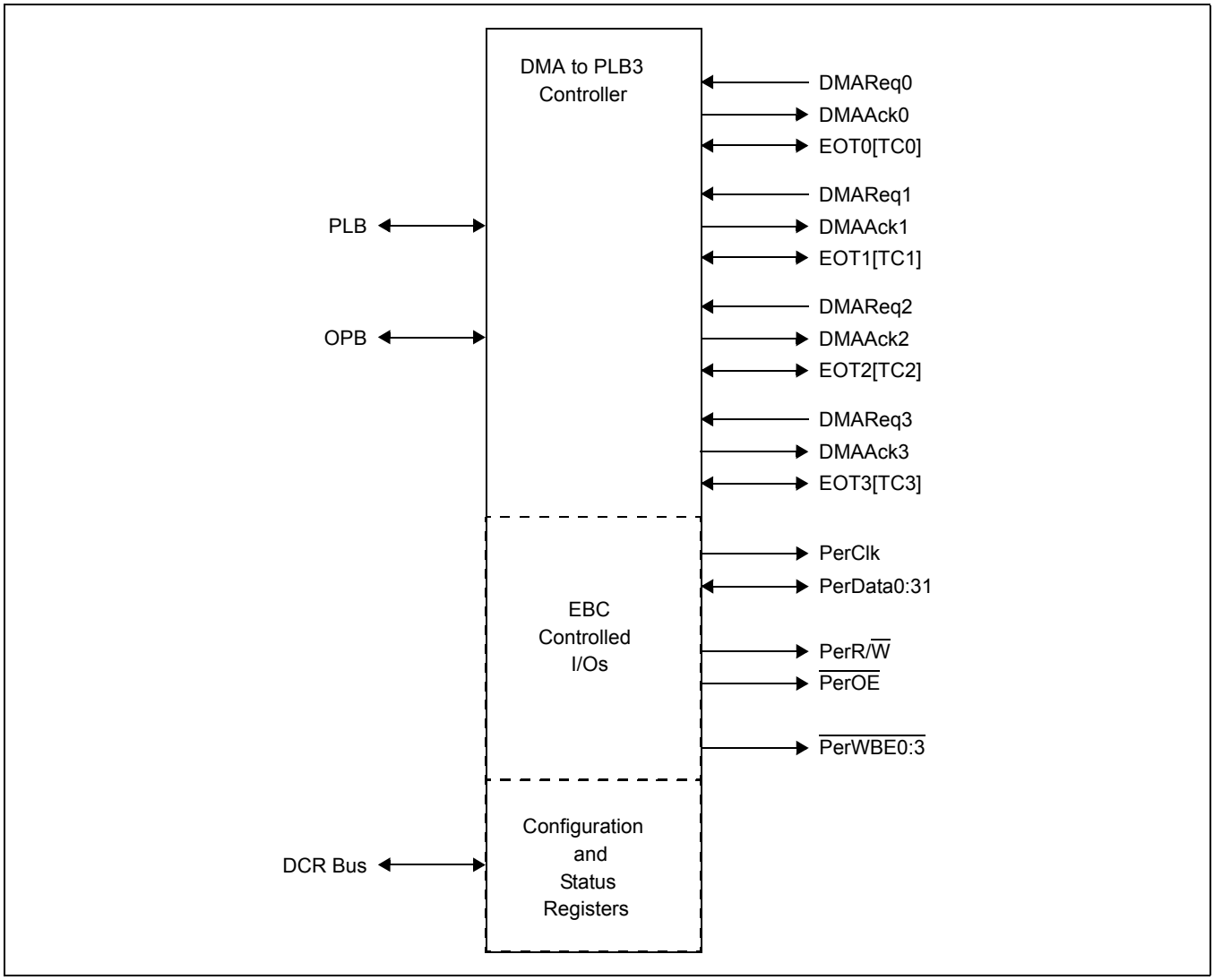
Figure 22-13 illustrates the external I/Os associated with the DMA controller and the EBC I/Os used during external peripheral transfers. External peripheral and EBC device-paced memory transfers request service from the DMA controller by driving a DMA request line (DMAReq0-3) active. For peripheral mode transfers the DMA controller acknowledges the request and transfers data by asserting a DMA acknowledge signal (DMAAck0-3). In contrast, an EBC device-paced memory-to-memory transfer occurs when the chip select ($\overline{\text{PerCSn}}$) associated with the memory location is driven active. The timing of $\overline{\text{PerCSn}}$ and the other EBC I/Os is determined by the Bank Access Parameter Register (EBC0_BnAP) for the particular memory location. See *External Bus Controller* on page 465 for details on EBC configuration and timings.

Table 22-7. DMA to PLB3 Controller External I/Os

Signal	Usage
DMAReqn	DMA Request, used to request either a peripheral mode transfer or an EBC device-paced memory-to-memory transfer.
DMAAckn	DMA Acknowledge, instructs an EBC-attached DMA peripheral to transfer data.
EOTn[TCn]	End of Transfer or Terminal Count. Used to stop the channel when programmed as EOT. When configured as TC, goes active when the channel transfer count register (DMA2P30_CTn) reaches zero.

Note: The active level (polarity) of the DMAReqn, DMAAckn, and EOTn[TCn] signals is individually programmable. See *DMA to PLB 3 Polarity Configuration Register (DMA2P30_POL)* on page 530.. Operating information and timing waveforms are presented generically for active-high signals.

Figure 22-13. DMA to PLB 3 External Bus Controller Signals



22.2.2 Functional Overview

As a specialized controller, the DMA unit provides system designers and programmers with a highly efficient method of moving data. During any DMA transfer the controller always buffers data read from the source prior to writing the data to the destination. Since many buses, including the internal PLB and the SDRAM interface, provide substantially better performance when bursting data, the DMA controller includes a 32-byte (4 double words) buffer. This buffer is enabled on a per-channel basis by setting DMA2P30_CRn[BEN] and serves to minimize the number of discrete memory transactions. Each of the four DMA channels is configurable for either peripheral or memory-to-memory transfers.

22.2.3 Peripheral Mode Transfers

Peripherals are either devices attached to the EBC interface via the DMAReqn and DMAAckn lines, or the internal serial port (UART0). Memory is any address accessible from the PLB, including PLB-mapped PCI address space, SDRAM memory, and memory and memory-mapped devices on the peripheral bus. During a peripheral mode transfer the peripheral requests a DMA transfer by asserting a DMA request line. For UART0, this signal is internal

to the PPC440EP, while external peripherals use one of the DMAReqn lines. When the requesting channel has the highest priority of any active channel, the requesting device receives a DMA acknowledge. In the case of external peripherals, the appropriate DMAAckn line is driven to the active state, with the timing specified in the DMA Control Register for the channel (DMA2P30_CRn).

There are two types of peripheral mode transfers: peripheral-to-memory and memory-to-peripheral. A peripheral-to-memory transfer reads data from a DMA device, while a memory-to-peripheral transfer writes data. In both cases, the peripheral interface never bursts and data is transferred at the width of the peripheral. If the DMA buffer is disabled for the active channel (DMA2P30_CRn[BEN]=0), each peripheral transfer causes a corresponding memory operation.

When buffering is enabled during a peripheral-to-memory transfer, data is collected until the 32-byte buffer is full, the peripheral deasserts DMAReqn, a higher priority DMA request becomes pending, or the channel completes. The buffer contents are then written to the target memory as efficiently as possible. If the initial programming of the channel's destination address register (DMA2P30_DAn) is 32-byte aligned, the buffer is emptied in one burst operation to the target memory.

Memory-to-peripheral transfers differ since the amount of data that will be requested by the peripheral is unknown. If the DMA buffer is disabled (DMA2P30_CRn[BEN]=0) a discrete source memory read occurs for each element in the DMA transfer. Since this is inefficient, the buffer should only be disabled for low data rate transfers or when the source memory is FIFO-like, and reads are therefore destructive. When the 32-byte buffer is enabled the controller uses the setting in DMA2P30_CRn[PF] to prefetch 1, 2, or 4 64-bit double words from the source memory. The DMA controller provides data from the buffer until the peripheral deasserts its request, the channel is interrupted by one of higher priority, or the transfer completes. Whenever any of these conditions occurs any unused data in the DMA buffer is discarded.

22.2.4 Memory-to-Memory Transfers

The DMA controller can perform either device-paced (hardware-initiated) or software-initiated memory-to-memory transfers. Device-paced memory transfers function identically to peripheral mode transfers, except that a chip select (PerCSn) serves as the DMA acknowledge instead of a DMAAckn output. As with peripheral mode transfers, bursts never occur on the device-paced side of the transaction.

Software-initiated memory-to-memory transfers between memories with fixed timings provide the best overall performance. During a software-initiated transfer the DMA controller knows the exact amount of data to be transferred. As a result, when the 32-byte DMA buffer is enabled (DMA2P30_CRn[BEN]=1) the controller uses bursts as much as possible. To ensure the highest bandwidth, source and destination addresses should be aligned on 32-byte boundaries.

There are three cases that limit the ability to burst during software-initiated memory-to-memory transfers. Bursting is not possible from the source memory when the source address increment is zero (DMA2P30_CRn[SAI]=0). Similarly, the DMA controller does not burst to the destination when the destination address increment is zero (DMA2P30_CRn[DAI]=0). Finally, the DMA controller cannot burst to or from any address that maps to a device-paced (EBC0_BnAP[RE]=1) EBC chip select (PerCSn).

22.2.5 Scatter/Gather Transfers

Each of the four DMA channels supports scatter/gather transfers. This scatter/gather capability allows the chaining of multiple DMA controller operations within a channel. During a normal DMA operation software must program the control, source address, destination address, and count registers for each transfer. Scatter/gather transfers differ in that these registers are automatically loaded from a linked list data structure in system memory. When a channel completes one transfer the DMA controller loads the next set of configuration values into the channel's registers and the channel continues with the new programming.

22.2.6 Configuration and Status Registers

Table 22-8 lists the DMA2P30 configuration and status registers, each of which is accessed using the **mtdcr** and **mfdcr** instructions. As example, the following assembly code writes DMA2P30_CR0 and then reads DMA2P30_SR:

```
#define DMA2P30_CR0  0x100
#define DMA2P30_SR   0x120

    mtdcr     DMA2P30_CR0,r3            ! write r3 to channel 0 control register
    mfdcr     r4,DMA2P30_SR             ! read contents of status register into r4
```

The DMA configuration and status registers are readable at any time. However, because each register read requires a separate operation, it is not possible to guarantee that the values read from multiple registers correspond to a state that ever existed in the DMA controller. To illustrate, consider software that reads the destination address for channel 0 (DMA2P30_DA0) and then the count for channel 0 (DMA2P30_CT0). If the DMA controller updates the count between these two operations, the values read differ from what is expected.

While reads can occur at any time, software must not write the configuration registers for any channel that is currently enabled (DMA2P30_CRn[CE]=1). The only exception is that a channel may be disabled by reading the channel control register, clearing the channel enable bit, and then writing the new value to the control register. Once a channel is disabled, all of its configuration registers may be reprogrammed as desired.

Table 22-8. DMA to PLB 3 Controller Configuration and Status Registers

Mnemonic	DCR Address	Access	Description	Page
DMA2P30_CR0	0x0100	R/W	DMA to PLB 3 Channel Control Register 0	532
DMA2P30_CT0	0x0101	R/W	DMA to PLB 3 Count Register 0	534
DMA2P30_DA0	0x0102	R/W	DMA to PLB 3 Destination Address Register 0	534
DMA2P30_SA0	0x0103	R/W	DMA to PLB 3 Source Address Register 0	534
DMA2P30_SG0	0x0104	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 0	535
DMA2P30_SC0	0x0107	Page	DMA to PLB3 Subchannel ID for Register 0	536
DMA2P30_CR1	0x0108	R/W	DMA to PLB 3 Channel Control Register 1	532
DMA2P30_CT1	0x0109	R/W	DMA to PLB 3 Count Register 1	534
DMA2P30_DA1	0x010A	R/W	DMA to PLB 3 Destination Address Register 1	534
DMA2P30_SA1	0x010B	R/W	DMA to PLB 3 Source Address Register 1	534
DMA2P30_SG1	0x010C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 1	535
DMA2P30_SC1	0x010F	Page	DMA to PLB3 Subchannel ID for Register 1	536
DMA2P30_CR2	0x0110	R/W	DMA to PLB 3 Channel Control Register 2	532
DMA2P30_CT2	0x0111	R/W	DMA to PLB 3 Count Register 2	534
DMA2P30_DA2	0x0112	R/W	DMA to PLB 3 Destination Address Register 2	534
DMA2P30_SA2	0x0113	R/W	DMA to PLB 3 Source Address Register 2	534
DMA2P30_SG2	0x0114	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 2	535
DMA2P30_SC2	0x0117	Page	DMA to PLB3 Subchannel ID for Register 2	536
DMA2P30_CR3	0x0118	R/W	DMA to PLB 3 Channel Control Register 3	532
DMA2P30_CT3	0x0119	R/W	DMA to PLB 3 Count Register 3	534

Table 22-8. DMA to PLB 3 Controller Configuration and Status Registers (continued)

Mnemonic	DCR Address	Access	Description	Page
DMA2P30_DA3	0x011A	R/W	DMA to PLB 3 Destination Address Register 3	534
DMA2P30_SA3	0x011B	R/W	DMA to PLB 3 Source Address Register 3	534
DMA2P30_SG3	0x011C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 3	535
DMA2P30_SC3	0x011F	Page	DMA to PLB3 Subchannel ID for Register 3	536
DMA2P30_SR	0x0120	R/Clear	DMA to PLB 3 Status Register	531
DMA2P30_SGC	0x0123	R/W	DMA to PLB 3 Scatter/Gather Command Register	535
DMA2P30_ADR	0x0124	R/W	DMA Address Decode Register	536
DMA2P30_SLP	0x0125	R/W	DMA to PLB 3 Sleep Mode Register	531
DMA2P30_POL	0x0126	R/W	DMA to PLB 3 Polarity Configuration Register	530

22.2.6.1 DMA to PLB 3 Polarity Configuration Register (DMA2P30_POL)

DMA2P30_POL is used to set the polarity (active state) of the external DMA I/O signals: DMAReqn, DMAAckn, and EOTn[TCn]. As shown in *Figure 22-14*, if a bit in DMA2P30_POL is 0, the corresponding signal is active high, otherwise the signal is active low.

Whenever any of the EOT polarities are changed (DMA2P30_POL[EnP), software must subsequently clear the corresponding EOT status bits in DMA2P30_SR[TS0:3] before enabling the associated DMA channel. This is necessary to prevent a channel from being disabled because of an incorrect EOT status stored in the status bits.

Figure 22-14. DMA to PLB3 Polarity Configuration Register (DMA2P30_POL)

0	R0P	DMAReq0 Polarity 0 DMAReq0 is active high 1 DMAReq0 is active low	
1	A0P	DMAAck0 Polarity 0 DMAAck0 is active high 1 DMAAck0 is active low	
2	E0P	EOT0[TC0] Polarity 0 EOT0[TC0] is active high 1 EOT0[TC0] is active low	
3	R1P	DMAReq1 Polarity 0 DMAReq1 is active high 1 DMAReq1 is active low	
4	A1P	DMAAck1 Polarity 0 DMAAck1 is active high 1 DMAAck1 is active low	
5	E1P	EOT1[TC1] Polarity 0 EOT1[TC1] is active high 1 EOT1[TC1] is active low	
6	R2P	DMAReq2 Polarity 0 DMAReq2 is active high 1 DMAReq2 is active low	
7	A2P	DMAAck2 Polarity 0 DMAAck2 is active high 1 DMAAck2 is active low	
8	E2P	EOT2[TC2] Polarity 0 EOT2[TC2] is active high 1 EOT2[TC2] is active low	

9	R3P	DMAReq3 Polarity 0 DMAReq3 is active high 1 DMAReq3 is active low	
10	A3P	DMAAck3 Polarity 0 DMAAck3 is active high 1 DMAAck3 is active low	
11	E3P	EOT3[TC3] Polarity 0 EOT3[TC3] is active high 1 EOT3[TC3] is active low	
12:31		Reserved	

22.2.6.2 DMA to PLB 3 Sleep Mode Register (DMA2P30_SLP)

DMA2P30_SLP enables the DMA controller to enter sleep (low-power) mode and programs the number of PLB clock cycles to wait when the controller is idle before going to sleep. The DMA controller only goes to sleep when no DMA channels are enabled and no configuration or status (DCR) register operations are in progress. Reading or writing any of the DMA control or status register awakens the DMA controller.

To enable sleep mode, set DMA2P30_SLP[SME] and CPM0_ER[DMA]. When sleep mode is enabled and the DMA controller becomes idle, the 10-bit idle timer begins counting down from the programmed value. Only the upper 5 bits of the idle counter are programmable; the lower 5 bits are hard coded to 0b11111. Therefore, the minimum granularity of the idle timer is 32 PLB clock cycles. When the counter reaches 0, the controller is placed in sleep mode.

Figure 22-15. DMA to PLB3 Sleep Mode Register (DMA2P30_SLP)

0:4	IDU	Idle Timer Upper 0–31	Upper 5-bits of the idle timer.
5:9	IDL	Idle Timer Lower Hard coded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the clock and power management logic to put the DMA controller to sleep.
11:31		Reserved	

22.2.6.3 DMA to PLB 3 Status Register (DMA2P30_SR)

As shown in *Figure 22-16*, DMA2P30_SR provides status information for each of the DMA channels. Bits in DMA2P30_SR are set in hardware, and can be either read or cleared by software. Clearing is performed by writing a word to DMA2P30_SR containing a 1 in any bit position to be cleared and 0 in all other bit positions.

The terminal count status (DMA2P30_SR[CSn]), end of transfer status (DMA2P30_SR[TSn]), and error status (DMA2P30_SR[RIn]) must be cleared for a DMA channel to operate. If a scatter/gather operation generates an interrupt for any of these conditions, the channel pauses until software clears any associated status fields in DMA2P30_SR.

Figure 22-16. DMA to PLB3 Status Register (DMA2P30_SR)			
0:3	CS[0:3]	Channel 0–3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0.
4:7	TS[0:3]	Channel 0–3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA2P30_CRn[ETD]=0.
8:11	RI[0:3]	Channel 0–3 Error Status 0 No error 1 Error occurred	See "Errors" on page -537 for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	
28:31		Reserved	

22.2.6.4 DMA to PLB 3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3)

DMA2P30_CR0–DMA2P30_CR3 are used to configure and enable their respective DMA channels. Before a DMA channel can transfer data, the channel control, source address, destination address, and transfer count registers must be programmed. If a DMA channel is programmed for scatter/gather transfers (DMA_SGC[SSGn]=1) the DMA channel control register is automatically loaded from memory. For additional details, see *Scatter/Gather Transfers* on page 522.

Figure 22-17. DMA to PLB3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3)			
0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts are generated for terminal count, end of transfer, and errors conditions. See *DMA Interrupts* on page 538.
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location 0 External peripheral (EBC) bus 1 OPB (UART0)	

4:5	PW	Peripheral Width/Memory alignment 00 Byte (8 bits) 01 Half word (16 bits) 10 Word (32 bits) 11 Double word (64 bits) memory-to-memory transfers only	Transfer width equals peripheral width for peripherals.
6	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address as shown in the next column	If set to 1 increment as follows: • 1, if the transfer width is a byte (8 bits) • 2, if the transfer width is a half word (16 bits) • 4, if the transfer width is a word (32 bits) • 8, if the transfer width is a double word (64 bit)
7	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address as shown in the next column	If set to 1 increment as follows: • 1, if the transfer width is a byte (8 bits) • 2, if the transfer width is a half word (16 bits) • 4, if the transfer width is a word (32 bits) • 8, if the transfer width is a double word (64 bit)
8	BEN	Buffer Enable 0 Disable DMA 32-byte buffer 1 Enable DMA 32-byte buffer	If BEN=0, discrete read and write operations occur for each data transfer.
9:10	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
11:12	PSC	Peripheral Setup Cycles 0–3	Number of PerClk cycles that the EBC peripheral bus is idle from the last peripheral bus transaction until DMAAckn is active. Used only for the peripheral side of peripheral mode transfers.
13:18	PWC	Peripheral Wait Cycles 0–63	DMAAckn remains active for PWC + 1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
19:21	PHC	Peripheral Hold Cycles 0–7	The number of PerClk cycles from the time DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only for the peripheral side of peripheral mode transfers.
22	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
23	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
24:25	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are prioritized by channel number; channel 0 has the highest priority. See *Channel Priorities* on page 536 for more information.
26:27	PF	Memory Read Prefetch Transfer 00 Prefetch 1 double word 01 Prefetch 2 double words 10 Prefetch 4 double words 11 Reserved	Used only during memory-to-peripheral and device-paced memory-to-memory transfers. To enable prefetching it is required that BEN=1.
28	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity checking for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers" on page -537.

29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30:31		Reserved	

22.2.6.5 DMA to PLB 3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3)

DMA2P30_SA0–DMA2P30_SA3 contain source addresses for memory-to-memory and memory-to-peripheral transfers. If a DMA channel is setup for scatter/gather transfers (DMA_SGC[SSGn]=1), the associated source address register is automatically loaded from memory. For additional details, see *Scatter/Gather Transfers* on page 538.

The source address must be aligned at the transfer width programmed in DMA2P30_CRn[TW]. Otherwise, the error bit (DMA2P30_SR[RIn]) is set for the channel and no transfer occurs. If the source address increment bit in the channel control register is set (DMA2P30_CRn[SAI]), the address is incremented by the transfer width after each data transfer. In contrast, if the channel is performing a memory-to-peripheral transfer and the address decrement bit is set (DMA2P30_CRn[DEC]=1), the address is decremented by the transfer width after each transfer.

Figure 22-18. DMA to PLB3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3)

0:31		Source address for memory-to-memory and memory-to-peripheral transfers.	

22.2.6.6 DMA to PLB 3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3)

DMA2P30_DA0–DMA2P30_DA3 contain the destination address for memory-to-memory and peripheral-to-memory transfers. When a DMA channel is configured for scatter/gather transfers (DMA_SGC[SSGn]=1), the destination address register is automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 538.

The destination address must be aligned at the transfer width programmed in DMA2P30_CRn[TW]. Otherwise, the error bit (DMA2P30_SR[RIn]) is set for the channel and no transfer occurs. If the destination address increment bit in the channel's control register is set (DMA2P30_CRn[DAI]) the address is incremented by the transfer width after each data transfer. However, if the channel is performing a peripheral-to-memory transfer and the address decrement bit is set (DMA2P30_CRn[DEC]=1), the destination address is decremented by the transfer width after each transfer.

Figure 22-19. DMA to PLB3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3)

0:31		Destination address for memory-to-memory and peripheral-to-memory transfers.	

22.2.6.7 DMA to PLB 3 Count Registers (DMA2P30_CT0–DMA2P30_CT3)

DMA2P30_CT0–DMA2P30_CT3 contain the number of transfers left in the DMA transaction for their respective channels when EOTn[TCn] is programmed as a terminal count output. When EOTn[TCn] is programmed as an end-of-transfer input (DMA2P30_CRn[ETD]=0), DMA2P30_CTn continues to count down past zero until EOTn is asserted.

If a DMA channel is setup for scatter/gather transfers (DMA_SGC[SSGn]=1), the count register is automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 538.

The value in the DMA count register is interpreted as the number of *transfers* of the width specified in DMA2P30_CRn[PW], *not* the total number of bytes. The maximum number of transfers is 64 K, and each transfer can be either 1, 2, 4, or 8 bytes as programmed in DMA2P30_CR[PW]. The maximum count of 64 K transfers is programmed by writing zero to DMA2P30_CTn.

Figure 22-20. DMA to PLB3 Count Registers (DMA2P30_CT0–DMA2P30_CT3)

0:15		Reserved	
16:31	NTR	Number of transfers remaining	

22.2.6.8 DMA to PLB 3 Scatter/Gather Descriptor Address Registers (DMA2P30_SG0–DMA2P30_SG3)

When a DMA channel is setup for scatter/gather transfers (DMA_SGC[SSGn]=1), the Scatter/Gather Descriptor Address Register (DMA2P30_SGn) contains the memory address of the next scatter/gather descriptor table. Prior to starting a scatter/gather transfer, software must write the address of the channel's descriptor table to DMA2P30_SGn. Once the scatter/gather transfer starts, DMA2P30_SGn is automatically updated from the descriptor table. For additional details see *Scatter/Gather Transfers* on page 538.

Figure 22-21. DMA to PLB3 Scatter/Gather Desc. Addr. Regs (DMA2P30_SG0–DMA2P30_SG3)

0:31		Address of next scatter/gather descriptor table.	

22.2.6.9 DMA to PLB 3 Scatter/Gather Command Register (DMA2P30_SGC)

DMA2P30_SGC[SSGn] are the start scatter/gather enable bits for channels 0 to 3. DMA2P30_SGC[EMn] are the corresponding enable mask bits for DMA2P30_SGC[SSGn]. Setting DMA2P30_SGC[EMn] = 1 causes the selected channel to begin a scatter/gather operation, while writing a 0 stops the scatter/gather operation. To start or stop a specific scatter/gather channel, the corresponding DMA2P30_SGC[EMn] bit must be set to 1; otherwise, DMA2P30_SGC holds the previous value. Note that halting a scatter/gather transfer does not stop the transfer currently in progress.

Upon completion of a scatter/gather sequence of transfers, the DMA controller clears DMA2P30_SGC[SSGn].

If an error occurs when the DMA controller is reading the scatter/gather descriptor table, DMA2P30_SGC[SSGn] is cleared for the affected channel, and the channel error status bit (DMA2P30_SR[RIn]) is set.

For additional details see *Scatter/Gather Transfers* on page 538.

Figure 22-22. DMA to PLB3 Scatter/Gather Command Register (DMA2P30_SGC)			
0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3. 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3. 0 Writes to SSG[n] are ignored 1 Allow writing to SSG[n]	To write SSG[n], EM[n] must be set. Otherwise, writing SSG[n] has no effect.
20:31		Reserved	

22.2.6.10 DMA to PLB 3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3)

The DMA Sub Channel ID is a 32-bit register that contains the 3-bit ID in bits 0–2. These bits are used to allow one DMA channel to support up to eight different peripheral devices. This register's contents are driven out of the core as primary outputs (O_DMA_subchnl_ID0:O_DMAsubchnl_ID3). External multiplexing uses these output signals to select the request and EOT signals from the active device. The value of this register can also be loaded during scatter/gather operations.

Figure 22-23. DMA to PLB3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3)			
0:2	SCID	DMA Subchannel ID	
3:31		Reserved	

22.2.6.11 DMA to PLB 3 Address Decode Register (DMA2P30_ADR)

OPB addresses are decoded by the DMA by comparing the most significant nibble of the source or destination address to the contents of the DMA2P30_ADR register. The DMA2P30_ADR register is written by a move to DCR operation. Upon reset, its value is set according to the I_DMA_PGM_dmaadr pins, which are tied high or low by the system designer upon instantiation of the DMA in a system. The 8-bit address compare decode field is implemented in bits 0–7 of the 32-bit register.

Figure 22-24. DMA to PLB3 Address Decode Register (DMA2P30_ADR)			
0:7	ADRD	Address decode compare field	
8:31		Reserved	

22.2.7 Channel Priorities

The priority of DMA transfers is controlled on a per-channel basis by the channel priority field in the channel control register. *Table 22-9* shows the different priority settings for DMA2P30_CRn[PW].

Table 22-9. DMA to PLB 3 Transfer Priorities

DMA2P30_CRn[CP]	Priority Level
0b00	Low
0b01	Medium Low

Table 22-9. DMA to PLB 3 Transfer Priorities

DMA2P30_CRn[CP]	Priority Level
0b10	Medium High
0b11	High

These priorities serve two purposes. First, the DMA controller arbitrates among all actively requesting channels and selects the highest priority channel for service. If multiple channels request at the same priority, the arbiter selects the lowest numbered channel for service.

DMA2P30_CRn[CP] determines the priority of the internal PLB transactions that the DMA controller uses to read and write data.

22.2.8 Data Parity During DMA Peripheral Transfers

When DMA2P30_CRn[PCE] = 1, parity checking is enabled for peripheral mode transfers on channel n.

Note: The peripheral bus controller (EBC) does not support data parity checking.

22.2.9 Errors

The DMA controller detects and reports three types of errors: address alignment, PLB timeout and slave errors. The DMA controller reports errors through the channel error status bit in the DMA status register (DMA2P30_SR[RIn]). If the error status bit for a channel is set, the channel enable bit (DMA2P30_CRn[CE]) is cleared, disabling the channel. An interrupt signal is also presented to the interrupt controller if DMA2P30_CRn[CIE] is set. See *DMA Interrupts* on page 538 for more information on interrupt processing.

When the DMA controller has multiple channels active, an error may be reported on the current channel which was in actuality caused by a previously active channel. This causes the current channel to have its error status bit set. Therefore, for deterministic error analysis with multiple DMA channels active the PLB slave bus controller's error status registers (the bus error address register in particular), must be queried to isolate the actual channel which encountered the error. In any case, the channel causing the errors will eventually cause all active channels, including itself, to be disabled.

22.2.10 Address Alignment Error

The source address (DMA2P30_SAn) and destination address (DMA2P30_DAn) registers must be aligned to the programmed transfer width (DMA2P30_CRn[TW]).The address alignment rules are outlined in *Table 22-10*. In addition, when a channel is configured for scatter/gather transfers, the scatter/gather table must be word-aligned. If the source, destination and scatter/gather address registers are not appropriately aligned an error occurs immediately after the channel is enabled.

Table 22-10. Address Alignment Requirements

DMA2P30_CRn[PW] Setting	Required Alignment for DMA2P30_SAn and DMA2P30_DAn
0b00	Byte (8-bit)
0b01	Half word (16-bit)
0b10	Word (32-bit)
0b11	Double word (64-bit)

22.2.11 PLB Timeout

The DMA controller uses PLB operations to read and write memory. A PLB timeout results if the DMA controller attempts to access a non-existent memory location. This will occur if the source, destination or scatter/gather address registers do not map to valid memory locations.

22.2.12 Slave Errors

If the DMA controller detects an error from a PLB slave, it finishes any active read/write pair transfer on the channel and then reports an error. An SDRAM uncorrectable ECC error and an EBC bank protection error are examples of PLB slave errors.

22.2.13 DMA Interrupts

Each DMA channel can generate interrupts for end of transfer, terminal count and error conditions. Interrupts from a particular DMA channel are enabled by setting the channel enable bit in the channel's control register (DMA2P30_CRn[CIE]=1). When an interrupt occurs for a given channel, the DMA controller sends a signal to the Universal Interrupt Controller. For the PPC440EP's CPU to take an exception, interrupts from the particular DMA channel must be enabled in the interrupt controller's interrupt enable register (UIC0_ER). Also, the CPU's machine state register's interrupt enable bit must be enabled for the appropriate interrupt type (critical or non-critical), MSR[EE,CE]. See *Universal Interrupt Controller* on page 223 for more information on interrupt controller processing.

For DMA channels with interrupts enabled (DMA2P30_CRn[CIE]=1) and not performing a scatter/gather transfer an interrupt is generated while any of the following are true:

DMA2P30_CRn[TCE]=1 and DMA2P30_SR[CSn]=1
DMA2P30_SR[TSn]=1
DMA2P30_SR[RIn]=1

When a channel is performing a scatter/gather transfer, interrupt generation is further qualified by the TCI, ETI and ERI bits loaded from the descriptor table (see *Table 22-11*). Any of the following conditions cause an interrupt during a scatter/gather transfer when interrupts are enabled for the channel (DMA2P30_CRn[CIE=1]):

TCI=1 and DMA2P30_CRn[TCE]=1 and DMA2P30_SR[CSn]=1
ETI=1 and DMA2P30_SR[TSn]=1
ERI=1 and DMA2P30_SR[RIn]=1

For both normal DMA and scatter/gather transfers the interrupt remains active until the appropriate bits are cleared in the DMA Status Register (DMA2P30_SR). In addition, interrupts from a channel performing a scatter/gather transfer cause the channel to pause until the interrupt is cleared.

22.2.14 Scatter/Gather Transfers

With a normal DMA transfer it is necessary to program a channel's control, source, destination, and count registers for each transfer. The scatter/gather capability of the DMA controller provides a more efficient solution for applications that require multiple transactions on a single DMA channel. Instead of individually programming a channel's registers, software creates a set (linked list) of descriptor tables in system memory. *Table 22-11* illustrates the required table format.

Table 22-11. Scatter/Gather Descriptor Table

Memory Address	Byte 0 (MSB)	Byte 1	Byte 2	Byte 3 (LSB)
x (word aligned)	DMA Channel Control Word			

Table 22-11. Scatter/Gather Descriptor Table

Memory Address	Byte 0 (MSB)							Byte 1	Byte 2	Byte 3 (LSB)
x + 4	Source Address									
x + 8	Destination Address									
x + 12	LK		TCI	ETI	ERI				Count	
x + 16	Next Scatter/Gather Descriptor Table Address									

Table 22-12 details the usage of the bit fields in the scatter/gather table.

Table 22-12. Bit Fields in the Scatter/Gather Descriptor Table

Bit	Mnemonic	Description
0	LK	Link 0 This is the last descriptor. 1 Fetch next descriptor from address DMA2P30_SGn when the channel completes
2	TCI	Enable Terminal Count Interrupt 0 Do not interrupt when terminal count occurs 1 Allow an interrupt when terminal count occurs
3	ETI	Enable End of Transfer Interrupt 0 Do not interrupt when end of transfer occurs 1 Allow an interrupt when end of transfer occurs.
4	ERI	Enable Error Interrupt 0 Do not interrupt if an error occurs 1 Allow an interrupt if an error occurs

To configure a channel for a scatter/gather transfer the DMA Scatter/Gather Descriptor Address Register (DMA2P30_SGn) for the channel is set to the address of the first descriptor table, which must be word-aligned. To begin the scatter/gather transfer, software then writes a start scatter/gather and enable mask to the Scatter/Gather Command Register (DMA2P30_SGC). The DMA controller then reads the descriptor table at address DMA2P30_SGn and updates the DMA controller registers as shown in *Table 22-13*. Upon receiving the data from the scatter/gather descriptor table, the channel's terminal count status bit (DMA2P30_SR[TCn]) is automatically cleared.

Table 22-13. DMA Registers Loaded from Scatter/Gather Descriptor Table

Descriptor Table Entry	Register Loaded
Channel Control Word	DMA2P30_CRn
Source Address	DMA2P30_SAn
Destination Address	DMA2P30_DAn
Count	DMA2P30_CTn
Next Descriptor Address	DMA2P30_SGn

After loading the channel's registers from the descriptor table, the transfer functions as a normal non-scatter/gather operation.

If the channel control word loaded from the descriptor table enables interrupts for the channel (DMA2P30_CRn[CIE]=1), the TCI, ETI, and ERI bits further qualify the generation of interrupts. See *DMA Interrupts* on page 538 for more information on scatter/gather interrupts.

If the LK (link) bit was not set the scatter/gather process stops when the current transfer completes. Otherwise, the DMA controller reads the descriptor table at address DMA2P30_SGn and the process repeats.

22.2.15 Programming the DMA to PLB3 Controller

Before the DMA controller can transfer data it must be configured, both globally and on a per-channel basis. Global registers include DMA2P30_POL and DMA2P30_SLP. For most applications, these registers should be configured when the DMA controller is initialized. To prevent spurious activity resulting from changing the active level for DMAReqn, DMAAckn, or EOTn[TCn], a channel configuration in tDMA2P30_POL should not be altered when the channel is enabled (DMA2P30_CRn[CE]=1).

The channel registers are DMA2P30_CRn, DMA2P30_SAn, DMA2P30_DAn, DMA2P30_CTn, and DMA2P30_SGn. The type of DMA transfer determines which of these registers must be programmed and what causes the channel to start. In all cases,DMA2P30_SR[CSn,TSn, RIn] must be cleared, or the channel will not start.

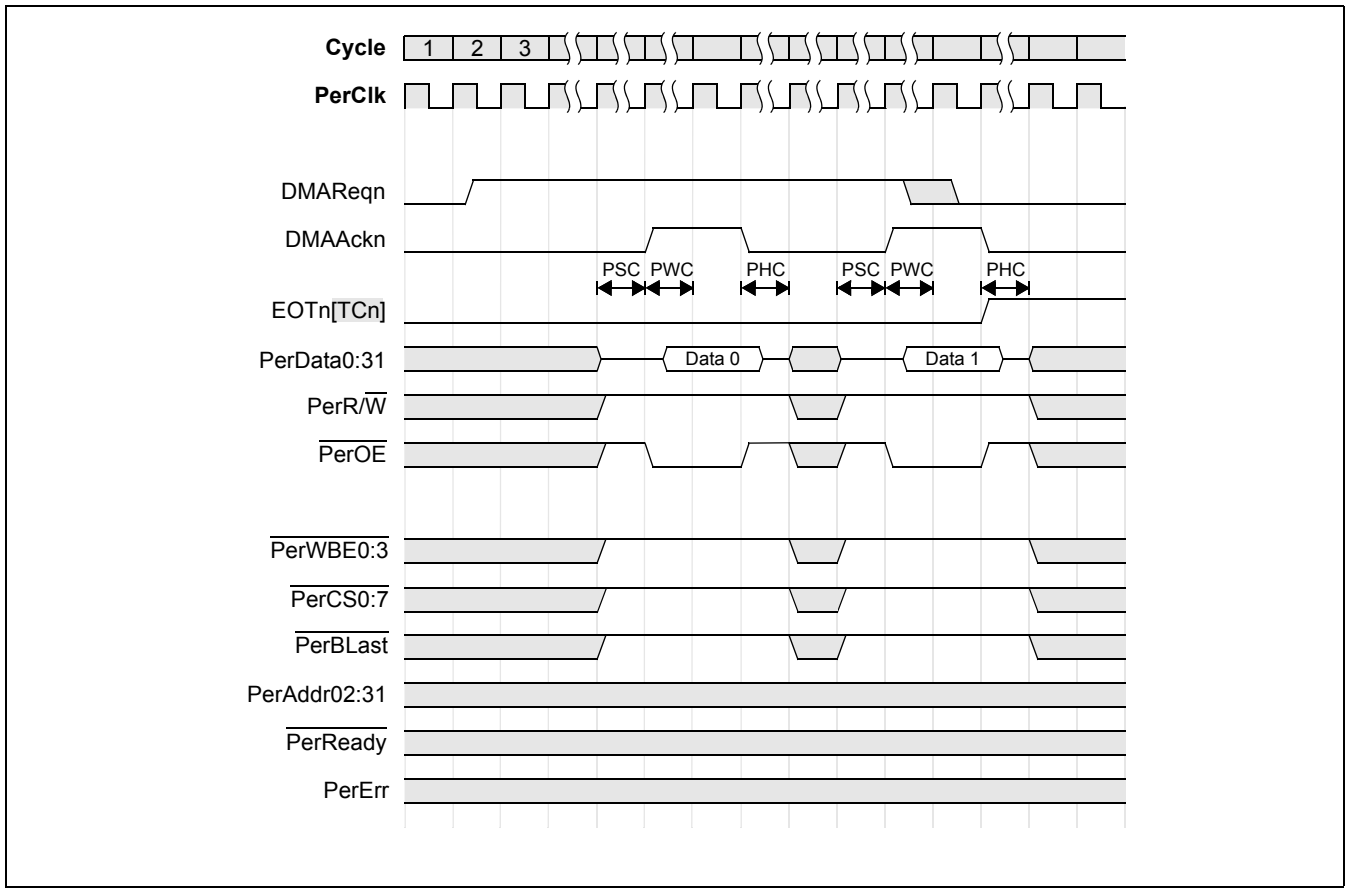
The programming information that follows assumes that the DMA controller is operating in non-scatter/gather mode. For scatter/gather transfers, the channel configuration data must be written into a set of descriptor tables in system memory, as described in *Scatter/Gather Transfers* on page 538.

22.2.16 Peripheral Mode Transfers

DMA peripherals are either devices attached to the EBC interface using the DMAReqn and DMAAckn lines, or the internal serial port (UART0). During a peripheral mode transfer an external peripheral asserts DMAReqn to request a DMA transfer. For metastability protection, DMAReqn is double-latched in the DMA on assertion, and sampled with a single latch on deassertion.

Timings on the memory-access portion of peripheral mode DMA transfers are governed by the configuration of the associated memory controller. In contrast, timing during the peripheral portion of the transfer is controlled by DMA2P30_CRn[PSC, PWC, PHC]. The effect of these parameters on peripheral timings is illustrated in *Figure 22-25* and *Figure 22-26*. Although shown as active high, the polarity (active state) of DMAReqn, DMAAckn, and EOTn[TCn] are programmable using DMA2P30_POL .

Figure 22-25. Peripheral to Memory DMA Transfers



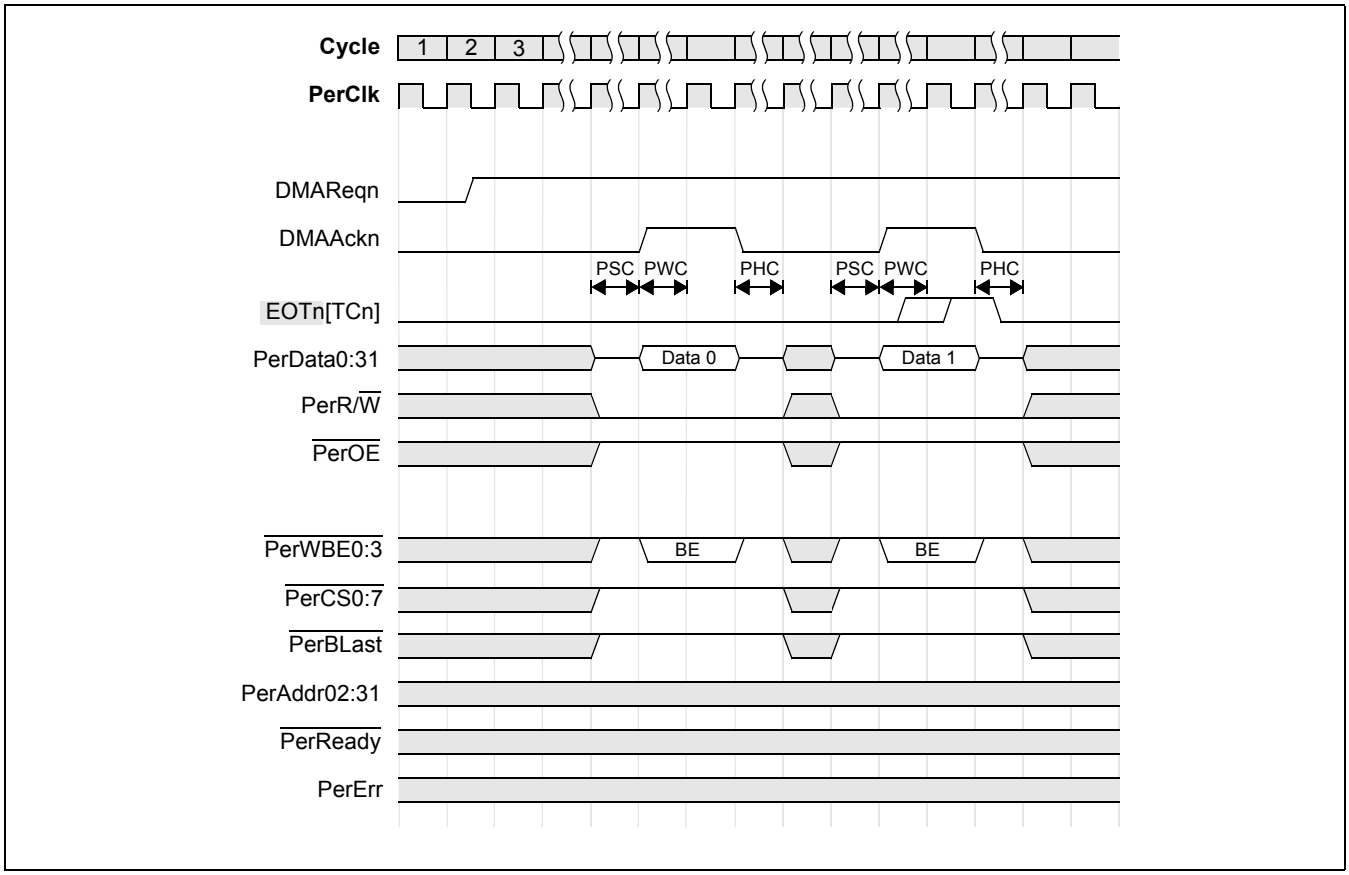
Peripheral setup cycles (DMA2P30_CRn[PSC]) provide a delay between any previous operation on the peripheral bus and DMAAckn becoming active. Following the setup time DMAAckn is driven active for (DMA2P30_CRn[PWC] + 1) PerClk cycles, until the rising edge of PerClk immediately before DMAAckn goes inactive. During peripheral-to-memory transfers read data from the peripheral is sampled on the rising PerClk edge where DMAAckn goes inactive. After DMAAckn becomes inactive the peripheral bus is held idle for DMA2P30_CRn[PHC] PerClk cycles.

The second transfer in *Figure 22-25* illustrates the required DMAReqn timing to prevent a subsequent DMA transfer. For all peripheral mode transfers DMAReqn must be sampled inactive at the end of the last PerClk cycle where DMAAckn is active.

The EOTn[TCn] I/O can be configured either as an end of transfer input (DMA2P30_CRn[ETD]=0) or a terminal count output (DMA2P30_CRn[ETD]=1). When programmed as a terminal count output, EOTn[TCn] is asserted in the cycle after DMAAckn became inactive and the channel's count register (DMA2P30_CTn) reached zero. EOTn[TCn] remains active until the terminal count status bit is cleared in the DMA status register (DMA2P30_SR[CSn]).

If EOTn[TCn] is configured as an end of transfer input (DMA2P30_CRn[ETD]=0), EOTn[TCn] must be sampled active during the last DMAAckn cycle. If the channel is configured for scatter/gather transfers EOTn[TCn] should be immediately deasserted to prevent the subsequent transfer from ending prematurely. *Figure 22-26* shows the required timing.

Figure 22-26. Memory to Peripheral DMA Transfers



For both peripheral-to-memory and memory-to-peripheral transfers the transfer width (DMA2P30_CR[PW]) must be set to the data bus width of the peripheral. This is because the DMA controller does not pack or unpack data on the peripheral side of transaction.

22.2.16.1 Peripheral-to-Memory Transfer

To perform a peripheral-to-memory DMA transfer from an EBC-attached DMA peripheral:

1. Set destination address register (DMA2P30_DAn) to the desired memory location. The address must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
2. Program the count register (DMA2P30_CTn) for the number of transfers.
3. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
4. In the channel control register (DMA2P30_CRn):
   a. Optionally enable the DMA buffer, BEN=1.
   b. Optionally enable parity checking, PCE=1.
   c. Set the destination address increment, DAI=1.
   d. Set the transfer mode to peripheral, TM=0b00.

e. Set the peripheral location to external, PL=0.
f. Set the transfer direction to peripheral-to-memory, TD=1.
g. Enable the channel CE=1.

Once the DMA channel is active, the peripheral initiates a transfer by activating the DMAReqn pin for the channel. The PPC440EP then activates the DMAAckn pin to read data from the peripheral. This continues until either a terminal count or end of transfer condition occurs.

22.2.16.2 Memory-to-Peripheral Transfer

To perform a memory-to-peripheral DMA transfer to an EBC-attached DMA peripheral:

1. Set source address register (DMA2P30_SAn) to the desired memory location. The address must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
2. Program the count register (DMA2P30_CTn) for the number of transfers.
3. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
4. In the channel control register (DMA2P30_CRn):
   a. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
   b. Optionally enable parity generation, PCE=1.
   c. Set the source address increment, SAI=1.
   d. Set the transfer mode to peripheral, TM=0b00.
   e. Set the peripheral location to external, PL=0.
   f. Set the transfer direction to memory-to-peripheral TD=0.
   g. Enable the channel, CE=1.

Once the DMA channel is active, the peripheral initiates a transfer by activating the DMAReqn pin for the channel. The PPC440EP then reads the source memory and subsequently activates the DMAAckn pin to write data to the peripheral. This continues until either a terminal count or end of transfer condition occurs.

22.2.17 Memory-to-Memory Transfers

Memory-to-memory transfers can be initiated either by software or by an external device. If initiated via software, the transfer begins as soon as the channel is configured and enabled. When initiated by hardware (also known as a device-paced memory-to-memory transfer), software configures the channel for a memory-to-memory move and transfers begin when an external device places an active request on the channel request line, DMAReqn.

22.2.17.1 Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers

To perform a device-paced memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P30_CRn[PW]) to the width of the device-paced memory.
2. Set the source (DMA2P30_SAn) and destination (DMA2P30_DAn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
3. Program the count register (DMA2P30_CTn) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
5. In the channel control register (DMA2P30_CRn):

a. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
b. If the device-paced memory is at the source memory location set PL=1.
c. Set the source address increment, SAI, and destination address increment, DAI, as desired.
d. Set the transfer mode to device-paced memory-to-memory, TM=0b11.
e. Enable the channel, CE=1.

Once the DMA channel is configured for this mode, the external device initiates a transfer by activating the DMAReqn input. The PPC440EP then reads the source memory, buffers the data in the DMA controller and then outputs the data to the destination memory address. Transfers continue as long as the controlling device maintains an active signal on DMAReqn and the channel count register (DMA2P30_CTn) is non-zero. To pause a device paced memory-to-memory transfer, the controlling device must deassert DMAReqn one PerClk cycle before the last cycle in the device-paced memory access.

22.2.17.2 Software-Initiated Memory-to-Memory Transfers (Non-Device Paced)

To perform a software-initiated memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P30_CRn[PW]) as desired.
2. Set the source (DMA2P30_SAn) and destination (DMA2P30_DAn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
3. Program the count register (DMA2P30_CTn) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
5. In the channel control register (DMA2P30_CRn):
   a. Optionally enable the DMA buffer, BEN=1.
   b. Set the source address increment, SAI, and destination address increment, DAI, as desired.
   c. Set the transfer mode to software-initiated memory-to-memory, TM=0b10.
   d. Enable the channel, CE=1.

Once the channel is enable the DMA controller transfers data from source to destination until the channel count reaches zero. Note that memory-to-memory transfers initiated by software do not use DMAReqn or DMAAckn.

23. Memory Access Layer

The Memory Access Layer (MAL) is a hardware core that manages data transfers between packet-oriented communications cores, known as COMMACs (communications media access controllers), and memory. The PPC440EP implements two Ethernet MACs that are managed by a single MAL. Only one MAL is required to handle all the data channels of the EMACs. Please note that COMMAC is a generic term used to refer to the EMACs throughout the chapter.

To communicate with software device drivers, the MAL utilizes a buffer descriptor ring structure in memory. A software device driver uses the buffer descriptor structure to inform the MAL about buffer locations and packet or buffer status. The MAL uses the buffer descriptors to convey packet transfer status from the COMMAC core back to the software device driver. Each MAL channel requires its own buffer descriptor table ring structure in memory.

The PPC440EP MAL manages the transfer of packets between the two Ethernet Media Access Controllers (EMAC0 and EMAC1) and the memory attached to the PPC440EP (SDRAM or SRAM). The primary function of MAL is to move packets directly between memory and a COMMAC core with minimal involvement of the processor core.

23.1 MAL Features

- No restrictions on buffer alignment
- Aligned bus accesses to enable burst operation with external memories
- Configurable receive buffer size (configurable per channel)
- No minimum transmit buffer size
- Maximum buffer sizes of 4095 bytes (TX) and 4080 bytes (RX)
- Up to 256 descriptors in the buffer descriptor table per channel
- Configures COMMAC according to commands specified in the descriptor status/control field
- Updates the descriptor status/control field at the end of packet transfer according to the status received from COMMAC
- Buffer-based interrupt capabilities for each channel
- Concurrent operation of RX and TX channels
- Configuration using Device Control Registers (DCRs)
- Programmable PLB arbitration priority
- PLB/OPB error detection

Refer to the block diagram in the *Power PC 440EP Embedded Processor Data Sheet* for a general system structure overview of an embedded PowerPC processor core integrated with a packet oriented communication core.

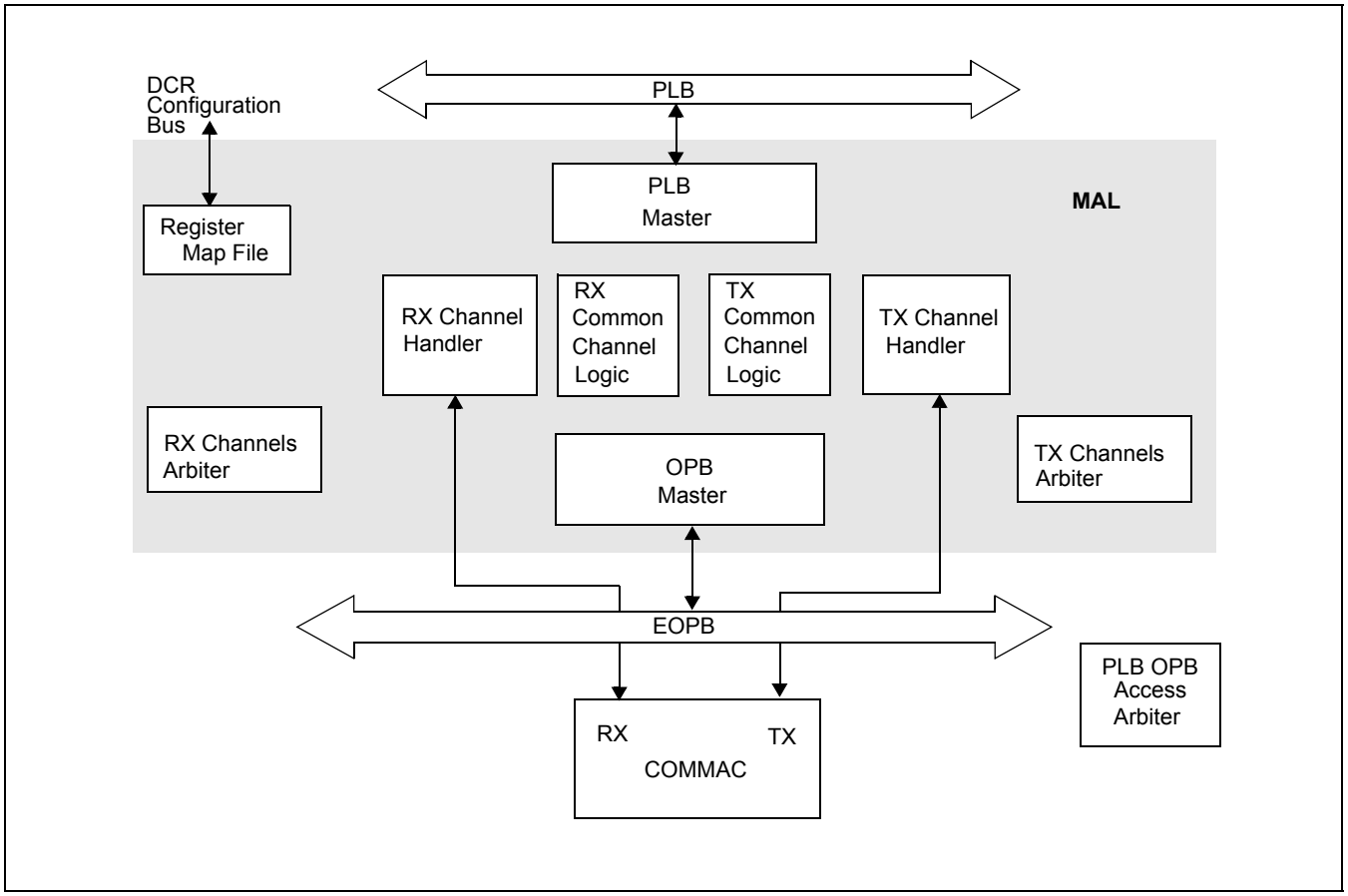
Two COMMACs are configured and controlled by the processor core using the OPB without MAL intervention. Packet data to be transmitted and received are stored in buffers in external memory. The MAL processes buffer descriptors and provides all data access facilities to the COMMACs.

The MAL is not aware of COMMACs such as EMAC as an entity. It is only aware of the COMMAC's channels. In the PPC440EP, each EMAC contains two TX channels and one RX channel. Transmit and receive operations can be performed simultaneously by the MAL (full duplex). When a channel wins arbitration for the EOPB, the MAL transfers data between system memory and the COMMAC. The MAL and the software driver maintain separate, dedicated buffer descriptor tables for each channel to maintain channel, packet, and buffer status. Packets can be constructed from one data buffer, or several data buffers (known as buffer chaining).

23.1.1 MAL Internal Structure

Figure 23-1 illustrates the MAL internal structure. Brief descriptions of the MAL components follow.

Figure 23-1. MAL Internal Structure



23.1.1.1 PLB Master

The PLB Master logic performs PLB transactions for the MAL, and is used to transfer data between a COMMAC and memory, fetch buffer descriptors, and communicate status regarding data transfer.

23.1.1.2 OPB Master

The OPB Master logic performs EOPB transactions for the MAL, and is used to transfer data between a COMMAC and memory.

23.1.1.3 TX Channel Handler

The TX channel handler has a dedicated section for each TX channel. It keeps a record of the descriptor information and the current state of each channel.

23.1.1.4 RX Channel Handler

The RX channel handler has a dedicated section for each RX channel. It keeps a record of the descriptor information and the current state of each channel.

23.1.1.5 TX Channel Arbiter

The TX channel arbiter, connected to request lines from each TX channel, arbitrates among the TX channels and decides which channel gains access to the TX common channel logic.

23.1.1.6 RX Channel Arbiter

The RX channel arbiter, connected to request lines from each RX channel, arbitrates between the RX channels and decides which channel gains access to the RX common channel logic.

23.1.1.7 TX Common Channel Logic

The TX common channel logic is shared by all TX channels. It services a single TX channel at a time (selected by the TX arbiter). This logic activates the PLB and OPB masters for data and buffer descriptor transactions.

23.1.1.8 RX Common Channel Logic

The RX common channel logic is shared by all RX channels. It services a single RX channel at a time (selected by the RX arbiter). This logic activates the PLB and OPB masters for data and buffer descriptor transactions.

23.1.1.9 Register Map File

The register map file is used to configure MAL and read its status registers. Software accesses the MAL register file using the **mtdcr** and **mfdcr** instructions.

23.2 MAL0 Interfaces and Channel Assignments

MAL0 consists of 6 channels (4 transmit channels and 2 receive channels). Each channel is dedicated to one of two EMAC cores. See *Table 23-1* for MAL0 channel assignments.

In the PPC440EP, MAL0 uses 1/1 clocking, that is, the interface between MAL0 and the EMAC cores is clocked at the same frequency as the interface between MAL0 and the PLB.

Table 23-1. MAL0 Channel Assignment

MAL0 Channel	EMAC Channel
RX Channel 0	EMAC0 RX Channel 0
RX Channel 1	EMAC1 RX Channel 1
TX Channel 0	EMAC0 TX Channel 0
TX Channel 1	EMAC0 TX Channel 1
TX Channel 2	EMAC1 TX Channel 0
TX Channel 3	EMAC1 TX Channel 1

Sideband signals control slave selection and data transfer framing.

23.3 Transmit and Receive Operations

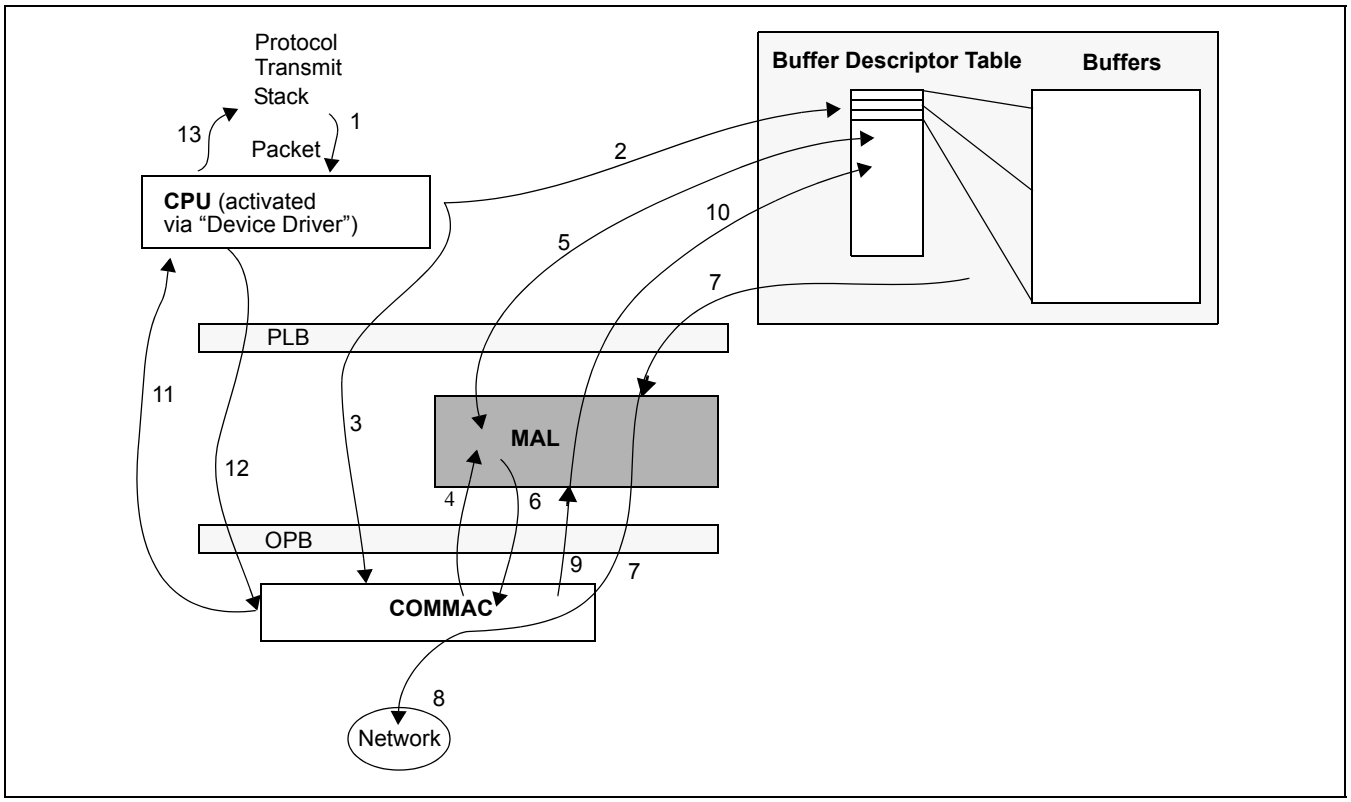
The device driver is responsible for configuring the MAL before a COMMAC can begin requesting the MAL to process packets of data. The device driver should ensure that channels are not enabled during reconfiguration; otherwise, fatal errors may occur.

For more information about the MAL software interface, see *MAL Programming Notes* on page 559.

23.3.1 Transmit Operations

Figure 23-2 describes the software and hardware operations involved in a typical transmit operation.

Figure 23-2. Transmit Operations



Following are descriptions for the numbered process steps in the figure:

1. The protocol stack (high-level software layer) initiates a packet transmit.
2. Software device driver parses the protocol stack buffer into descriptor table entries and buffers.
3. Software device driver instructs the COMMAC to process a new transmit packet.
4. The COMMAC channel requests MAL to process a new packet.
5. MAL fetches descriptor information.
6. MAL writes control information into the COMMAC and initiates the data move.
7. Packet data is transferred from memory into the COMMAC (the COMMAC controls the pace of the data transfer).
8. The packet is transmitted on the media (steps 7 and 8 can overlap).
9. The COMMAC requests that MAL read the packet status.

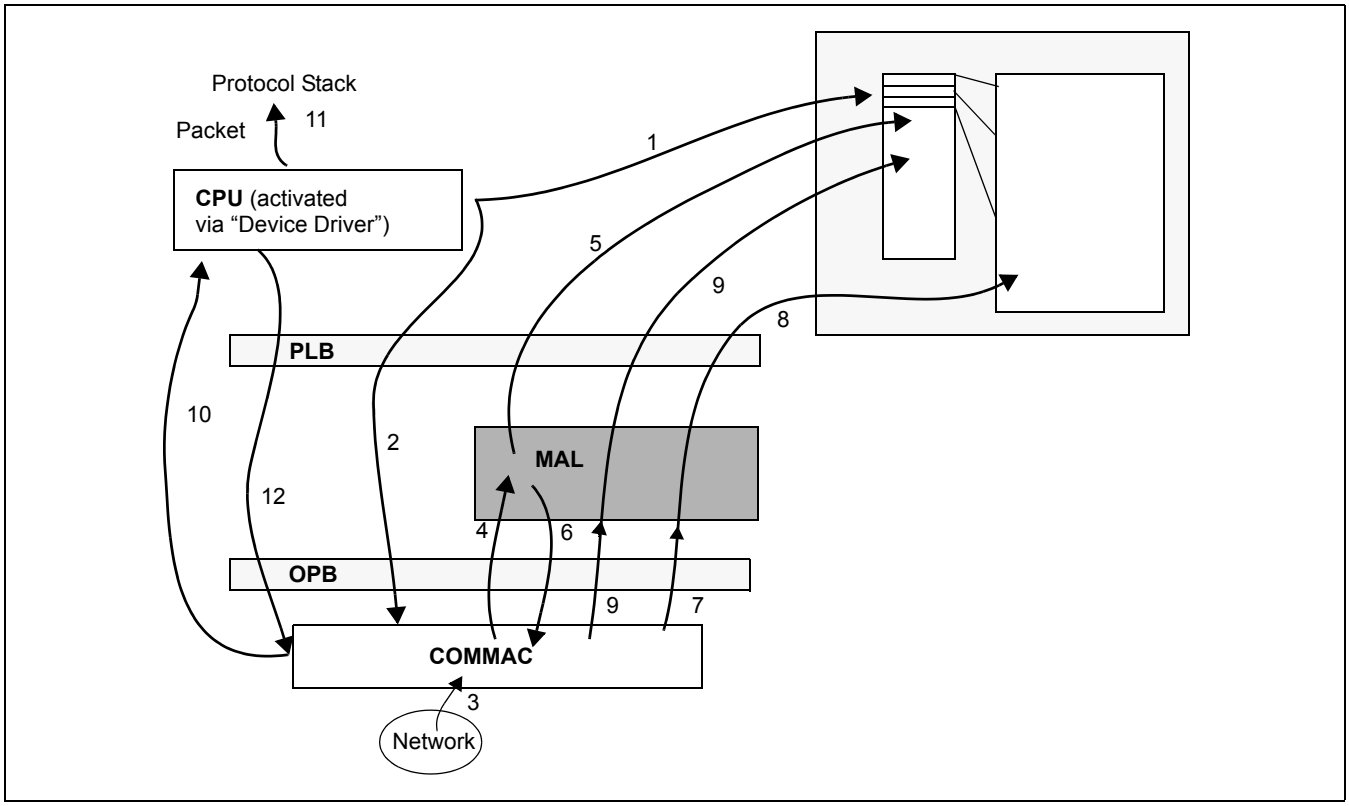
10. The status read by MAL is written back into a buffer descriptor.
11. Software is interrupted (if interrupt conditions are met) by the COMMAC or by the MAL end-of-buffer interrupt.
12. Software clears the interrupt status bits in the COMMAC and in MAL.
13. Software informs the protocol stack that transmission is complete. (Typically, the stack assumes the packet will go once after step 1. In step 13 the software may return the buffer to the stack)

Note: The description in *Figure 23-2* is the general scheme for MAL transmit operation in the software environment. The device driver should follow recommendations from *MAL Programming Notes* on page 559.

23.3.2 Receive Operations

Figure 23-3 describes the software and hardware operations when receiving a typical packet.

Figure 23-3. Receive Operations



Following are descriptions for the numbered process steps in the figure:

1. Software device driver initializes the receive buffer descriptors.
2. Software device driver enables the COMMAC to process a new packet.
3. A packet is received from the network (steps 2 and 3 can change in order).
4. The COMMAC channel requests that MAL process a new packet.
5. MAL fetches receive buffer information from the descriptor table.
6. MAL writes the control word from the descriptor to the COMMAC and initiates the data transfer.
7. The COMMAC channel fills its FIFO storage.
8. MAL stores the packet in system memory buffers pointed to by the descriptors.

9. MAL reads status information from the COMMAC and writes it to the buffer descriptors.
10. Software device driver is interrupted (if interrupt conditions are met) by the COMMAC or by the MAL end-of-buffer interrupt.
11. The receive packet is passed to the protocol stack.
12. Software updates the receive buffer descriptor positions allowing them to be used again.

Note: The description in *Figure 23-3* is the general scheme for MAL receive operation in the software environment. The device driver should follow recommendations from *MAL Programming Notes* on page 559.

23.4 Buffer Descriptor

The software interface for buffer descriptor (BD) processing consists of a set of registers within the MAL and a set of circular queues in memory. Each transmit and receive COMMAC channel has a descriptor table that contains buffer location and status information allocated to the channel. Each individual transmit or receive channel has its own buffer descriptor table. They are managed independently of each other. This section describes the individual transmit and receive interfaces.

Note: Because the MAL uses a flat addressing scheme on the PLB, the physical memory can be located anywhere on the PLB address space. However, each data structure that the MAL works with (data buffers and buffer description tables) must not cross a 4GB address border.

During its operation, the MAL is able to modify the contents of memory directly without processor core knowledge. Data cache coherency is the responsibility of the software device driver. To simplify device driver software, the MAL buffer descriptor tables should be placed in non-cached memory when possible. If this is not possible, the software driver must maintain cache coherency of the buffer descriptor tables by performing data cache flushes or invalidates when appropriate. When descriptors are in cached memory, the driver software must be aware that multiple descriptors are present in a single cache line and that cache invalidate or flush operations will be performed on multiple descriptors at the same time. This is significant because a cache line flush done by the driver to force a descriptor from the data cache to physical memory could corrupt another descriptor that occupies the same data cache line and is simultaneously being updated in physical memory by the MAL.

Data buffers, in contrast, should be placed in cachable memory if possible. The software driver can easily maintain cache coherency of data buffers if:

- All buffers are aligned on a data cache line boundary
- All buffers are a multiple of a data cache line in size

Note: The data cache line size and alignment in the PPC440EP is 32 bytes.

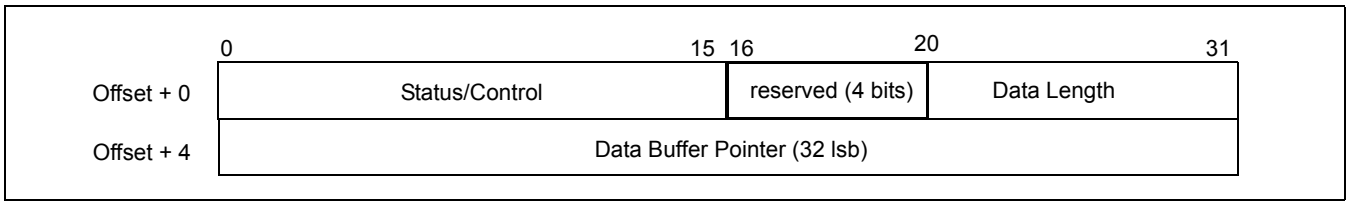
Before setting the empty bit in a RX buffer descriptor, the software driver must invalidate the memory occupied by the buffer in the data cache for the length specified in the RX buffer size register. Before transmitting a packet the software driver must flush the data buffer from the data cache before setting the Ready bit in the TX buffer descriptor. The software device driver or protocol stack fills the buffers pointed to by transmit buffer descriptors with packets to be transmitted, and/or provides empty buffers pointed to by receive buffer descriptors to be filled with received packets. Meanwhile, the hardware processes the descriptors, transfers the packet data to/from the COMMAC, and updates the status fields of the descriptors.

Packet data associated with each transmit or receive channel is stored in buffers. Each buffer has an entry dedicated to it in one of the channel's buffer descriptor tables. The MAL has a Channel Table Pointer Register for each of its channels. The COMMAC (EMAC in the PPC440EP) device driver sets the contents of these registers to point to the starting address of the buffer descriptor table for the associated channel.

The buffer descriptor table forms a circular queue with a programmable length. The last descriptor in the table is defined by setting the Wrap bit in the status/control field (see *Status/Control Field Format* on page 557). If there is no Wrap bit set in the table, then the MAL automatically wraps after processing 256 descriptors (the maximum number of descriptors allow per channel).

The format of the buffer descriptor (BD) (see *Figure 23-4*) is the same for all COMMACs, and has the same structure for both transmit and receive. The most significant halfword in each buffer descriptor contains a status/control halfword. This field contains two parts: the first part (6 bits) is BD handling information used by the MAL for descriptor processing, the second field (10 bits) is content specific for each COMMAC. The second halfword determines the data length (in bytes) referenced in this buffer descriptor. The second word in the buffer descriptor contains the 32-bit data buffer pointer that points to the actual data buffer in memory. It is suggested that each data buffer start on a cache line boundary and be a multiple of a cache line in size if it resides in cachable memory. (The cache line size in the PPC440EP processor core is 32 bytes.)

Figure 23-4. Buffer Descriptor Structure

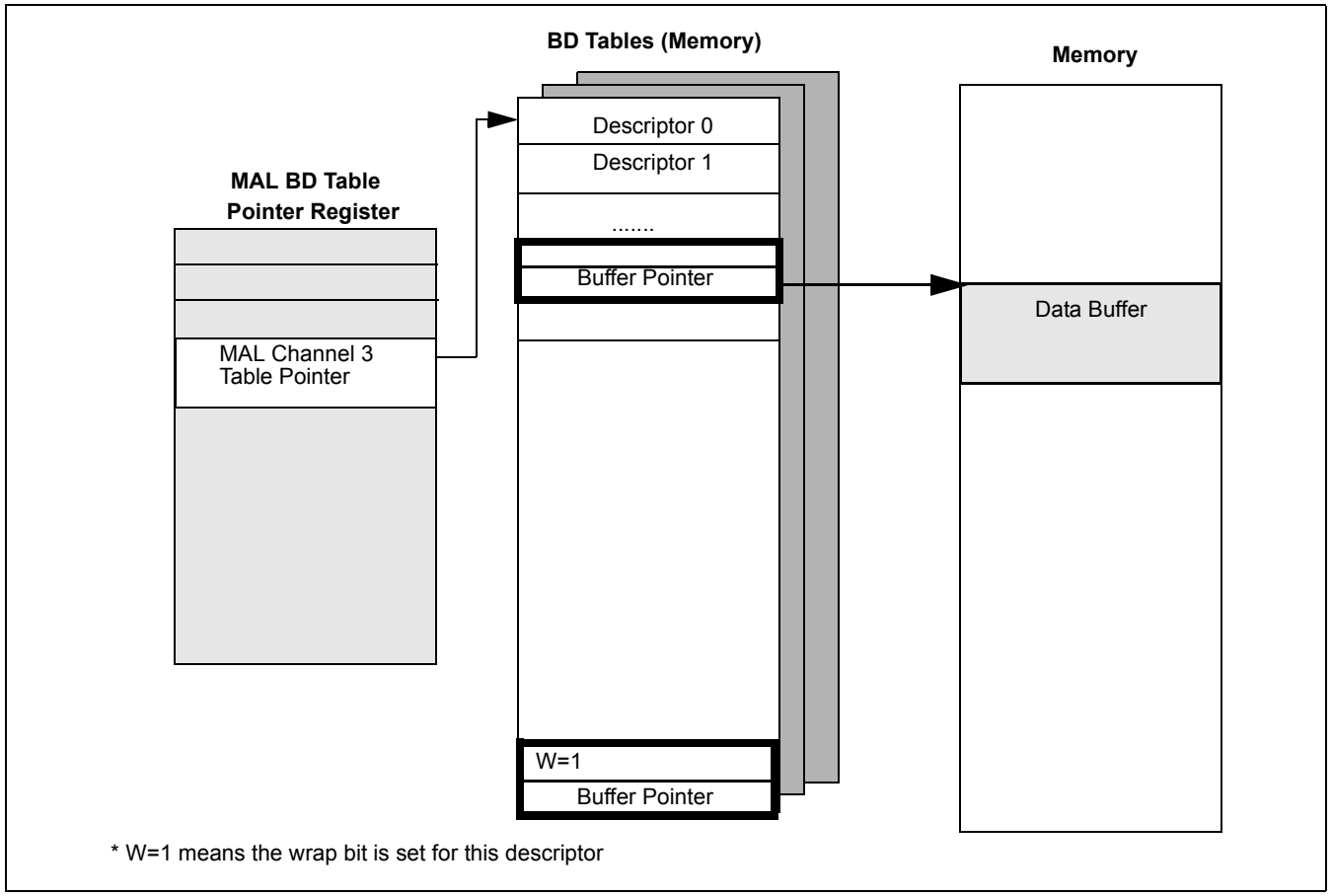


A packet can reside in as many buffers as necessary (transmit or receive). Each buffer has a maximum length of (4KB – 16) bytes. In TX channels, the buffer descriptor length field is written by the device driver and defines the number of bytes in the data buffer that is identified by the data buffer pointer. In RX channels, the buffer descriptor length field is written by the MAL and defines the number of bytes written by the MAL to the buffer that is identified by the data buffer pointer (see *Receive Software Interface* on page 555).

When processing a packet, the MAL does not require that all buffers of the current packet are already valid. It requires the buffers to be ready in due time to be transmitted or received. Failure of the software to provide the descriptors in time may result in an error.

Figure 23-5 describes the structure of the packet in memory.

Figure 23-5. Packet Memory Structure



23.5 Transmit Software Interface

Once a channel is enabled in the MAL (this is done by setting the appropriate bit in the Channel Active Register), a channel can request service from the MAL. When the first transmit request comes in from a COMMAC TX channel, the MAL finds the starting address of the buffer descriptor table for the channel by looking in the corresponding Transmit Channel Table Pointer Register. If the first descriptor is marked as ready, the MAL starts processing the associated buffer.

When the MAL begins processing a packet, it writes the contents of the descriptor status/control field into the COMMAC. This information, (depending on communication core implementation), may be used by the communication core to configure each packet transfer.

Once all data from the current buffer has been transferred to the communication core on the channel, MAL moves on to the next buffer descriptor in the table.

If a given buffer descriptor indicates that it contains the last section of the current packet, the MAL informs the channel that the last data transferred to the channel completed the transfer of a data packet. At this point, the COMMAC asks the MAL to read the packet status. The MAL then writes this information back into the status/control field of the last buffer descriptor of the packet.

The COMMAC channel can request that the MAL process the next buffer descriptor and the same packet handling process will be initiated. The first descriptor in the next packet follows the descriptor marked "last" in the previous packet.

23.5.1 Wrapping the BD Table for Transmit

When the MAL processes a buffer descriptor (while handling a packet for a COMMAC channel), it may encounter a Wrap indication within a buffer descriptor control field. This causes the MAL to go back to the beginning of the buffer descriptor table for the next descriptor table entry. (This will also happen when the MAL reaches the maximum number of descriptors.) The wrapping of the BD table, like all other BD table handling processes, is transparent to the COMMAC.

23.5.2 Continuous Mode for Transmit

After transmitting the data pointed to by a buffer descriptor, the MAL clears the Ready bit in the buffer descriptor control/status field. In this way, the MAL will not process the same buffer descriptor again until software has filled the buffer with valid data and set the Ready bit in the descriptor again. While the Continuous Mode (CM) bit is set in the status/control field, the MAL will not clear the Ready bit. The Continuous Mode allows re-transmission of the current data buffer without software intervention. This mode is generally used by protocols in which frequent re-transmission is an integral part of the protocol itself. In such cases, re-transmission can be performed without software intervention.

23.5.3 Back Up a Packet for Transmit

The MAL is capable of re-transmitting the last packet ("back up a packet") following a request from a COMMAC. If re-transmission is requested by the COMMAC, it must be assured that all the buffers of the re-transmitted packet are available and were not re-processed by the device driver. In regular operation, the MAL resets the Ready bit of each buffer descriptor when finished processing the descriptor. When the MAL is requested by the COMMAC to retransmit the last packet (the Back Up a Packet bit in the COMMAC TX channel Status Halfword is set), the MAL doesn't reset the READY bit in the last processed buffer descriptor, activate the end of packet interrupt, or write the status back to the descriptor in the memory. The MAL also doesn't consider this as an end of packet event.

On the next service request from the same channel, the MAL starts transmitting the packet again, starting from the first descriptor.

Note: The last processed buffer descriptor can be either the last descriptor of the packet or, in case of early packet termination, the buffer descriptor that was being processed when the transmit channel initiated the early packet termination. The MAL will retransmit the backed-up packet regardless of the Ready bit value.

During retransmission of a backed-up packet, the MAL may use descriptors on which the Ready bit was already cleared. Therefore, the device driver should not reuse descriptors before the Ready bit of the last descriptor is cleared.

Note: In the case of descriptor not valid, which is the first one in TX channel, COMMAC is not allowed to return a status that contains a Back-Up a Packet request.

23.5.4 Descriptor Not Valid for Transmit

When the MAL accesses a buffer descriptor, it checks whether or not the Ready bit is set. If the Ready bit is not set, two cases apply (special treatment of the READY bit is performed in the case of Back-up a packet):

For the case when the READY bit is not set:

If the descriptor is the first descriptor of the packet the MAL informs the channel that data is currently unavailable. Further handling of this scenario is COMMAC-specific. The channel might either instruct MAL to access the same buffer descriptor periodically (by keeping its service request to the MAL active) until it becomes ready, or 'give up' on the descriptor, completing the end-of-packet protocol with the MAL. The channel might also indicate the buffer descriptor status to the device driver via an interrupt. However, in this case the COMMAC should eventually complete the packet transfer protocol with MAL. Following a descriptor not valid indication, the MAL's BD pointer continues pointing to the same location in the BD table. The next time a descriptor read is initiated by the COMMAC, the MAL will search for the buffer in the same location.

If the descriptor is not the first descriptor of the packet, it is considered a descriptor error. the MAL deactivates the channel and from its point of view, the processing of the current packet has ended. Software may learn about this situation from one of two MAL interrupts (or from both). The first one is a non-maskable interrupt that indicates the number of the TX channel, in which the descriptor error had occurred (interrupt bit for each TX channel, see *MAL Interrupt Enable Register (MAL0_IER)* on page 572). The second one is a maskable interrupt which indicates a descriptor error event, regardless the channel number (one interrupt bit for all the channels, see *MAL Error Status Register (MAL0_ESR)* on page 570). For more information about error handling, see *Error Handling* on page 560.

For the case of a back-up packet:

When the current transmitted packet is a backed-up packet, all descriptors except the last, are valid even if the READY bit is not set. In this case, (not the last descriptor) the MAL processes the packet descriptors regardless the READY bit value. If the READY bit of the last descriptor in the backed-up packet is not set, the MAL treats it as a descriptor error. The MAL handles the descriptor error as described above for the case when the packet isn't a backed-up packet.

23.5.5 Scroll Descriptors for Transmit

The MAL can be configured by software, in the case of early packet termination, to scroll in the buffer descriptor table to the first descriptor of the next packet.

When a multiple-buffer packet is terminated early by the COMMAC, while the MAL is processing a buffer which is not the last buffer in the packet, the MAL can operate in one of the following ways:

The MAL Scroll Descriptor in the configuration register is set:

In this case the MAL reads the status word from the COMMAC channel. Then the MAL resets the READY bit in all the remaining buffer descriptors of the current packet. In addition, the MAL writes the status to all the buffer descriptors. On the next service of this channel, the MAL fetches the first descriptor of the next packet.

The MAL Scroll Descriptor in the configuration register is clear:

In this case the MAL will read the Status word from the COMMAC channel. Then the MAL will terminate the current channel service by resetting the READY bit of the last processed buffer descriptor (the one in which there was an early termination) and will write the status only to this descriptor. On the next service of this channel, the MAL will fetch the next descriptor in the current packet. In this case, the software is responsible to monitor the MAL location in the buffer descriptor table.

In the case that the COMMAC requests a re-transmit of the early terminated packet (when the "backup" bit in the COMMAC status is set), the MAL will re-transmit the packet regardless of the MAL Scroll Descriptor bit.

23.6 Receive Software Interface

The MAL uses the RX channel buffer descriptors in a manner similar to that used for transmission. Once an RX COMMAC channel requests that a new packet be processed, the MAL starts processing the channel's next buffer descriptor in the table. Once a channel is enabled in the MAL, the channel may request the MAL service. When it does, the MAL accesses the first buffer descriptor (in that channel's buffer descriptor table) that is pointed to by the COMMAC channel table pointer register. If that descriptor is ready (empty for RX), the MAL will start processing the buffer.

When it begins processing each packet, the MAL writes the contents of the status/control field into the COMMAC. This information can be used by the COMMAC for a per-packet configuration.

Once data is received from the memory, the MAL moves the data from the RX channel FIFO into the data buffer pointed to by the first buffer descriptor. The current buffer descriptor may be closed for two reasons: there is no more room left in the buffer, or the COMMAC channel indicated that the packet reception ended. If additional buffering space is needed for the current packet, the MAL moves on to the next buffer descriptor. As each buffer descriptor is closed, the MAL updates the length field with the actual amount of bytes written into the buffer. The maximal length of the buffers for each channel is defined by a configuration register (see *Receive Channel Buffer Size Register (MAL0_RCBSx)* on page 574). The maximal receive buffer length is defined per channel.

Once the COMMAC channel indicates that the packet reception has ended, it is expected to request that the MAL update the received packet status in the BD status/control field. The MAL updates the packet status and notifies the COMMAC. At this point the packet is considered received and the COMMAC may request that the MAL begin the process of receiving a new packet. The first buffer of the next packet is the buffer in the BD table that followed the last descriptor of the previous packet.

23.6.1 Wrapping the BD Table for Receive

When the MAL processes the buffer descriptor, it may encounter a Wrap indication within a buffer descriptor control field. This causes the MAL to go back to the head of the channel's buffer descriptor for the next buffer descriptor. This also happens when the MAL reaches the maximal number of descriptors.

23.6.2 Continuous Mode for Receive

After using a buffer descriptor, the MAL sets the buffer descriptor control to the Not-Empty state. In this way, the MAL will not use the same buffer descriptor a second time until the software has processed the not-empty buffer descriptor and set it to Empty again. The MAL will not clear the Empty bit while the Continuous Mode (CM) bit is set in the status/control field. The Continuous Mode is generally used by protocols where frequent collisions are an integral part of the protocol itself (forcing the COMMAC to abort a reception process and restart). In such cases, re-reception can be performed without software intervention.

23.6.3 Descriptor Not Valid for Receive

When the MAL accesses a buffer descriptor it may find that the Empty bit is not set. In the case of an RX channel descriptor, this situation is considered as a descriptor error. The MAL deactivates the channel and from its point of view, the processing of the current packet has ended. Software may learn about this situation from one of two MAL interrupts (or from both):

- An RXDE interrupt with the MAL0_RXDEIR indicating which channel caused the interrupt
- An SERR interrupt (system error) with one interrupt bit for all channels in the MAL0_ESR

For more about error handling, see *Error Handling* on page 560.

23.6.4 Buffer Length for Receive

The maximum length of an RX buffer descriptor is predetermined for all RX descriptors in each channel. The data-length value is programmable through a set of MAL registers (see *MAL Registers* on page 567). The actual data length field within the RX buffer descriptor is written by the MAL. If the buffer is completely filled up, the value written will match the value programmed into the matching RX-Channel-Descriptor data-length register. If the buffer is only partially filled up (for example, when the RX packet ended before running out of buffer space), the actual amount of space filled is written into the length field.

23.7 Buffer Descriptor Status/Control Fields

The following sections detail the status/control field bits. The information fields within the status/control field can be divided as follows:

- Information from a software device driver directed to MAL and COMMAC
- Information from MAL and COMMAC directed to software
- Status/control field handling
- Status/control field format
- TX status/control field format
- RX status/control field format

23.7.1 Information from a Software Device Driver Directed to MAL and COMMAC

- MAL-related buffer descriptor processing information:
  - Buffer Ready/Not Ready (determines the buffer's validity).
  - Wrap to top of table or continue to next descriptor.
  - In a transmit buffer descriptor – Is the current buffer the last one in the packet?
  - Continuous or normal mode; that is, should the MAL change the Ready/Not Ready value?
  - Should the channel generate an interrupt following the end of packet processing?

- COMMAC channel configuration information: Protocol specific configuration.

23.7.2 Information from MAL and COMMAC Directed to Software

- MAL generated status information:
  - Buffer Ready/Not Ready (passes the buffer handling to software).
  - In receive buffer descriptor - Is the current buffer the first one in the packet?
  - In receive buffer descriptor - Is the current buffer the last one in the packet?
- COMMAC channel generated status information:
  - Protocol specific error and status information (transmit and receive).

23.7.3 Status/Control Field Handling

When the MAL accesses a new buffer descriptor, the status/control word is written to the COMMAC channel. This allows the channel to configure itself for the current packet.

For all "intermediate" buffer descriptors (all descriptors that do not contain the packet's ending), the status/control field is written by the MAL (rather than the COMMAC). In this case, the status/control field indicates that the current buffer is not the last one in the current packet.

As the MAL finishes processing the last buffer descriptor in a given packet, it reads the channel's status (via an OPB transaction) and writes it into the buffer descriptor's status/control field.

In effect, since all of the various control and status fields do not overlap, the status/control halfword is read/written as a whole. Each agent (MAL, COMMAC channel, and software) reads the entire status/control halfword, relates to specific fields of interest, and updates another subset of fields within the same halfword. While an agent modifies its related fields, all other fields remain unchanged.

23.7.4 Status/Control Field Format

The status/control halfword is divided into COMMAC channel data and MAL related data. As explained above, the MAL related fields are either aimed at controlling the MAL or written by the MAL for use by the software. The MAL fields are of no interest to the COMMAC (except the Ready and Empty bits).

The same applies to the COMMAC channel fields. The COMMAC related fields are either aimed at controlling the COMMAC or written by COMMAC for use by the software. These fields are of no interest to the MAL.

The MAL will not manipulate the COMMAC related fields, and COMMAC is not allowed to manipulate the MAL related fields.

23.7.5 TX Status/Control Field Format

The bit numbering in *Figure 23-6* relates to the Buffer Descriptor's fullword which contains both the status/control and the length fields.

Figure 23-6. TX Status Control Field			
0	R	Ready 0 Not ready 1 Ready	This bit is set by the device driver and is cleared by MAL. The device driver sets this bit after preparing the buffer for transmission. MAL clears this bit when it finishes processing the buffer descriptor. MAL doesn't clear the Ready bit in the case of backing-up a packet request and in case of continuous mode (see *Back Up a Packet for Transmit* on page 553 and*Continuous Mode for Transmit* on page 553).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first buffer descriptor in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the R-bit is not cleared by MAL. This ensures that the data buffer is ready for transmission when MAL next accesses this buffer descriptor. However, the R-bit is cleared if an error occurs during transmission. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
4		Reserved	It is assumed that this bit is set to zero by the software.
5	I	Interrupt 0 The End of Buffer field and interrupt is not set after processing the current buffer. 1 After finishing processing the current buffer, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the COMMAC.
6:15		COMMAC Related Data	These bits are COMMAC specific and may contain control fields generated by the software in order to control the COMMAC channel. They may also contain status fields, generated by the COMMAC channel, that will be processed by software.

23.7.6 RX Status/Control Field Format

The bit numbering in *Figure 23-7* relates to the buffer descriptor's fullword which contains both the status/control and the length fields.

Figure 23-7. RX Status Control Field			
0	E	Empty 0 The data buffer associated with this buffer descriptor has been filled with received data, or data reception has been aborted due to an error condition. 1 The data buffer associated with this buffer descriptor is empty, or reception is currently in progress.	When this bit is 0 (not empty), software is free to examine or write to any fields of this buffer descriptor. While this bit is set to Not Empty, the MAL will not use this buffer descriptor again. When this bit is 1 (empty), this buffer descriptor and its associated receive buffer are owned by MAL. Once the E-bit is set, software should not write to any fields of this RX buffer descriptor. The MAL clears this bit after the buffer has been filled with received data or after an error is encountered. Software sets this bit to Empty after preparing the buffer for reception. This bit controls MAL and software activities (see bit 2: continuous mode below).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first buffer descriptor in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the E-bit is not cleared by MAL. This ensures that the data buffer is ready to receive data when MAL next accesses this buffer descriptor. However, the E-bit is cleared if an error occurs during reception. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
4	F	First 0 This is not the first buffer in the current packet. 1 This is the first buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
5	I	Interrupt 0 The End of Buffer field and interrupt is not set after processing the current buffer. 1 After finishing processing the current buffer, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the COMMAC.
6:15		COMMAC Related Data	These bits are COMMAC specific and may contain control fields generated by the software in order to control the COMMAC channel. They may also contain status fields, generated by the COMMAC channel, that will be processed by software.

23.8 MAL Programming Notes

The following sections contain information about programming the MAL.

23.8.1 MAL Initialization

MAL initialization includes two parts: configuration and channel activation.

Configuration involves two steps:

- MAL configuration - This step is done only after a power on reset or after a MAL soft reset. The following registers are involved:
  - Configuration Register (MAL0_CFG). This register defines MAL operation on the PLB and OPB.
  - Interrupt Enable Register (MAL0_IER). This register is used to enable interrupts for various MAL error conditions.
- Channel specific configuration - This information may be changed only when the associated channel is not active. (The bit for the channel in the TX or RX Channel Active Set Register, is cleared.) The following registers are involved:
  - MAL0_RCBSx – RX Buffer Size (one register for each RX channel). This register defines the length of the RX buffers in memory.
  - MAL0_TXCTPxR or MAL0_RXCTPxR – Channel Table Pointer Register (one register for each channel). This register is programmed with the memory address of the first buffer descriptor table entry for the channel.

Setting the channel specific configuration can be done as part of MAL initialization or as part of the COMMAC initialization process. In order to activate a channel, the following actions should be taken:

- The channel has to be configured in the MAL
- The related bit in Channel Active Set Register (MAL0_TXCASR or MAL0_RXCASR) has to be set
- The channel operation must be enabled (COMMAC configuration)

23.8.2 Interrupts

The MAL in the PPC440EP has five interrupt lines which are connected to the Universal Interrupt Controller (*Universal Interrupt Controller* on page 223) Two interrupt lines, one for TX and one for RX, are used for interrupt events during packet transfer. An additional two interrupt lines, one for TX and one for RX, are used to report descriptor errors on a per-channel basis. The fifth interrupt is used to report MAL errors.

- TXEOB interrupt line is used to report end of buffer or end of packet for a specific TX channel. A bit for the related channel is set in the MAL0_TXEOBISR. See *End of Buffer Interrupt Status Registers* on page 570.
- RXEOB interrupt line is used to report end of buffer or end of packet for a specific RX channel. A bit for the related channel is set in the MAL0_RXEOBISR. See *End of Buffer Interrupt Status Registers* on page 570.
- TXDE interrupt line is used to indicate a descriptor error event in a specific TX channel descriptor table. A bit for the related channel is asserted in the MAL0_TXDEIR. See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 572.
- RXDE interrupt line is used to indicate a descriptor error event in a specific RX channel descriptor table. A bit for the related channel is asserted in the MAL0_RXDEIR. See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 572.
- SERR interrupt is used to report a system error indicated by the MAL. For more information on handling the SERR interrupts, see *Error Handling* on page 560.

23.8.3 Error Handling

MAL handles errors on a per-channel basis. Within a COMMAC channel, errors may arise from the COMMAC (detected as an OPB error), or from the memory access operations involved in MAL activity (detected as a PLB/ descriptor error).

When a bus error occurs, The MAL is notified by an OPB or PLB error signal. OPB errors are related to a specific channel and therefore channel operation is stopped. In the case of a PLB error, the MAL cannot identify which channel is involved, therefore channel operation is not stopped. When a descriptor error occurs, the MAL can again identify the channel involved, so channel operation is stopped. the MAL stops channel operation by clearing the associated bit in the MAL0_TXCASR or MAL0_RSCASR register.

The MAL keeps a record of the channels that experience errors and are made inactive. It also keeps a record of the characteristics of the first (or last) error detected (see *End of Buffer Interrupt Status Registers* on page 570).

Note: The device driver is responsible for configuring the MAL before a COMMAC can begin requesting the MAL to process packets of data. The device driver should ensure that channels are not enabled during reconfiguration; otherwise, fatal errors may occur.

23.8.3.1 Error Detection

The MAL communication, both with COMMACs and with memory, is carried out via the OPB or PLB. As long as this bus communication is error-free and no descriptor errors are detected, the MAL maintains normal activity with the channels set by the processor as active in the Channel Active Registers.

When an error is detected while performing a transfer for a channel, the MAL asserts a maskable interrupt signal. If the identity of the channel is known (as is the case for OPB errors or descriptor errors) then MAL immediately halts the dialogue with the channel. No further transactions are made, and that channel is registered by the MAL as a non-active channel. The MAL resets the channel by resetting its active bit in the Channel Active Register. Software must access the Channel Active Register in order to reactivate the channel.

If the identity of the channel that caused the error is not known (as is the case for PLB errors) then the MAL continues to work normally. Error resolution and channel de-activation are the responsibility of the software.

23.8.3.2 Indicated Errors

Error description is stored in the Error Status Register (MAL0_ESR), (see *MAL Error Status Register (MAL0_ESR)* on page 570).

- Descriptor Error

  A descriptor error is a data error recognized during access to the descriptor table. The error can occur during TX or RX transmission.

  For RX channels, a descriptor error occurs when the MAL accesses a descriptor in which the Empty bit is cleared.

  For TX channels, a descriptor error occurs when the MAL accesses a descriptor in which the Ready bit is cleared. The following cases are exceptions.

  - On access to the first buffer descriptor in a TX packet.
  - On access to a buffer descriptor that is not the last descriptor in a backed-up packet.

  As a result of this error, the following actions are taken by the MAL:

  - The Active bit of the related channel is reset and the channel activity is halted until software reactivates channel activity.
  - The associated bit in the TX Descriptor Interrupt Error Register (MAL0_TXDEIR) or RX Descriptor Error Register (MAL0_RXDEIR) is set, causing a non-maskable TXDE interrupt or RXDE interrupt respectively.
  - When the channel is reactivated, the MAL points to the descriptor at the head of the BD table.
- OPB Non-Fullword Error

This error indicates that a non-fullword acknowledge was asserted by a slave.

Following this error, the active bit of the associated channel is reset and channel activity is halted until it is reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- OPB Time-Out Error

  This error indicates that an OPB time-out error was reported by the OPB arbiter.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until reactivated by software. When the channel is reactivated, the MAL points to the descriptor at the head of the BD table.

- OPB Error

  This error indicates that an OPB error was detected.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until reactivated by software. When the channel is reactivated, the MAL points to the descriptor at the head of the BD table.

- PLB Error

  This error indicates that a PLB error was detected (from the PLB slave).

  In this case, the MAL cannot determine which channel caused the error. Therefore, operation is not halted for any of the channels.

23.8.3.3 Error Handling Registers

The MAL error handling logic includes two registers.

- Error Status Register (ESR)

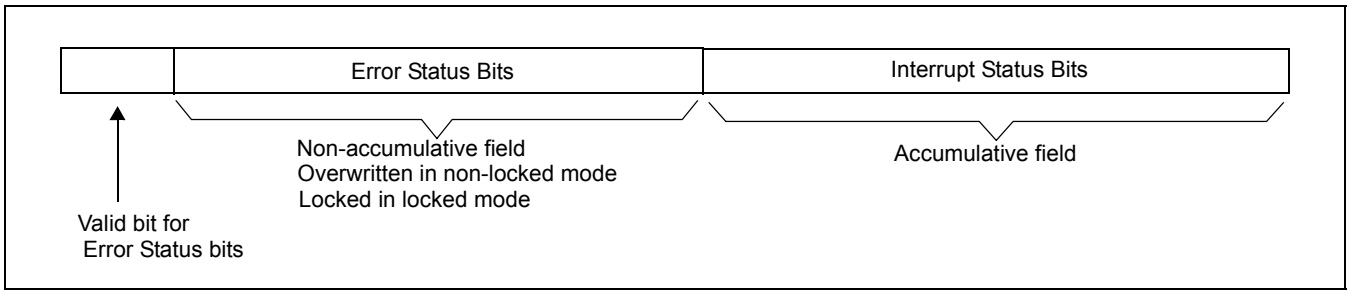
  This register holds information about the error that occurred and the interrupt status. The register includes the following fields:

  Error status – This field holds the error information. The information includes the number of the channel on which the error occurred (if known) and the type of the error. The error can be either the last detected error or a locked error if "Locked error mode" is active. See *Operational Error Modes* on page 563 for description of the Locked error mode.

  The error status field includes an "Error Valid" bit which indicates whether there is valid error information in the error status field or not. The error status field is not valid when the "Error Valid" bit is cleared (by writing 1 to this bit).

Interrupt status – Every error detected by the MAL sets a related bit in the interrupt status field. Software can clear an interrupt status bit by writing 1 to the bit to be cleared. The bits in this field are accumulative which allows more than one interrupt to be indicated in the register.)

Figure 23-8. Error Status Register Field



23.8.3.4 Operational Error Modes

The MAL can operate in two different error handling modes:

- Locked Error Mode: Information about the error is written to the Error Status Register, and the Valid bit in that register is set. Information in the Error Status field of the register stays locked until software unlocks it by resetting the error Valid bit. The Interrupt Status bits of the Error Status Register are not locked in this mode, so software can find out if more errors occur. However, the Error Status field applies only to the first error that is locked.
- Non-Locked Error Mode: Information about the error is written in the Error Status Register, and the error Valid bit is set. Each new error will be overwritten, so the information in the Error Status Field is valid only for the last error that occurred.

In both modes, each error written in the error description field will set the error Valid bit, and it is the responsibility of software to reset this bit.

The error handling mode is programmed in the MAL Configuration Register (see *MAL Configuration Register (MAL0_CFG)* on page 567).

23.8.3.5 Resolution of an Error Situation

When the MAL encounters an error, it reacts as follows:

- Writes information about the error in the Error Status Register (ESR). This information includes the channel ID of the channel which caused the error (if known), the bus on which the error occurred, and the kind of error that occurred.
- Resets the channel that caused the error (if known) in the Channel Active Register.
- Updates the Interrupt Status bits in the MAL0_ESR. Then, depending on the mask defined in MAL0_IER (Interrupt Enable Register), it sends an interrupt to the Universal Interrupt Controller.

After receiving an interrupt from the MAL, software can analyze the error information read from the Error Status Register. Software can restart channel activity by setting the associated bit in the Channel Active Register.

When a channel is stopped and restarted, the MAL starts processing descriptors from the first descriptor in the channel descriptor table. Therefore, software may also update the value of the other channel related registers (see *Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)* on page 573) in order to continue from the same buffer in memory.

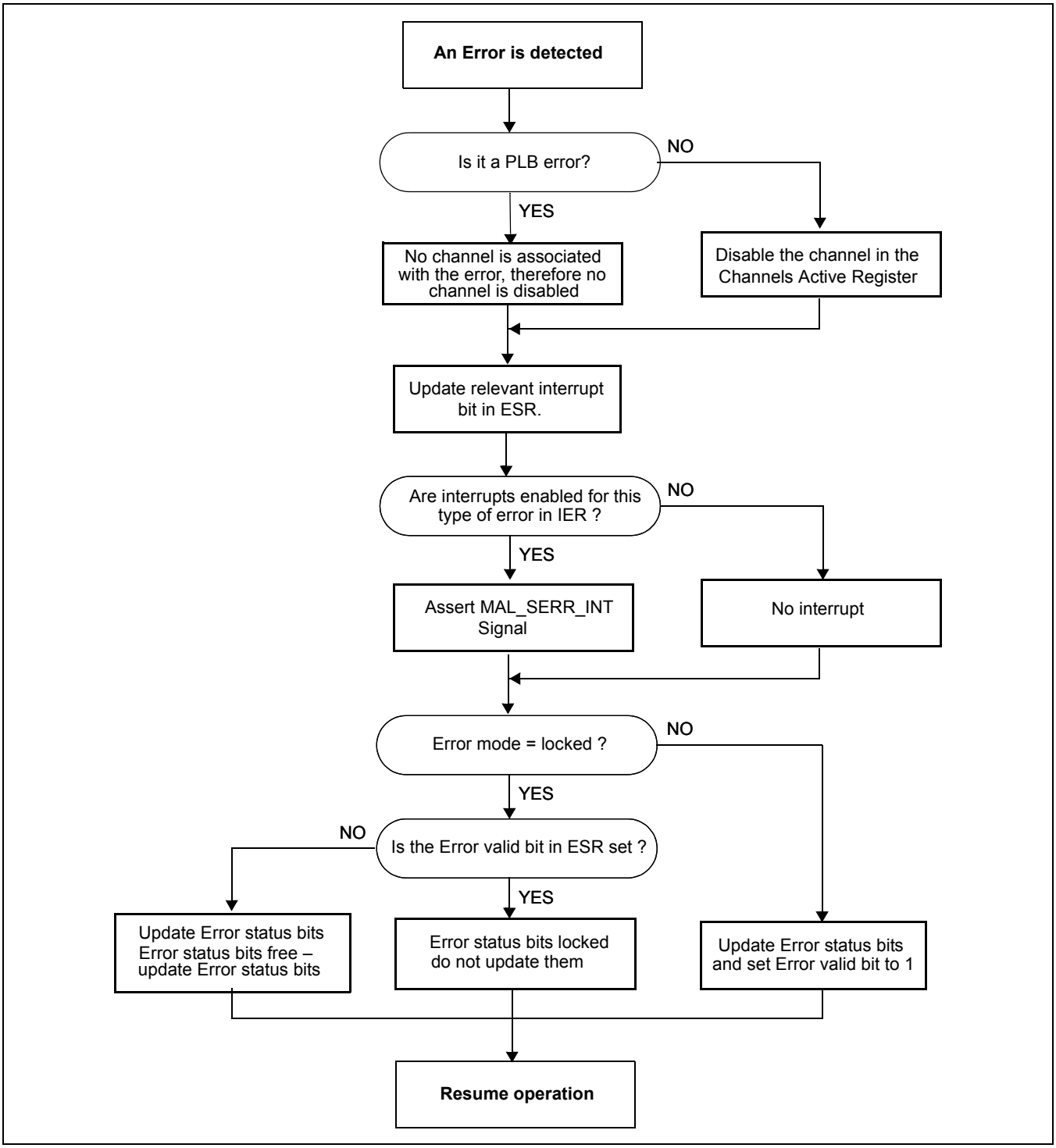
In the case of PLB errors, the MAL does not know which channel caused the error. It is the responsibility of the software to analyze the MAL error registers and the PLB slave error registers to determine which channel caused the error. Software should reset the channel within the MAL, resolve the problem, and then reactivate the channel.

See *Resolution of an Error Situation* on page 563 for a flow chart illustrating the steps the MAL performs when resolving an error situation.

23.8.3.6 Interrupts To Software

Figure 23-9 describes MAL actions once an error is detected. Note that the actual decisions the MAL makes may be in a different order than represented by this figure. In any case, the device driver should consider that all of the MAL actions are performed at the same time.

Figure 23-9. MAL Error Processing



23.9 MAL Serial Device Control Registers

The following serial device control registers control PPC440EP MAL functions.

23.9.1 SDR0_MALRBL Register

Figure 23-10 describes the SDR0_MALRBL register.

Reset value = 0x55550000.

Figure 23-10. MAL Receive Burst Length Register (SDR0_MALRBL)

0		Reserved	
1:2	RBL0	MAL Receive Burst Length 0 00 2 words 01 4 words 10 8 words 11 16 words	Indicates the requested amount of data to be received between MAL and EMAC0 in a single transaction.
3:4		Reserved	
5:6	RBL1	MAL Receive Burst Length 1 00 2 words 01 4 words 10 8 words 11 16 words	Indicates the requested amount of data to be received between MAL and EMAC1 in a single transaction.
7:31		Reserved	

23.9.2 SDR0_MALRBS Register

Note: The SDR0_MALRBS register is reserved and must always be set to 0xF0000000.

23.9.3 SDR0_MALTBL Register

Reset value = 0x55550000.

Figure 23-11. MAL Transmit Burst Length Register (SDR0_MALTBL)

0		Reserved	
1:2	TBL0	MAL Transfer Burst Length 0 00 2 words 01 4 words 10 8 words 11 16 words	Indicates the requested amount of data to be transferred between MAL and EMAC0 in a single transaction.
3:4		Reserved	
5:6	TBL1	MAL Transfer Burst Length 1 00 2 words 01 4 words 10 8 words 11 16 words	Indicates the requested amount of data to be transferred between MAL and EMAC1 in a single transaction.
7:31		Reserved	

23.9.4 SDR0_MALTBS Register

Note: The SDR0_MALTBS register is reserved and must always be set to 0xF0000000.

23.10 MAL Registers

The MAL registers are Device Control Registers (DCRs). Unless otherwise specified, all register fields are initialized at chip reset to 0. Reserved fields are read as undefined and must be written as 0s.

Table 23-2. MAL Register Summary

Register	DCR Number	Access	Description	Page
MAL0_CFG	0x0 0180	R/W	MAL Configuration Register	567
MAL0_ESR	0x0 0181	R/Clear	MAL Error Status Register	570
MAL0_IER	0x0 0182	R/W	MAL Interrupt Enable Register	572
MAL0_TXCASR	0x0 0184	R/W	Transmit Channel Active Set Register	569
MAL0_TXCARR	0x0 0185	R/W	Transmit Channel Active Reset Register	569
MAL0_TXEOBISR	0x0 0186	R/Clear	Transmit End of Buffer Interrupt Status Register	570
MAL0_TXDEIR	0x0 0187	R/Clear	Transmit Descriptor Error Interrupt Register	572
MAL0_RXCASR	0x0 0190	R/W	Receive Channel Active Set Register	569
MAL0_RXCARR	0x0 0191	R/W	Receive Channel Active Reset Register	569
MAL0_RXEOBISR	0x0 0192	R/Clear	Receive End of Buffer Interrupt Status Register	570
MAL0_RXDEIR	0x0 0193	R/Clear	Receive Descriptor Error Interrupt Register	572
MAL0_TXCTP0R	0x0 01A0	R/W	Transmit Channel 0 Table Pointer Register	573
MAL0_TXCTP1R	0x0 01A1	R/W	Transmit Channel 1 Table Pointer Register	573
MAL0_TXCTP2R	0x0 01A2	R/W	Transmit Channel 2 Table Pointer Register	573
MAL0_TXCTP3R	0x0 01A3	R/W	Transmit Channel 3 Table Pointer Register	573
MAL0_RXCTP0R	0x0 01C0	R/W	Receive Channel 0 Table Pointer Register	573
MAL0_RXCTP1R	0x0 01C1	R/W	Receive Channel 1 Table Pointer Register	573
MAL0_RCBS0	0x0 01E0	R/W	Receive Channel 0 Buffer Size Register	574
MAL0_RCBS1	0x0 01E1	R/W	Receive Channel 1 Buffer Size Register	574

23.10.1 MAL Configuration Register (MAL0_CFG)

This register defines the operational mode of MAL. Unless a configuration change is required during system operation, the configuration register needs to be set only during system initialization.

Figure 23-12. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	
8:9	PLBP	PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB.
10	GA	Guarded Active 0 GUARDED signal not applied to the PLB slave 1 GUARDED signal applied to the PLB slave	When this bit is set, MAL applies the GUARDED signal to the PLB slave when it is the initiator on the bus. When set, the slave can access all the memory in the current page as well as the subsequent page.
11	OA	Ordered Active 0 ORDERED signal not applied to the PLB slave 1 ORDERED signal applied to the PLB slave	When this bit is set, MAL applies the ORDERED signal to the PLB slave when it is initiator on the bus during data write transactions. Note that the ORDERED signal is always driven active during status write transactions.
12	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.
13:16	PLBLT	PLB Latency Timer	Determines the number of cycles allowed for burst transactions on the PLB.
17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the COMMACs.
25:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). **Note:** An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.
30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode, otherwise it will handle errors in a non-locked mode.

31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related COM-MAC. When set, Scrolling mode is active.

23.10.2 Channel Active Set and Reset Registers

For the Channel Active Set/Reset Registers (MAL0_TXCASR, MAL0_TXCARR, MAL0_RXCASR, MAL0_RXCARR), each bit represents its associated channel (bit 0 for channel 0, and so on). When a bit is set to1, the channel is enabled for operation. When a bit is set to 0, the channel is disabled and MAL ignores any requests for service on the channel. If a channel is active when its enable bit is set to 0, MAL stops processing the current packet. After the enable bit associated with a channel is set to 0, MAL goes back to the top of the channel descriptor table (pointed to by MAL0_TXCPTxR or MAL0_RXCPTxR).

- To enable a channel:
  - Write a 1 to its corresponding bit in MAL0_CASR.
  - Multiple channels can be enabled with a single MAL0_CASR register write.
- To stop and reset a channel:
  - Write a 1 to the corresponding bit in MAL0_CARR.
  - Writing a 0 to bits in MAL0_CARR has no effect.
  - Multiple channels can be reset with one MAL0_CARR register write.

MAL also clears the enable bit associated with a channel after an error occurs on the channel. The Channel Active Set/Reset Registers can be read to determine the currently active channels.

Figure 23-13. MAL Transmit Channel Active Set Register (MAL0_TXCASR)

0:31		Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.

Figure 23-14. MAL Transmit Channel Active Reset Register (MAL0_TXCARR)

0:31		Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.

Figure 23-15. MAL Receive Channel Active Set Register (MAL0_RXCASR)

0:31		Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two receive channels.

Figure 23-16. MAL Receive Channel Active Reset Register (MAL0_RXCARR)			
0:31		Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 0 is written to the bit, channel operation is disabled. MAL0 has two receive channels.

23.10.3 End of Buffer Interrupt Status Registers

Each bit in the end of buffer interrupt status registers (MAL0_TXEOBISR and MAL0_RXEOBISR) is associated with the descriptor buffer table of a channel.

MAL0_TXEOBISR contains the end of buffer status bits for each transmit channel. MAL0_RXEOBISR contains the end of buffer status bits for the receive channels. The mechanism for both registers is identical.

MAL sets a bit associated with a channel bit under any of the following conditions:

- When MAL finishes the processing of a buffer (writes back the status to the current descriptor), the related bit in this register is set if the I bit in the descriptor status is set.
- When MAL finishes the processing of a packet (writes back the status of the last buffer of the packet) and MAL0_MCR[EOPIE] is set.

Note: If MAL finishes processing a packet that is backed up, MAL does not consider it as an end of packet. Therefore, MAL does not set the appropriate channel bit in the end of buffer interrupt status registers.

- When the Bad Packet bit is set in the COMMAC channel status halfword.

The device driver resets the interrupt by writing a 1 to the related bit. Writing a 0 has no effect.

Figure 23-17. MAL Transmit End of Buffer Interrupt Status Register (MAL0_TXEOBISR)			
0:31		Transmit Channel End of Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four transmit channels.

Figure 23-18. MAL Receive End of Buffer Interrupt Status Register (MAL0_RXEOBISR)			
0:31		Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two receive channels.

The following paragraphs describe MAL error registers. For more information about MAL errors, see "Error Handling" on page -560.

23.10.4 MAL Error Status Register (MAL0_ESR)

This register holds the information about the error that occurred and the interrupts status. The register includes the following fields:

- **Error status bits** – This field holds the error information. The information includes the number of the channel on which the error occurred (if known) and the type of the error. The error can be either the last detected error or a locked error if "Locked error mode" is active. (See *Operational Error Modes* on page 563 for description of the Locked error mode.)

The error status field includes an "Error Valid" bit which indicates whether there is an error information in the error status field or not. The error status bits are not valid when the "Error Valid" bit is cleared (by writing 1 to this bit).

- **Interrupt status bits** – Every error detected by MAL sets a related bit in the interrupt status field. The interrupt status bits may be cleared by software by writing 1 to the bit to be cleared. The bits in this field are accumulative (more than one interrupt may be indicated here). These bits are masked by the IER (Interrupt Enable Register) to create a maskable interrupt, which is implemented by the MAL_SERR_INT signal.

Note: In order to reset the interrupt bits and the Error valid bit in the Error Status register, 1 must be written to the related bit. Writing 0 has no effect.

More than one bit can be cleared at a time and only R/W bits can be reset.

Figure 23-19. MAL Error Status Register (MAL0_ESR)

0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error. In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared. This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set
1:6	CID	Channel ID	This field contains the number of the channel which caused the locked error. Bit 1 indicates whether the channel ID represents an RX channel (1) or a TX channel (0). Bits 2:6 indicates the number of the channel that caused the error. **Note:** An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.
7:10		Reserved	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.
12	ONE	OPB Non-fullword Error 0 No error 1 Non-fullword asserted	Indicates that the error is a non-fullword acknowledge asserted by an OPB slave.
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout	Indicates the error is an OPB timeout.
14	OSE	OPB Slave Error 0 No error 1 OPB slave error	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.

16:26		Reserved	
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is *not* the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28	ONEI	OPB Non-fullword Error Interrupt 0 No error 1 Non-fullword acknowledgment from a slave	This bit is set following a non-fullword acknowledgment coming from a slave. Set condition for this bit generates a maskable interrupt.
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

23.10.5 MAL Interrupt Enable Register (MAL0_IER)

Each bit in the following register, when it is set, enables assertion of the interrupt signal (MAL0_SERR_INT) when the associated bit is set in MAL0_ESR.

Figure 23-20. MAL Interrupt Enable Register (MAL0_IER)

0:26		Reserved	
27	DE	Descriptor Error	When set, this bit enables the descriptor error (descriptor not valid) interrupt.
28	NWE	Non_W_Err_Int_Enable	When set, this bit enables OPB non-word transfer error interrupt.
29	TO	Time_Out_Int_Enable	When set, this bit enables OPB time-out error interrupt.
30	OPB	OPB_Err_Int_Enable	When set, this bit enables the OPB Slave error interrupt.
31	PLB	PLB_Err_Int_Enable	When set, this bit enables the PLB error interrupt.

23.10.6 Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)

Each bit in the following registers is related to a channel descriptor buffer table. Each bit indicates a descriptor data error related to a certain channel.

MAL0_TXDEIR contains the descriptor errors bits of the transmit channels. The MAL0_TXDEIR contains the descriptor errors bits of the receive channels. The mechanism (as described below) for both registers is the same.

MAL sets the bit associated with a channel bit when a descriptor data error was recognized during access to the descriptor table of a specific channel (see "Descriptor Error" on page -561).

The device driver resets the interrupt by writing a 1 to the related bit. Writing a 0 has no effect. When one or more of the MAL0_TXDEIR bits is set, the MAL_TX_DESC_ERR_INT bit is set. When one or more of the MAL0_RXDEIR bits is set, the MAL_RX_DESC_ERR_INT signal is set.

Figure 23-21. MAL Transmit Descriptor Error Interrupt Register (MAL0_TXDEIR)			
0:31		Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL_DESC_ERR_INT is set. Writing 1 to a bit clears it. MAL 0 has four transmit channels.

Figure 23-22. MAL Receive Descriptor Error Interrupt Register (MAL0_RXDEIR)			
0:31		Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL_DESC_ERR_INT is set. Writing 1 to a bit clears it. MAL0 has two receive channels.

23.10.7 Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)

MAL uses receive channel table pointer registers, one for each receive channel, and transmit channel table pointer registers, one for each transmit channel. The channel table pointer registers point to the base address, in memory, of the descriptor buffer table used by each channel.

Note 1: Bits 0–12 of the MAL0_TXCTPxR registers are mapped to the same physical register. Writing into any of these registers overwrites bits 0–12 in all MAL0_TXCTPxR registers; read operations have no effect. Similarly, bits 0–12 of the MAL0_RXCTPxR registers are mapped to the same physical register. Writing into any of these registers overwrites bits 0–12 in all MAL0_RXCTPxR registers, Read operations have no effect.

Note 2: When changing the value of any MAL0_TXCTPxR registers, all transmit channels must be idle. To verify that a channel is idle, check the Transmit Idle bit of the device. Another way to ensure that the channels are idle is to disable the channels before changing the MAL0_TXCTPxR register, and then reenable them once the MAL0_TXCTPxR register is set to its new value.

The transmit and receive channel table pointer registers have identical formats, as shown in *Figure 23-23* and *Figure 23-24*. MAL0 has four transmit channel table pointer registers, and two receive channel table pointer registers.

Figure 23-23. MAL Transmit Channel Table Pointer Register (MAL0_TXCTPxR)			
0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned double word (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.

Figure 23-24. MAL Receive Channel Table Pointer Register (MAL0_RXCTPxR)			
0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by a channel. The value entered should point to a location in memory accommodating an aligned double word (the three least significant bits of the pointer must be 000). MAL0 has two receive channels.

The Table Pointer Registers retain their value following Soft Reset or Channel Reset.

23.10.8 Receive Channel Buffer Size Register (MAL0_RCBSx)

Each receive channel has a fixed buffer length. MAL0_RCBSx is configured by the device driver to specify the length of the buffer. This size is the maximum number of bytes that can be stored in the buffer. Multiple buffers are used to store an incoming packet if the length of the incoming packet data is greater than the buffer length for the channel, as defined by the associated register for the channel.

The buffer length for a channel can be from 16 bytes to (4KB – 16) bytes, in 16-byte increments. This length is represented by a 8-bit field.

The receive buffer size, in bytes, is calculated as:

Receive Channel Buffer Size $\times$16

Figure 23-25 illustrates the MAL0_RCBSx registers.

Figure 23-25. MAL Receive Channel Buffer Size Register (MAL0_RCBSx)			
0:23		Reserved	
24:31		Receive Channel Buffer Size	Each channel is associated with a MAL0_RCBSn register. MAL0 has two receive channels.

24. EMAC to PHY Interface Bridge

The PPC440EP provides a bridge called the ZMII bridge that connects the ethernet media access controllers (EMACs) to standard physical devices (PHYs). Media independent interface (MII) and management data interface (MDI) signals to and from an EMAC are passed to the ZMII bridge. The ZMII bridge passes MII and MDI signals through to the PHY, or formats the MII signals to support the reduced media independent interface (RMII) or serial media independent interface (SMII).

Depending on the ZMII configuration, ZMII provides one of the following off-chip interfaces: one MII, two RMII, or two SMII. The interfaces are mutually exclusive. Interface selection is controlled by the Function Enable Register (ZMII0_FER).

RMII and SMII significantly reduce the external signal count required to support multiple EMAC ports. The interfaces share two management data interface (MDI) signals (EMCMDIO and EMCMDCLK). While the MII requires 16 additional pins for a single interface, RMII requires only seven pins (six data and control and a common clock) and SMII requires only four pins (two data, sync, and a clock). Configuring the ZMII bridge for SMII or RMII allows the use of both EMACs using a reduced number of external pins.

24.1 ZMII Features

The ZMII bridge features:

- Support for one MII PHY
- Support for two RMII PHYs
  - 10Mbps and 100Mbps data rates
  - 50MHz reference clock sourced from EMAC, or an external source, to the PHY
  - Independent 2-bit transmit and receive data paths
- Support for two SMII PHYs
  - 10Mbps and 100Mbps data rates
  - Half and full duplex operation
  - System clock sharing for multiple EMACs and PHYs
  - Independent 1-bit transmit and receive data paths
- Programmable selection of any EMAC to drive MDI
- Programmable selection of any EMAC to drive MII
- Programmable selection of either or both EMACs to drive RMII or SMII

24.2 ZMII Bridge Interface Signals

Table 24-1 lists the signals in the ZMII bridge interface. The "Pin" column lists the signals associated with package pins. Each pin is associated with one or more ZMII bridge signals, which are listed in the "MII," "RMII," and "SMII" columns. The ZMII bridge signal associated with each pin depends on the ZMII bridge interface selection. Note that MII, RMII, and SMII share the MDI signals ENETMDIO and EMCMDCLK.

Table 24-1. ZMII Bridge PHY Interface

MII	RMII	SMII	I/O	Description
EMCMDIO	EMCMDIO	EMCMDIO	I/O	MDIO data input/output
EMCMDClk	EMCMDCLK	EMCMDCLK	I	MDIO data clock
EMCRxD0	EMC0RxD0	EMC0RxD	O	MII receive data 0 RMII0 receive data 0 SMII0 receive data
EMCRxD1	EMC0RxD1	EMC1RxD	I	MII receive data 1 RMII0 receive data 1 SMII1 receive data
EMCRxD2	EMC1RxD0		I	MII receive data 2 RMII1 receive data 0
EMCRxD3	EMC1RxD1		I	MII receive data 3 RMII1 receive data 1
EMCRxDV	EMC1CRSDV		I	MII receive data valid RMII1 receive data valid
EMCRxClk			I	MII receive medium clock
EMCRxErr	EMC0RxErr		I	MII receive error RMII0 receive error
EMCCRS	EMC0CRSDV		I	MII carrier sense RMII0 carrier sense data valid
EMCTxClk	EMCRefClk	EMCRefClk	I	MII transmit clock RMII reference clock SMII reference clock
EMCCCD	EMC1RxErr		I	MII collision RMII1 receive error
EMCTxErr	EMC1TxEn		O	MII transmit error RMII1 transmit enable
EMCTxEn	EMC0TxEn	EMCSync	O	MII transmit enable RMII0 transmit enable SMII sync
EMCTxD0	EMC0TxD0	EMC0TxD	O	MII transmit data 0 RMII0 transmit data 0 SMII0 transmit data 0
EMCTxD1	EMC0TxD1	EMC1TxD	O	MII transmit data 1 RMII0 transmit data 1 SMII1 transmit data 1
EMCTxD2	EMC1TxD0		O	MII transmit data 2 RMII1 transmit data 0

Table 24-1. ZMII Bridge PHY Interface (continued)

MII	RMII	SMII	I/O	Description
EMCTxD3	EMC1TxD1		O	MII transmit data 3 RMII1 transmit data 1

24.3 EMAC-ZMII Bridge Interfaces

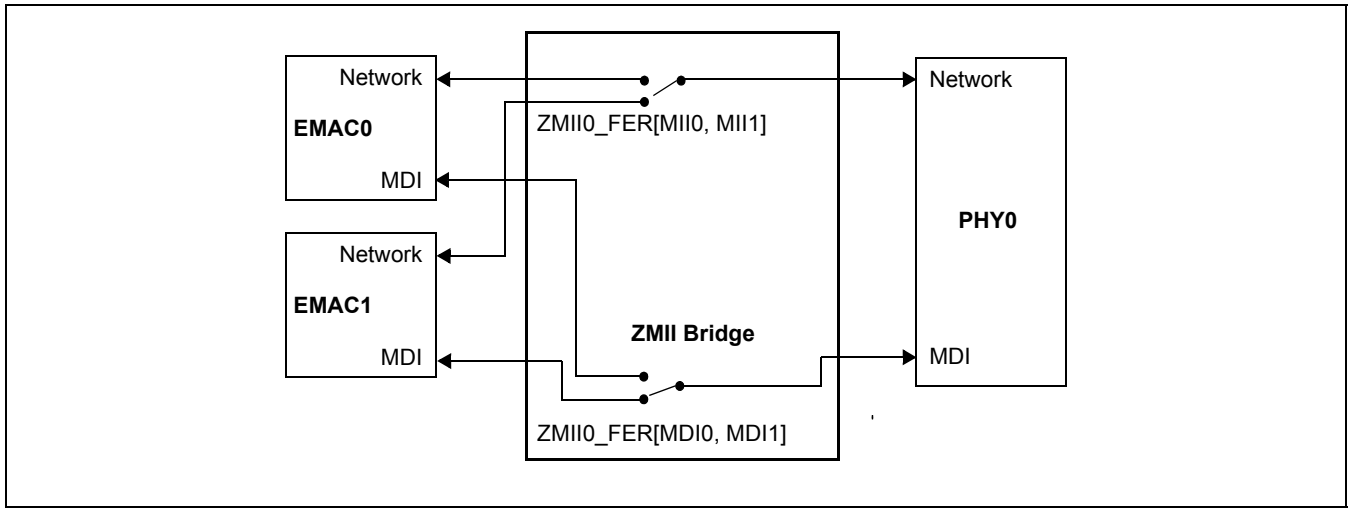
The ZMII bridge attaches to the EMACs on the standard MII, which consists of three sections: transmit data, receive data, and MDI. The transmit and receive data sections move data nibbles across the interface at 25 MHz and 2.5 MHz for 100 Mbps or 10 Mbps operation, respectively. Control signals define valid data, error conditions, and collision detection.

The standard MDI interface enables software to access registers in the PHY and consists of a single bidirectional data signal and a clock driven from ZMII. The MDI connection is always a pass-through interface in the ZMII Bridge.

24.3.1 MII Interface

When the ZMII bridge is configured for MII, the MII signals from one EMAC are simply passed through the ZMII bridge to a PHY. *Figure 24-1* shows an EMAC connected to the ZMII bridge and a PHY. If using MII mode, only a single EMAC may be used.

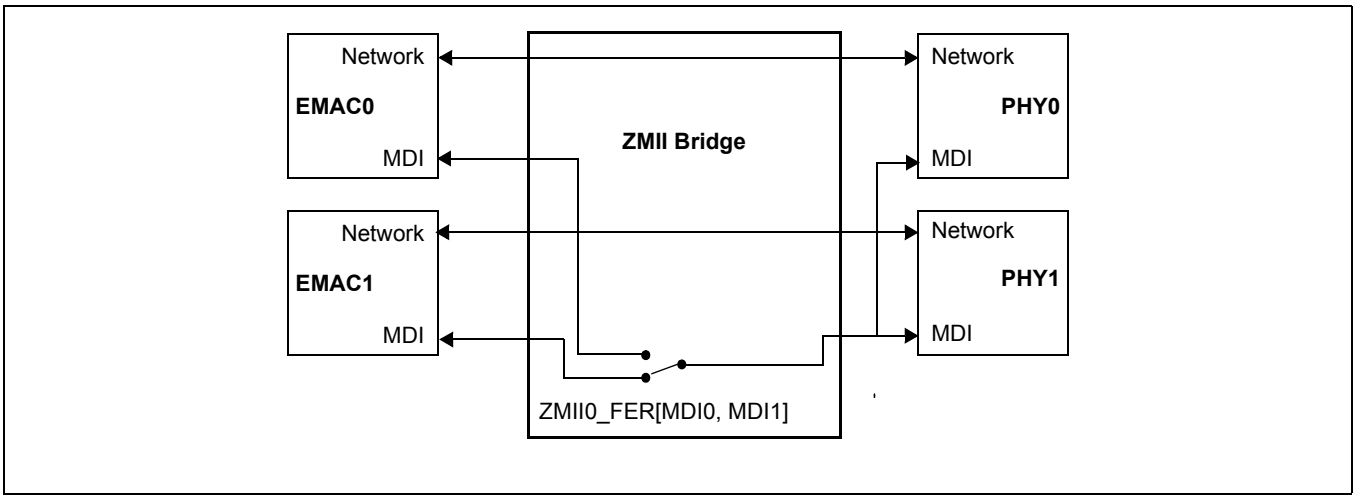
Figure 24-1. EMAC to PHY Using MII



24.3.2 RMII Interface

When the ZMII bridge is configured for RMII, the number of data signals is halved and the operating frequency is doubled to 50MHz. Each EMAC can connect through the ZMII bridge to a PHY. *Figure 24-2* shows EMAC0 and EMAC1 connected to the ZMII bridge and two PHYs.

Figure 24-2. EMAC to PHY Using RMII



24.3.2.1 SMII Interface

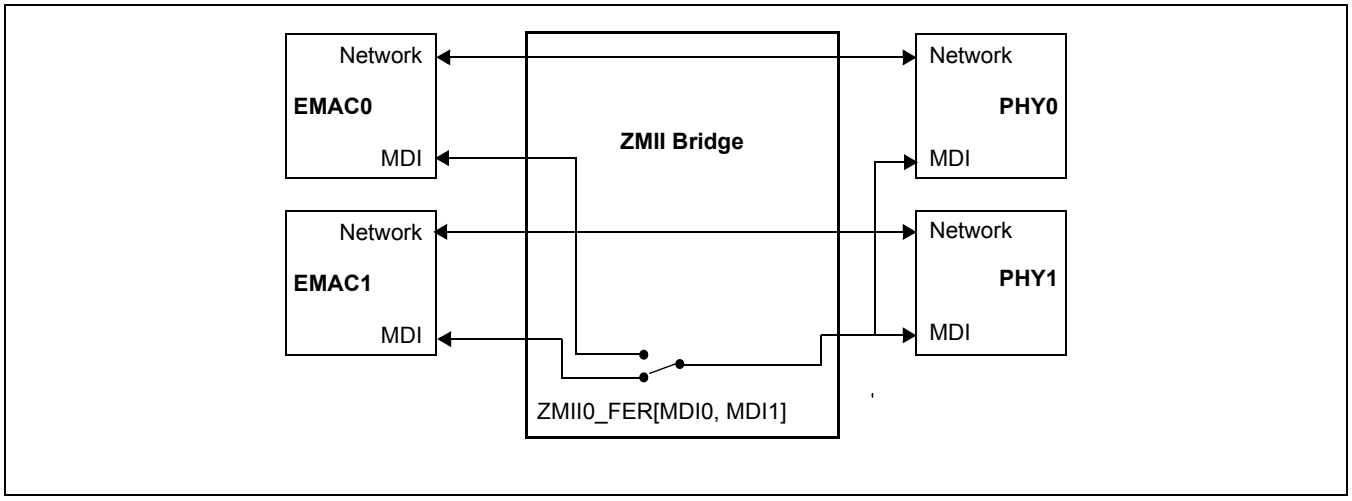
The SMII data and control paths are similar to those of RMII. When the ZMII bridge is configured for SMII, MII signal requirements are further reduced by quartering the data width (to a single bit) and multiplying the clock frequency by 10. All signals are synchronous to a 125 MHz clock and the sync signal.

Receive information (data and control) is moved in 10-bit segments. At 100 Mbps, each segment represents a new byte. At 10 Mbps, each segment is repeated ten times. Segment boundaries are defined by a sync signal. The ZMII bridge generates a pulse on the sync signal every 10 clocks.

Transmit operates in the same way. Transmit information (data and control) is sent in 10-bit segments. At 100 Mbps, a new byte is sent every 10 clocks. At 10 Mbps, each segment is repeated 10 times; the PHY can sample any of the repeated 10-bit segments.

Figure 24-3 shows the connection of the EMACs to the ZMII bridge with PHYs.

Figure 24-3. EMAC to PHY Using SMII



24.4 ZMII Bridge Registers

This section describes the registers in the ZMII bridge, which are listed in *Table 24.1*. The ZMII bridge registers are accessed using an OPB slave interface.

Table 24-2. Register Address List

Register	Description	Address	Access	Page
ZMII0_FER	ZMII Function Enable Register	0x0 EF60 0D00	R/W	579
ZMII0_SSR	ZMII Speed Select Register	0x0 EF60 0D04	R/W	580
ZMII0_SMIISR	ZMII SMII Status Register	0x0 EF60 0D08	R/W	581

24.4.1 ZMII Function Enable Register (ZMII0_FER)

ZMII_FER selects the various interface conversion functions provided by the ZMII bridge. ZMII_FER also deselects (functionally isolates) an unused EMAC. The interface conversion functions are mutually exclusive, that is, all EMACs must convert to the same interface (SMII, RMII, or MII) in the bridge, or be deselected (an EMAC must be deselected for MII). Mixed interface conversion is not supported.

Figure 24-4. ZMII Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC0 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.

3	MII0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC0 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
7	MII1	EMAC0 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, MII0, SMII1, or RMII1 is 1.
8:31		Reserved	

24.4.2 ZMII Speed Select Register (ZMII0_SSR)

ZMII_SSR selects the speed at which each EMAC transfer data (10Mbps or 100Mbps). When 10Mbps is selected, the ZMII bridge generates a 2.5MHz clock. When 100Mbps is selected, the ZMII bridge generates a 25MHz clock. ZMII_SSR also controls whether detected collisions are signalled to the PHY.

Figure 24-5. ZMII Speed Selection Register (ZMII0_SSR)

0		Reserved	
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is asserted. 1 EMAC0 collision indication signal is deasserted.	
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SSR[SP0] does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is asserted. 1 EMAC1 collision indication signal is deasserted.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SSR[SP1] does not override IPG status field. 1 ZMII0_SSR[SP1] overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8:31		Reserved	

24.4.3 ZMII SMII Status Register (ZMII0_SMIISR)

ZMII_SMIISR contains the status transferred during the last inter-packet gap in the SMII interface, at the moment it is read. This register is used only if SMII interfaces are enabled. The register bits pertain to EMAC0RxD and EMAC1RxD SMII inter frame status used to convey packet data, RX_ER, and PHY status and are valid only when RX_DV (receive data valid) is set to 0.

Figure 24-6. ZMII SMII Status Register (ZMII0_SMIISR)

0	E01	EMAC0RxD Set to 1	
1	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
2	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
3	E0J	EMAC0RxD Jabber 0 OK 1 Error	
4	E0L	EMAC0RxD Link 0 Down 1 Up	
5	E0D	EMAC0RxD Duplex 0 Half 1 Full	
6	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
7	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
8	E11	EMAC1RxD Set to 1	
9	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
10	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
11	E1J	EMAC1RxD Jabber 0 OK 1 Error	
12	E1L	EMAC1RxD Link 0 Down 1 Up	
13	E1D	EMAC1RxD Duplex 0 Half 1 Full	
14	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	

15	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
16:31		Reserved.	

25. Ethernet Media Access Controllers

The PPC440EP provides two Ethernet media access controllers (EMACs) that are generic implementations of the Ethernet Media Access Control (MAC) protocol complying with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. Both EMACs support half-duplex (CSMA/CD) and full-duplex operation for 10-Mbps and 100-Mbps operations.

Both EMACs are implemented identically, with the exception of register addresses. Because both EMACs operate identically, the rest of this chapter describes a single EMAC. The EMACs are referred to as EMAC0 and EMAC1. Except in the register summary tables in *EMAC Registers* on page 609, the EMAC registers are prefixed "EMACx_" to denote the identical implementation of registers in each EMAC.

Each EMAC provides two on-chip peripheral bus (OPB based 32-bit bidirectional) slave interfaces. The first OPB interface provides access to the EMAC configuration and status registers. The PLB/OPB bridge enables the processor core to access these registers.

The second EOPB slave interface is used to exchange packet information with the memory access layer (MAL). The MAL is a multi-channel, intermediate hardware layer that resides between packet-based communication cores (such as EMAC) and external memory (such as SDRAM or SRAM). The MAL transfers packet information and status between the EMACs and external memory separately for each of the three EMAC channels (one receive and two transmit). Software (a device driver) maintains a buffer descriptor ring and a set of data buffers in external memory for each channel, and manages the exchange of packet data between the data buffers and the software protocol.

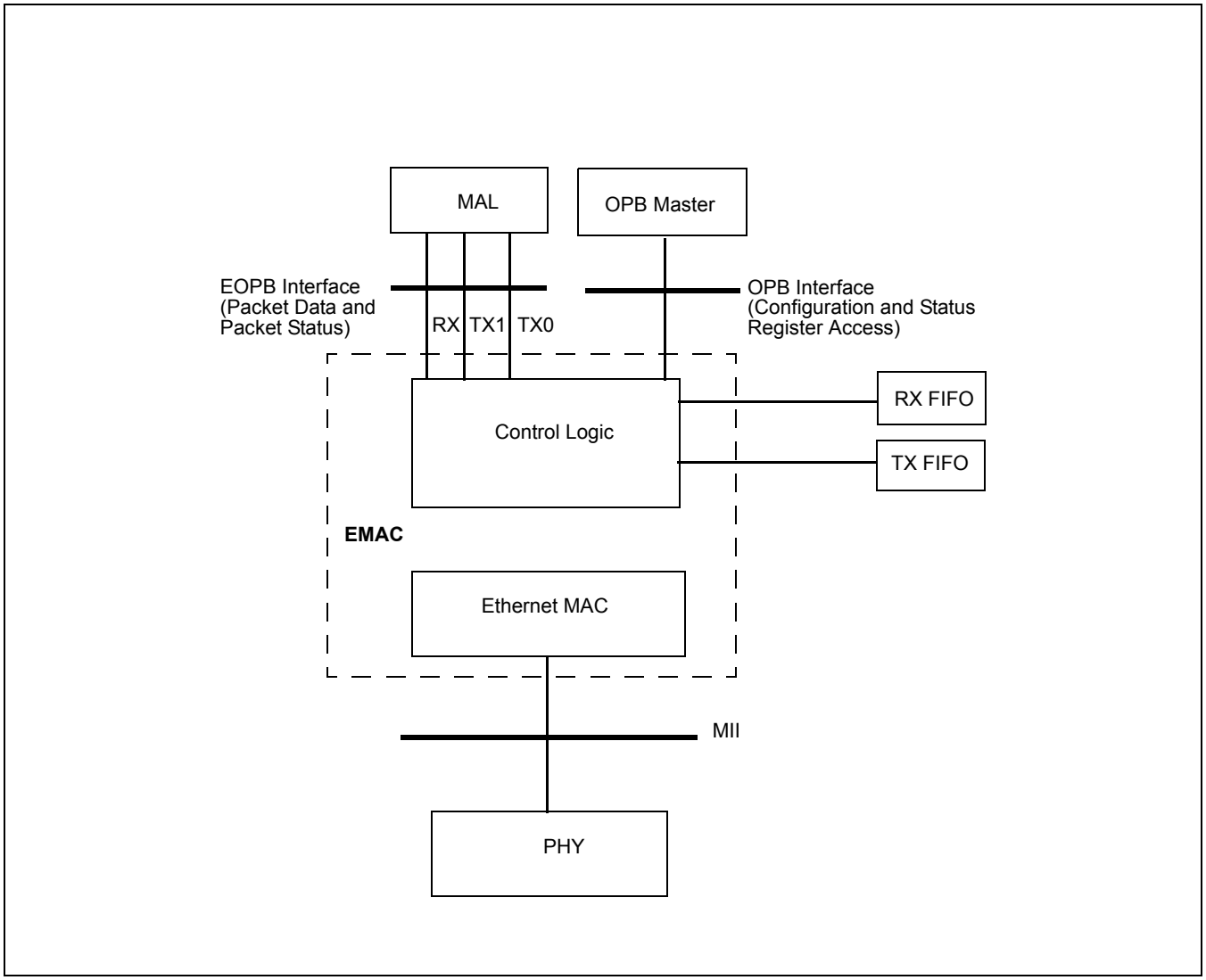
The MAL performs functions such as arbitration between service requests, handling the buffer descriptor memory structure, updating the descriptor status/control fields at the end of packet transfer, and so on. EMAC supports unlimited burst length transactions on the MAL interface.

Each EMAC uses a media independent interface (MII) to communicate with the Z media independent interface (ZMII) bridge. The ZMII bridge, described in detail in *EMAC to PHY Interface Bridge* on page 575 communicates with standard physical devices (PHYs). The ZMII bridge supports the media independent interface (MII), the reduced pin count reduced media independent interface (RMII), and serial media independent interface (SMII). Details of PHY communication are in *EMAC to PHY Interface Bridge* on page 575.

The EMAC uses independent receive and transmit FIFOs. Programmable FIFO thresholds minimize overflows and underruns, and can launch integrated IEEE 802.3x pause packets for flow control.

Figure 25-1 illustrates an EMAC in a typical Ethernet application.

Figure 25-1. EMAC in a Typical Ethernet Application



Note: Each EMAC connects to two OPBs: the OPB through which an OPB master configures the EMAC, and the EOPB through which the MAL and EMAC pass packets. EMAC transmit channels operate independent of receive channel operations.

25.1 EMAC Features

EMAC features:

- Dual speed (10/100 Mbps) CSMA/CD (half-duplex) and full-duplex Ethernet MAC complying with ANSI/IEEE Std. 802.3 and IEEE 802.3u supplement.
- Automatic source address insertion or replacement for transmitted packets is a programmable option.
- Automatic stripping of frame padding bytes and frame check sequence (FCS) is a programmable option.

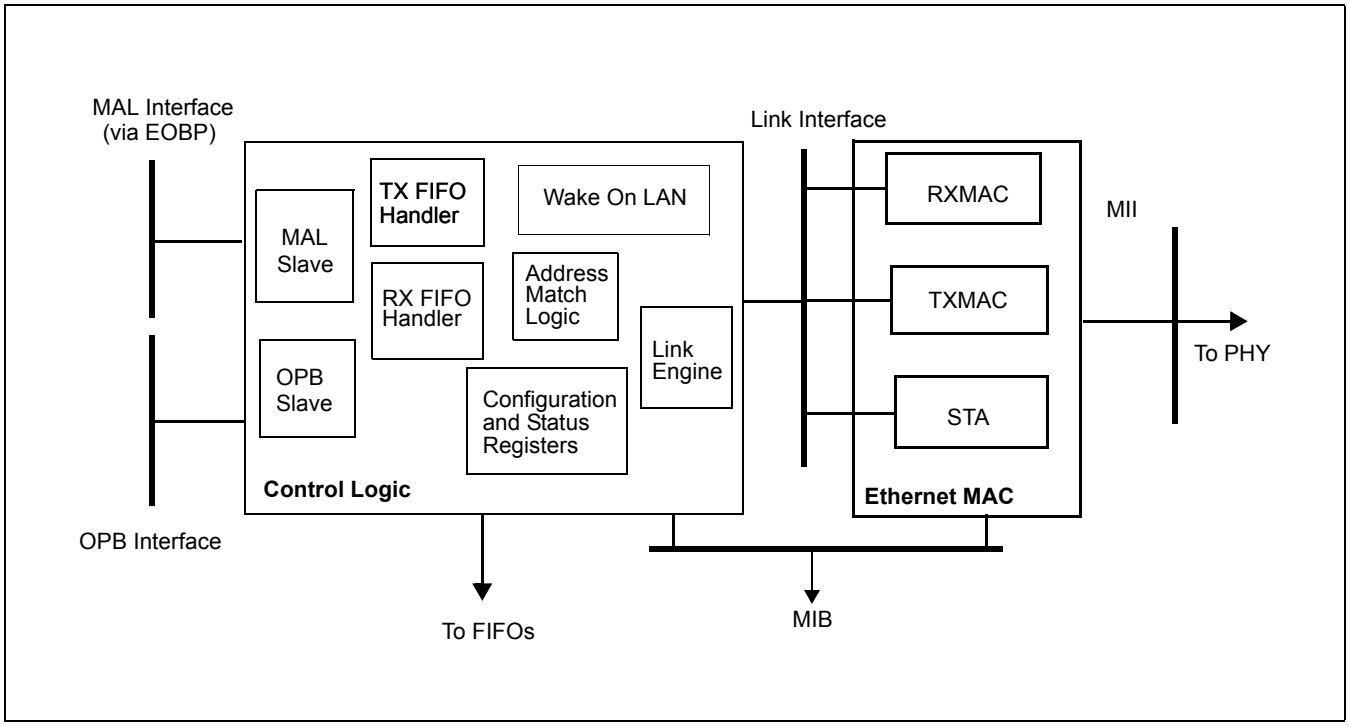
  When padding bytes are stripped, the padding and FCS field are removed. FCS stripping removes only the FCS field.

- FCS control for transmit/receive packets.
- Access to registers with support for burst processing.
- MAL for packet moving having one-cycle MAL slave latency.
- Independent, large (2 KB) transmit and up to (4 KB) receive FIFOs with programmable thresholds to minimize overruns and underruns.
- Multiple packet handling in transmit and receive FIFOs.
- Unicast, multicast, broadcast, and promiscuous address filtering capabilities.
- Two 64-bit hash filters for unicast and multicast packets.
- Automatic retransmission of collided packets.
- Rejection of runt packets before providing them to MAL.
- MII for connection to a variety of PHY layer devices.
- Programmable inter-packet gap to enable tuning for better system performance.
- Compliance with IEEE 802.3x standard packet-based flow control, including self-assembled control pause packet transmitting.
- Support for VLAN tag ID in compliance with IEEE Draft 802.3ac/D1.0 standard.
- VLAN tag insertion or replacement for transmit packets is a programmable option.
- Wake on LAN (WOL) handling.
- Programmable internal and external loop-back capabilities.
- Extensive error/status vector generation for each processed packet.
- Power management using a clock and power management (CPM) unit.

25.2 EMAC Operation

The EMAC hardware components and its internal structure are illustrated in Figure 25-2. The EMAC is connected to the ZMII bridge via the MII interface defined by EMAC configuration.

Figure 25-2. Internal EMAC Structure



The control logic sub-block implements the following functions:

- OPB slave device
- MAL slave device
- FIFO management logic
- Ethernet address and pause packet match logic
- Register file for FIFOs and Ethernet MAC handler management
- Logic for support of WOL technology
- Link engine

The Ethernet MAC sub-block implements the following functions:

- Transmit MAC Handler (TXMAC)
- Receive MAC Handler (RXMAC)
- MII management function unit (STA)

The above functions are described in the following sections.

25.2.1 MAL Slave Logic

The MAL slave (MALS) logic controls MAL transactions. MALS transfers TX and RX data between MAL and the OPB on one side, and the EMAC FIFO handlers on the other side. MALS is a dedicated MAL slave.

25.2.2 OPB Slave Logic

The OPB slave (OPBS) logic controls all OPB transactions between the processor core and the EMAC configuration and status registers.

25.2.3 FIFO Management Logic

The FIFO management logic is used for data interchange handling with the attached receive (RX FIFO) and transmit (TX FIFO modules.

25.2.4 Ethernet Address Match Logic

Address match logic checks the destination address of received packets against a set of predefined addresses specified by the current address filtering mode. EMAC contains one unicast (individual address) register, two hash tables for filtering individual and group address, and logic for detecting broadcast address (all ones). EMAC supports promiscuous mode and multicast promiscuous mode.

This logic also checks the destination address of the incoming packet against a special multicast address used for control (pause) packet recognition.

All checks for address matching are performed only after the entire destination address field is received (except for promiscuous and multicast promiscuous modes).

25.2.5 Configuration and Status Registers

Configuration and status registers define the EMAC configuration and reflect error/status of recent transmitted or received packets.

25.2.6 Wake On LAN Logic

EMAC supports Wake On LAN (WOL) technology, an industry standard described in the *Wired for Management (WFM)* specification. This technology allows a sleeping or powered-off network node to be awakened with a special packet called a Magic Packet. In the PPC440EP, with WOL mode enabled, the EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1).

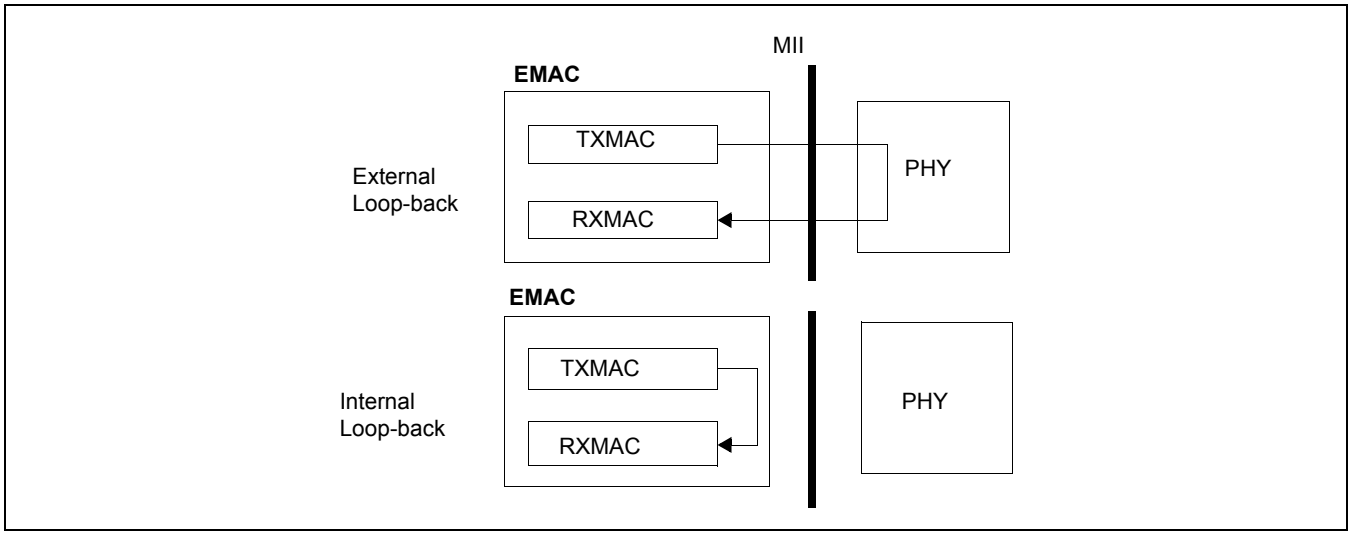
25.2.7 Ethernet MAC

The Ethernet MAC logic supports media independent interface (MII).

25.2.8 EMAC Loopback Modes

EMAC supports the external and internal loop-back modes illustrated in *Figure 25-3*.

Figure 25-3. EMAC Loopback Modes



External loop-back mode uses the PHY device. To EMAC, external loop-back is identical to full-duplex operation. Configuring EMAC to operate in a full-duplex mode enables external loop-back. EMAC does not need an external loopback configuration signal.

In internal loopback mode, data from the EMAC transmit channel is routed to the EMAC receive channel. The loopback mode functions correctly with or without a connected PHY. In internal loopback mode, EMAC does not activate (monitor) any MII signals. Transmit channel signals are buffered internally to the receive channel. However, if internal loopback is used without a PHY, the EMAC transmit and receive clocks must be provided by another means. In internal loopback mode, the EMAC transmit clock and receive clock must be sourced from a single clock.

25.2.9 Packet Rejection

EMAC3 has an external input called RejectPkt (shared with USB2LS0). This external input allows the user's custom logic, attached to the MII interface, to specify to EMAC3 if the recently received packet should be accepted or rejected. The EMAC3 control logic evaluates the level of this signal on the cycle when the MRX_STATUS_VALID signal, coming from the GMAC core, is asserted. If RejectPkt is set to 1 in this cycle, the recently received packet is removed from the Receive FIFO. The user must connect the RejectPkt signal to GND if this option is not used.

The following registers are used in connection with this function. There are three registers for EMAC0 and EMAC1.i

Table 25-1. Packet Reject Registers

Register	Offset	Access	Description
SDR0_EMAC0RXST	0x4301	R/W	EMAC0 RX Status Register
SDR0_EMAC0TXST	0x4302	R/W	EMAC0 RX Status Register
SDR0_EMAC0REJCNT	0x4303	R	EMAC0 RX Packet Reject Counter
SDR0_EMAC1RXST	0x4304	R/W	EMAC1 RX Status Register

Table 25-1. Packet Reject Registers

Register	Offset	Access	Description
SDR0_EMAC1TXST	0x4305	R/W	EMAC1 RX Status Register
SDR0_EMAC1REJCNT	0x4306	R	EMAC1 RX Packet Reject Counter

25.2.9.1 EMACx RX Status Register

Figure 25-4. EMACx RX Status Register(SDR0_EMACxRXST)

<table>
<tr><td>0:1</td><td></td><td>Reserved</td><td></td></tr>
<tr><td>2</td><td>FOR</td><td>RXFIFOOverrun</td><td>The current frame was aborted because an underrun. Flow of received data applied to the FIFO was detected, and the data was invalidated since there was no empty space in the Receive FIFO.</td></tr>
<tr><td>3</td><td>BC</td><td>BroadcastAddr</td><td>The frame was received with the broadcast destination address and at least 6 bytes are received.</td></tr>
<tr><td>4</td><td>MC</td><td>MulticastAddr</td><td>The frame was received with a group destination address other than broadcast and at least 6 bytes are received.</td></tr>
<tr><td>5</td><td>UC</td><td>UnicastAddr</td><td>The frame was received with a unicast destination address and at least 6 bytes are received.</td></tr>
<tr><td>6:8</td><td>UPR</td><td>User Priority Field</td><td>Provides the content of User Priority field (bits 7:5 respectively) from received VLAN Tagged frame.</td></tr>
<tr><td>9</td><td>VLAN</td><td>RX VLAN Tagged Frame</td><td>The currently received frame was a VLAN Tagged frame.</td></tr>
<tr><td>10</td><td>LOOP</td><td>Received in Loop-Back Mode</td><td>The currently received frame is an echo of frame transmitted by TXMAC block.</td></tr>
<tr><td>11</td><td>UOP</td><td>Rx Unsupported Opcode</td><td>The currently received frame was a control frame with an unsupported opcode.</td></tr>
<tr><td>12</td><td>CPF</td><td>RX Control Pause Frame</td><td>The currently received frame was a pause frame.</td></tr>
<tr><td>13</td><td>CF</td><td>RX Control Frame</td><td>The currently received frame was a control frame.</td></tr>
<tr><td>14</td><td>MSIZ</td><td>1024–MaxSize bytes received</td><td>Indicates that the currently received frame (including bad frames) has a byte length between 1024 and maxFrameSize. This is defined by the Jumbo Packets support enabled and VLAN support enabled bits as follows:
<table>
<tr><th>Jumbo Packets Support Enabled</th><th>VLAN Support Enabled</th><th>maxFrameSize (Bytes)</th></tr>
<tr><td>0</td><td>0</td><td>1518</td></tr>
<tr><td>0</td><td>1</td><td>1522</td></tr>
<tr><td>1</td><td>0</td><td>9018</td></tr>
<tr><td>1</td><td>1</td><td>9022</td></tr>
</table></td></tr>
<tr><td>15</td><td>1023</td><td>512–1023 bytes received</td><td>The currently received frame (including bad frame) was between 512 and 1023 bytes length (excluding framing bits but including FCS bytes).</td></tr>
<tr><td>16</td><td>511</td><td>256–511 bytes received</td><td>The currently received frame (including bad frame) was between 256 and 511 bytes length (excluding framing bits but including FCS bytes).</td></tr>
</table>

17	255	128–255 bytes received	The currently received frame (including bad frame) was between 128 and 255 bytes length (excluding framing bits but including FCS bytes).
18	127	65–127 bytes received	The currently received frame (including bad frame) was between 65 and 127 bytes in length (excluding framing bits but including FCS bytes).
19	64	64 bytes received	The currently received frame (including bad frame) was 64 bytes in length (excluding framing bits but including FCS bytes).
20	RUNT	Runt frame	The frame length was less than MinFrameSize (64 bytes, excluding the preamble and SFD bytes) and a Short Event was not detected.
21	SEVT	Short event	The duration of PHY_RX_DV signal was less than the Short-EventMaxTime constant. For this indication, GMAC always assumes that the preamble and SFD fields contain eight bytes regardless of the number actually received.
22	AERR	Alignment error	The received frame is not an integral number of octets in length. This is applicable only to 10/100 Mbps operation.
23	SERR	Received with symbol error	For 100 Mbps operation, indicates a case when a valid carrier was present and there was at least one occurrence of Data Reception Error. For 1000 Mbps operation, indicates a case when the receiving media is non-idle (a carrier event) for a period of time greater then or equal to slotTime for half duplex, or greater then or equal to minFrameSize for full-duplex, and during which there was at least one occurrence of an event that causes the PHY to indicate Data reception error on the GMII.
24		Reserved	
25	BURST	Received burst	Indicates that carrier event (from the first byte of DA until the end of transmission) was equal or greater than slotTime. This is applicable only in HDX mode at 1000 Mbps.
26	FL2	Frame is too long	The length of the received frame exceeded the maximum allowed value (maxFrameSize): - 1518 octets for standard frame (checked only when the length/type field of the transmitted frame contains a length value and REG_JUMBO_EN is inactive) - 1522 octets for VLAN tagged frame (checked only when the length/type field of the transmitted frame contains a length value and REG_JUMBO_EN is inactive) - 9018 octets for a jumbo frame (with no VLAN support) - 9022 octets for a VLAN tagged jumbo frame
27	OERR	Out of range length error	The received frame has a length field value greater than the maximum allowed LLC data size (greater than 1500 and less than 1536).
28	IERR	In range length error	The content the length field in the received frame is less or equal to the maximum allowed MAC Client data size (1500 bytes).
29	LOST	Frame lost due to internal EMAC receive error	Frame reception was interrupted internally within the EMAC.
30	BFCS	Bad FCS in the received frame	Received frame had an FCS value which does not match the FCS value calculated by the EMAC.

31	RXOK	Receive OK	None of the following events occurred during a receive operation: - Receive FIFO Handler interrupted receive process - FCS error - Length mismatch error - Too long frames - Alignment errors - Received with symbol error - Short event
Note: If the receive frame is interrupted internally within the EMAC, then all bits except bits 2, 26, and 29 are in an unpredictable state.			

25.2.9.2 EMACx TX Status Register

Figure 25-5. EMACx TX Status Register(SDR0_EMACxTXST)			
0:5		Reserved	
6	FUR	TXFIFOUnderrun	The current frame was aborted because an underrun. The Data indication from the FIFO was not valid in time to allow continuous data transmission on the MII/GMII interface.
7	BC	BroadcastAddr	The frame was transmitted to the broadcast destination address.
8	MC	MulticastAddr	The frame was transmitted to a group destination address other than broadcast.
9	UC	UnicastAddr	The frame was transmitted to a unicast destination address.
10	FP	Frame paused by control packet	Indicates that the currently transmitted frame was delayed due to a pause packet received by EMAC.
11	BFCS	Bad FCS in the transmitted frame	The transmitted frame had an FCS which does not match the FCS calculated by EMAC. EMAC asserts this indication only if the frame being transmitting ended without interference (there was no collision or stop request during transmission).
12	CPF	TX control pause frame	The currently transmitted frame was a pause frame. The control packet was self-assembled by EMAC.
13	CF	TX control frame	The currently transmitted frame was a control frame. The control packet was self-assembled by EMAC.
14	MSIZ	1024–maxsize bytes transmitted	The currently transmitted frame was between 1024 and 1518 (1522 in case of VLAN mode) bytes length (excluding framing bits but including FCS bytes).
15	1023	512–1023 bytes transmitted	The currently transmitted frame was between 512 and 1023 bytes length (excluding framing bits but including FCS bytes).
16	511	256–511 bytes transmitted	The currently transmitted frame was between 256 and 511 bytes length (excluding framing bits but including FCS bytes).
17	255	128–255 bytes transmitted	The currently transmitted frame was between 128 and 255 bytes length (excluding framing bits but including FCS bytes).
18	127	65–127 bytes transmitted	The currently transmitted frame was between 65 and 127 bytes in length (excluding framing bits but including FCS bytes).
19	64	64 bytes transmitted	The currently transmitted frame is 64 bytes in length (excluding framing bits but including FCS bytes).

20	SQE	SQE indication	This bit is applicable only for the HDX 10Mbps. Indicates that Signal Quality Error test has failed. Following the end of a successful EMAC transmission (ended with no collision), EMAC expects the PHY to assert the collision signal until the end of the first part of the inter frame-gap (IFG1). If collision was not asserted, EMAC indicates a signal quality error. When the IGNORE_SQE_TEST bit in Mode Register 1 is set, SQE test is not performed.
21	LOC	Loss of carrier sense	During transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all. This is applicable only in HDX mode. Returns 0 in FDX mode.
22:23		Reserved	
24	IERR	EMAC internal error	Transmit was interrupted internally within EMAC.
25	EDF	Excessive deferral	The current frame has been deferred for an excessive period of time. The value of this period in bits is calculated as follows: - For 10 and 100Mbps operation: 2 x (maxFrameSize x 8) bit times - For 1000Mbps operation: 2 x (burstLimit + maxFrameSize x 8 + headerSize) bit times. This is applicable only in half-duplex mode.
26	ECOL	Excessive collisions	The current frame transmission had ended with a collision at the 16th consecutive attempt. This is applicable only in half duplex mode.
27	LCOL	Late collision	The frame collided with other data outside of the collision window. This is applicable only in half duplex mode.
28	DFFR	Deferred frame	The current frame transmission is deferred due to activity on the medium on its first transmission attempt. (Note that if EMAC waits to the end of the IFG period without indicating an active medium, the frame is not considered as a deferred frame). This is applicable only in half duplex mode.
29	MCOL	Multiple collision frame	The transmitted frame collided with other data on the bus more than once, but less than 16 times. This is applicable only in half duplex mode.
30	SCOL	Single collision frame	The transmitted frame collided with other data on the bus. This is applicable only in half-duplex mode.
31	TXOK	Transmit OK	None of the following events was discovered during transmission: - Collision - Late collision - Lost of carrier sense - Excessive deferral - Bad FCS - TX FIFO Handler interrupted transmit process - SQE error (for HDX 10Mbps only)
Note: If the transmit frame is interrupted internally within the EMAC, then all bits except bits 6, 24, 25, 26, and 27 are in an unpredictable state.			

25.2.9.3 EMACx RX Packet Reject Counter

Figure 25-6. EMACx Reject Count Register (SDR0_EMACxREJCNT)

0:31	RC	Reject counters.	Count the number of packets rejected.

25.3 EMAC Transmit Operation

The transmit part of EMAC handles packet transmission from the MAL device to the MII interface. At the end of a transmission process, EMAC provides a status/error word which allows monitoring the transmission operation.

EMAC implements dual MAL transmit channels (two transmit channels are allocated within the MAL) to support efficient use of the transmit FIFO. Both channels share resources inside the EMAC. The transmit channels can be configured to independently request packets from the MAL and drive them into the transmit FIFO, or to function as a single channel (in dependent mode).

25.3.1 Arbitration Between TX Channels

Because the transmit channels (referred to as TX Channel 0 and TX Channel 1) for each EMAC drive data into one FIFO, they cannot request packet data from the MAL at the same time. The MAL ensures that only one EMAC transmit channel for each EMAC is active at a given time.

25.3.2 Independent Mode

In independent mode, each EMAC transmit channel independently requests packets from the MAL. Each channel can be configured to work in either single packet or multiple packet mode.

In single packet mode, EMACx_MR1[TR0] = 00 and EMACx_MR1[TR1] = 00. The channel requests one packet from the MAL and resets EMACx_TMR0[GNP0, GNP1] as appropriate. The channel asks for service again only after EMACx_TMR0[GNP0] = 1 or EMACx_TMR0[GNP1] = 1 (set by the device driver).

In multiple packet mode, EMACx_MR1[TR0] = 01 and EMACx_MR1[TR1] = 01. After the channel finishes transferring a packet, the channel asks the MAL for the next packet as soon as the other channel is in its Idle Phase and there is enough room in the FIFO. The channel continues to request more packets until one of the following events occur:

- The channel receives notification from the MAL that the next buffer descriptor is not marked ready for transmission. When this occurs, the channel sets EMACx_TMR0[GNP0, GNP1] = 0, as appropriate, and waits for software to reactivate the channel by setting EMACx_TMR0[GNP0] = 1 or EMACx_TMR0[GNP1] = 1.
- A transmit error or signal quality error (SQE) occurs and the corresponding interrupt is not masked in the EMACx_ISER. After such an error, the channel sets EMACx_TMR0[GNP0, GNP1] = 0, as appropriate, and sets EMACx_ISR[DB0] = 1 or EMACx_ISR[DB1] = 1 (the EMACx_ISR field that is set depends on which channel is active) and the corresponding EMACx_ISR error. The channel does not request service again until EMACx_TMR0[GNP0] = 1 or EMACx_TMR0[GNP1] = 1 and EMACx_ISR[DB0] = 0 or EMACx_ISR[DB1] = 0 (again, depending on channel).

In independent mode, if both channels are configured to work in multiple packet mode and both EMACx_TMR0[GNP0] = 1 and EMACx_TMR0[GNP1] = 1 at the same time, the channels operate in a sequential repeating manner as long as no errors occur.

25.3.3 Dependent Mode

In dependent mode, EMACx_MR1[TR0] = 10 and EMACx_MR1[TR1] = 10. The two TX channels act as if they were one channel, sharing EMACx_TMR0[GNPD]. When EMACx_TMR0[GNPD] = 1, the channel specified by EMACx_TMR0[FC] starts requesting MAL service. Then, both channels continue to request packets from the MAL in an alternating, sequential, repeating manner, until one of the following occurs:

- One of the channels receives notification from the MAL that the next buffer descriptor is not marked ready for transmission. When this occurs, the EMAC clears EMACx_TMR0[GNPD]. At this point, neither channel requests a packet from the MAL until EMACx_TMR0[GNPD] = 1. The first channel to request a new packet

from MAL when this occurs is the channel that received notification from MAL that no packets were ready to transmit (regardless of the setting of EMACx_TMR0[FC]). Further requests continue in an alternating, repeating manner.

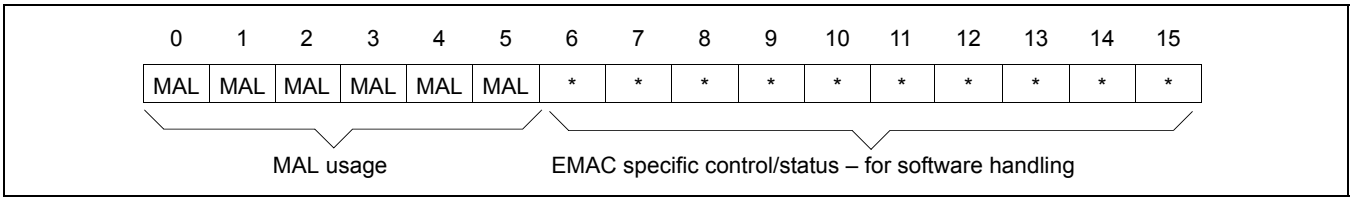
- Either a transmit error or an SQE occurs on one of the channels and the corresponding interrupt is not masked in the EMACx_ISER. One of the following scenarios can occur.
  - If the other channel has not yet requested MAL service when the channel for which the error occurred receives notification from the MAL that the transmit operation has completed, EMACx_TMR0[GNPD] is cleared (EMACx_TMR0[GNPD] = 0) and the EMACx_ISR[DBDM] is immediately set (EMAC_ISR[DBDM]=1).
  - If the other channel was receiving data from the MAL, it initiates early termination. If a second packet was being transmitted on the media, it is stopped. In these cases, EMACx_TMR0[GNPD] is cleared (EMACx_TMR0[GNPD] = 0) and the EMACx_ISR[DBDM] = 1 only after notification from the MAL that the transmit operation has completed has been received for the second channel.

At this point, neither channel activates a request to the MAL until EMACx_TMR0[GNPD] = 1 and EMACx_ISR[DBDM] = 0. The channel specified by EMACx_TMR0[FC] is the first to request service from MAL. Subsequent requests continue in an alternating, sequential manner.

25.3.3.1 MAL TX Descriptor Control/Status Field

For each transmitted packet, the MAL uses the descriptor control/status field of the buffer descriptor to provide an EMAC with control information (write), and to obtain packet status from the EMAC after transmission is complete (read). Software writes the control bits in the buffer descriptor before packet transmission, and reads the status bits from the buffer descriptor after packet transmission has completed. See *Buffer Descriptor* on page 550 for more information on the buffer descriptor structure.

Figure 25-7. MAL TX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See *TX Status/Control Field Format* on page 557	R
TX Control Information (Write Access)			
6	Generate FCS	0 FCS is not generated by EMAC. 1 EMAC calculates and adds the FCS field to the packet to be transmitted.	W
7	Generate padding	0 Padding is not generated by EMAC. 1 EMAC adds the padding field to the packet to be transmitted (only when Generate FCS is also set).	W
8	Insert source address	0 EMAC will not insert source address. 1 EMAC inserts the source address field into the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
9	Replace source address	0 EMAC will not replace source address. 1 EMAC replaces the source address field in the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W

Bits	Bit Name	Bit Description	Mode
10	Insert VLAN Tag	0 EMAC will not insert a VLAN tag. 1 EMAC inserts the VLAN Tag field into the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
11	Replace VLAN Tag	0 EMAC will not replace the VLAN tag. 1 EMAC replaces the VLAN Tag field in the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
TX Status Information (Read Access)			
6	Bad FCS on transmitted frame	0 FCS was correct in the transmitted packet. 1 Indicates that a bad FCS was found in the transmitted packet.	R
7	Bad previous packet in depen-dent mode	0 Packet transmission OK. 1 Indicates that a Descriptor Error, Transmit Error, or SQE Error occurred in the previously transmitted frame. This bit will only be activated in dependent mode.	R
8	Loss of carrier sense	0 No loss of carrier. 1 During the transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all. Applicable only in half duplex mode.	R
9	Excessive deferral	0 No excessive deferral. 1 Indicates that the current frame has been deferred for an excessive period of time. Applicable only in half duplex mode. The value of this period in bit times is calculated in the following ways: For 10/100 Mbps operation it is: 2 x (maxFrameSize x 8) bit times.	R
10	Excessive collisions	0 Less than 16 collisions. 1 Indicates that the current frame transmission had ended with a collision on the 16th consecutive attempt. Applicable only in half-duplex mode.	R
11	Late collision	0 No late collision. 1 Frame collided outside of the collision window. Applicable only in half-duplex mode.	R
12	Multiple collision	0 More than 1 but less than 16 collisions did not occur. 1 Transmitted frame collided more than once but less than 16 times. Applicable only in half-duplex mode.	R
13	Single collision	0 Single collision did not occur. 1 Activates if transmitted frame collided once. Applicable only in half-duplex mode.	R
14	Underrun	0 Underrun did not occur. 1 Frame transmission was aborted because of underrun; data from the Transmit FIFO was not valid in time to allow continuous data transmission on the MII interface.	R
15	SQE	0 Signal Quality Error did not occur. 1 Signal Quality Error test failed during packet transmission. Applicable only in half -duplex mode during 10 Mbps operation.	R

25.3.3.2 Early Packet Termination during Transmit

EMAC can initiate early packet termination during transmit, terminating packet transmission before MAL finishes transferring all packet data from memory to the EMAC transmit FIFO. Early packet termination is typically used when error conditions force the EMAC to abort a transmission.

EMAC performs early termination on the MAL interface if any of the following conditions occur:

- Underrun in the transmit FIFO.

- Excessive collisions.
- Excessive deferral.
- Late collision.

25.3.3.3 Empty Packets

EMAC treats empty packets as if a normal packet had been written, but does not write data to the transmit FIFO. A status word of all 0s is returned after an empty packet. EMAC expects that for word-aligned packets, MAL activates the related word transfer indication during the last data transfer, rather than providing an empty packet indication.

25.3.3.4 Automatic Retransmission of Colliding Packets

EMAC automatically retransmits packets that collide on the MII interface. The transmit FIFO always preserves the first 64 bytes of a packet until it receives an indication that the collision window has passed. Otherwise, if a collision was detected within the collision window, the packet is retransmitted without a new request from MAL.

25.3.3.5 Inter-Packet Gap (IPG) Tuning

EMAC supports user-programmable IPG length using the EMACx_IPGVR register, which contains one-third of the IPG value. Changing the contents of EMACx_IPGVR enables the user to adjust the fairness or aggressiveness of EMAC on the medium. Programming a lower number (but not less than four) causes EMAC becomes more aggressive on the media. This can result in EMAC capturing the network by forcing less aggressive nodes to defer. Programming a larger number of bit times causes EMAC to becomes less aggressive on the network; it might defer more often than normal. EMAC performance might decrease as the IPG period is increased from the default value, but the resulting behavior can improve media performance by reducing the occurrence of collisions.

25.3.3.6 Full-Duplex Operation

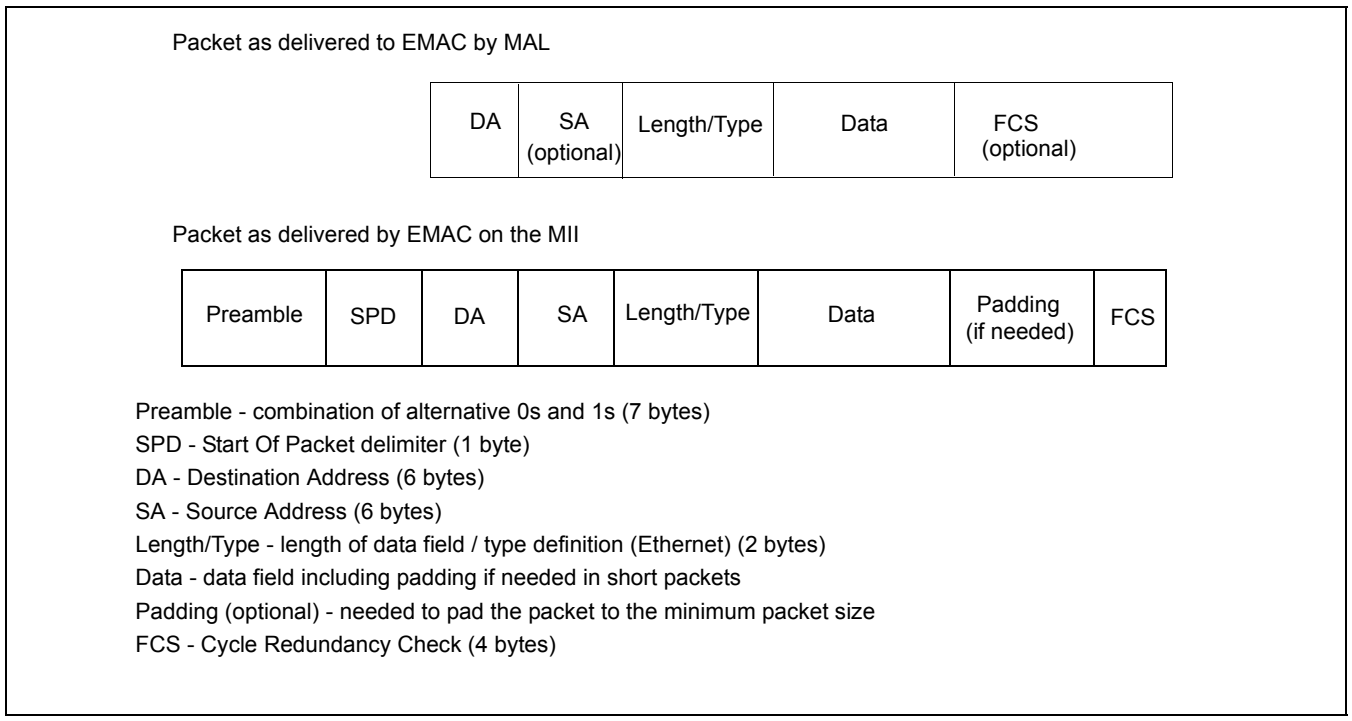
Full-duplex operation allows simultaneous transmit and receive activity on the MII interface. Software can set EMACx_MR1[FDE] = 1 to enable full-duplex mode. During full-duplex operation, the following changes occur in EMAC functionality.

- Transmission is not deferred while receive is active.
- The IPG counter, which controls transmit deferral during the IPG between back-to-back transmits, is started when transmit activity for the first packet ends, instead of when transmit and carrier activity end.
- SQE test is not performed.
- Collision indication is ignored.

25.3.3.7 Packet Content Configuration Options

EMAC can modify the content of the packet coming from the MAL before issuing it to the MII interface. *Figure 25-8* illustrates possible changes in the transmit packet format.

Figure 25-8. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)



The following options, unless mutually exclusive, can be set for each packet, and can be provided as a part of command write transactions from MAL (see *MAL TX Descriptor Control/Status Field* on page 594).

- Automatic data padding for short transmit packets

  EMAC pads the transmit packet with extra bytes between the data and the FCS field to reach a total length of 64 bytes (including FCS). This feature is supported only when the packet coming from the transmit FIFO does not contain the FCS. Automatic padding enables software to avoid sending padding as a part of the packet data field, and, therefore, reduces the amount of data transferred on the system bus during the short packet transmission.

- Source address insertion

  EMAC adds the source address (SA) field to the transmitted packet. EMAC uses the contents of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) registers for the source address value. *This option is mutually exclusive with source address replacement.*

- Source address replacement

  EMAC replaces the source address received from the transmit FIFO with the contents of EMACx_LSAH and EMACx_LSAL. *This option is mutually exclusive with source address insertion*.

- Add four FCS bytes

  EMAC calculates the FCS for the transmitted packet. The FCS is appended to data coming from the Transmit FIFO or padding bytes (if added).

- VLAN Tag insertion

EMAC adds the content of the VLAN Tag field to the transmitted packet (see *VLAN Support* on page 604). EMAC uses the content of the VLAN TPID (EMACx_VTPID) and VLAN TCI (EMACx_VTCI) registers for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag replacement*.

- VLAN Tag replacement.

  EMAC replaces the content of the VLAN Tag field in the transmitted packet. EMAC uses the contents of EMACx_VTPID and EMACx_VTCI for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag insertion*.

Table 25-2 summarizes the possible options for adding FCS and SA to the transmitted packet.

Table 25-2. FCS/SA Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	**Insert SA**	**Replace SA**	**Add FCS**	**Add SA**	**Replace SA**
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

Table 25-3 summarizes the possible options for adding FCS and padding to the transmitted packet.

Table 25-3. FCS/Pad Enable - Possible Configurations

Configuration Options		EMAC Action	
Generate FCS	**Generate Pad**	**Add FCS**	**Add Pad**
0	Don't care	N	N
1	0	Y	N
1	1	Y	Y

Table 25-4 summarizes the possible options for adding FCS and VLAN Tag to the transmitted packet.

Table 25-4. FCS/VLAN Tag Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	**Insert VLAN Tag**	**Replace VLAN Tag**	**Add FCS**	**Insert VLAN Tag**	**Replace VLAN Tag**
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

25.4 EMAC Receive Operation

The receive part of EMAC is responsible for receiving packets coming from the physical layer (PHY) device (via the MII) and forwarding them to the receive channel of the attached MAL. At the end of the reception process, EMAC provides a status/error word that enables software to monitor the receive operation.

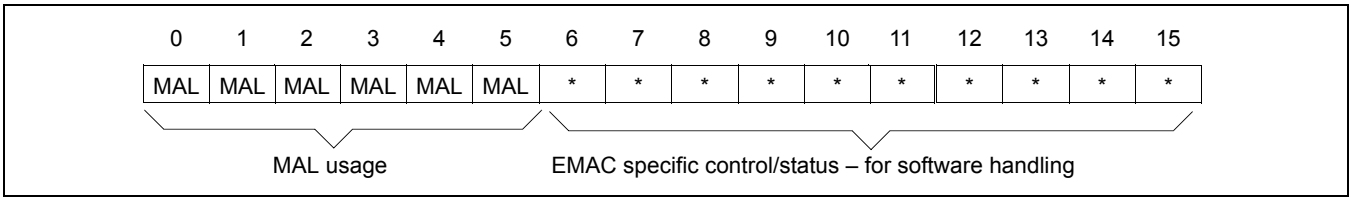
25.4.1 EMAC – MAL RX Packet Transfer Flow

EMAC initiates request for service from MAL when the number of occupied entries in the receive FIFO reaches the low water mark specified in EMACx_RWMR[RLWM].

25.4.2 MAL RX Descriptor Status

For each packet that is received, MAL obtains status from EMAC after reception is complete, and writes this information into the buffer descriptor status/control field. Software uses this information to monitor the status of received packets. See *Buffer Descriptor* on page 550 for more information on the buffer descriptor structure.

Figure 25-9. MAL RX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See *RX Status/Control Field Format* on page 558.	R
RX Status Information (Read Access)			
6	Overrun Error	0 No overrun error. 1 EMAC detected an overrun error. An overrun error occurs if the flow of received data to the RX FIFO is corrupted because of insufficient empty space.	R
7	Pause Packet	0 Received packet is not a control pause packet. 1 Received packet is a control pause packet.	R
8	Bad Packet	0 No packet errors. 1 Early termination caused by packet error.	R
9	Runt Packet	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal greater than ShortEventMax Time constant and less than collision windows.	R
10	Short Event	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal was less than ShortEventMaxTime constant	R
11	Alignment Error	0 Received packet length OK. 1 Received packet length not an integral number of octets.	R
12	Bad FCS	0 FCS OK. 1 The FCS value does not match the FCS value calculated by EMAC.	R
13	Packet Too Long	0 Received packet length OK. 1 Received packet length exceeds maximum packet length. 1518 octets for standard packet 1522 octets for VLAN tagged packet	R

Bits	Bit Name	Bit Description	Mode
14	Out of Range Error	0 Received packet length field value OK. 1 Received packet length field value greater than maximum allowed LLC data size. The maximum allowed logical link control (LLC) data size is greater than 1500 and less than 1536.	R
15	In Range Error	Refer to *Table 25-5* for a description of conditions for activating this status bit.	R

Table 25-5. In Range Length Error Behavior for Various Packet Lengths

Programmed Length (Bytes)	Actual Length	EMAC Action
Less than 46 (42 if VLAN Tagged packet)	Differs from 46 (42 in case of VLAN Tagged packet)	In range length error is activated
Less than 46 (42 if VLAN Tagged packet)	46 (42 in case of VLAN Tagged packet)	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Equals the length field value	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Differs from the length field value	In range length error is activated

25.4.3 Early Packet Termination during Receive

Early packet termination occurs when packet reception is aborted by EMAC before the packet data transfer to MAL is completed.

EMAC performs early termination in the following cases.

- An overrun occurs in the receive FIFO
- Packet is too long and the Receive Oversize Packet option is not enabled (EMACx_RMR[ROP] = 0)

25.4.4 Discarding Packets During Receive

Received packets can be discarded if certain error conditions are detected. EMAC behavior depends on whether the packet to be discarded is already being output to MAL. If the packet containing the error is not being provided to MAL when the discard condition is detected, the packet is flushed from the Receive FIFO. In this case, EMAC does not provide status information to MAL. If the packet containing the error is already being output to MAL, EMAC initiates an early packet termination procedure, as described in *Early Packet Termination during Receive* on page 600

Each receive discard condition can be individually controlled using appropriate settings, as described in *Receive Mode Register (EMACx_RMR)* on page 615.

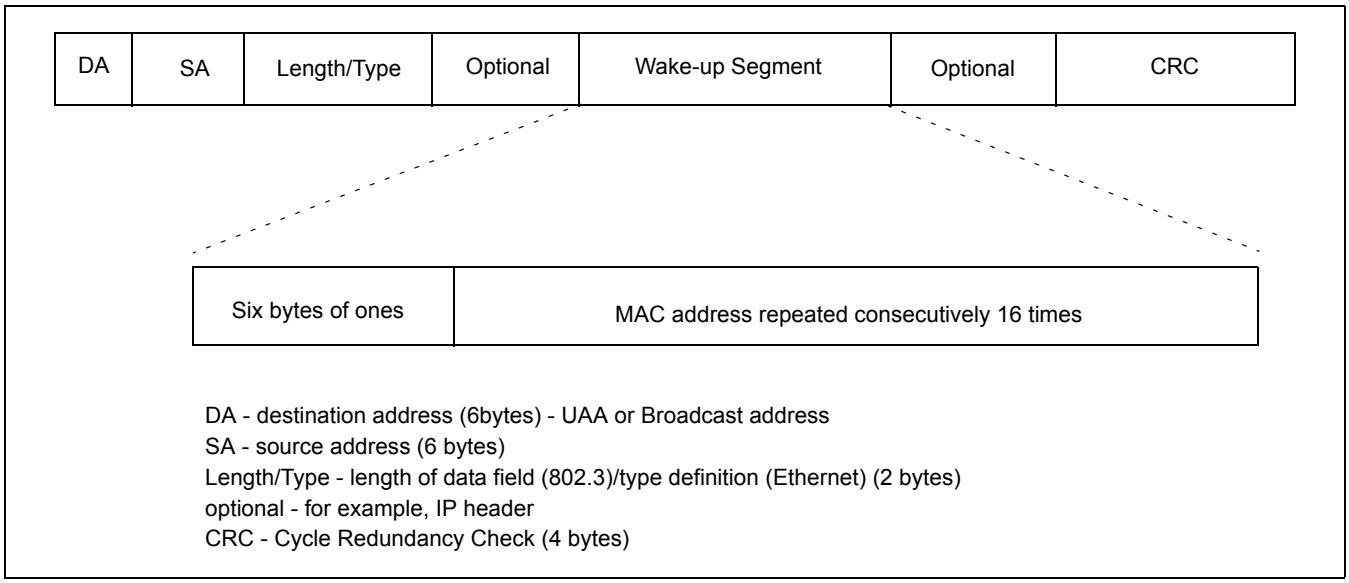
25.4.5 Wakeup On Lan (WOL) Support

WOL logic in EMAC supports WOL technology, an industry standard in the Wired for Management (WFM) Specification. WOL remotely awakens sleeping or powered-off nodes on a network. To wake up a node, a specific packet, called a *magic packet*, is sent to the network node. When so configured, EMAC monitors the incoming bit stream for magic packets.

The magic packet, also called a wake-up packet, contains a unique data field not normally expected in LAN traffic. When a WOL-enabled adapter on a powered-off client decodes this data field, a wake-up signal is generated. In the PPC440EP, with WOL mode enabled, EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1).

Figure 25-10 shows the wake-up packet format. The key to the wake-up packet is the MAC address of the target client, which is repeated 16 times. This pattern of 16 addresses in the data field is not expected to occur in any packet except the wake-up packet.

Figure 25-10. Wake-Up Packet Format



The destination address can be a specific address, called the universally administered address (UAA), or a broadcast address. If the destination address is a UAA, the wake-up packet is sent only to the client at that address. However, because the client is powered off and is no longer transmitting, some protocols remove the client MAC address from routing tables and internal caches at other nodes. In this case, wake-up packets addressed to a target client are discarded because nodes and routers do not know where to send them. The solution to this problem is to use a broadcast address. A directed broadcast has a valid network address and a broadcast host address. Network routers and nodes forward directed broadcasts to the appropriate network, where it is seen as a MAC-level broadcast and detected by the powered-off client.

25.4.5.1 EMAC WOL Support

EMAC enters WOL mode when EMACx_MR0[WKE] = 1.

WOL mode should only be changed while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0. After EMACx_MR0[WKE] = 1, EMACx_MR0[RXE] can be set to 1.

A reset (soft or hard) should be issued before programming EMAC to WOL mode. In WOL mode, EMAC does not propagate any received packets to MAL. Also, EMAC transmit channels do not request data from MAL.

25.5 Flow Control

For efficient system performance, each EMAC implements full-duplex flow control by handling specific MAC control packets contained in the pause opcode. EMAC supports flow control as defined in the IEEE 802.3x-1997 standard.

25.5.1 MAC Control Packet

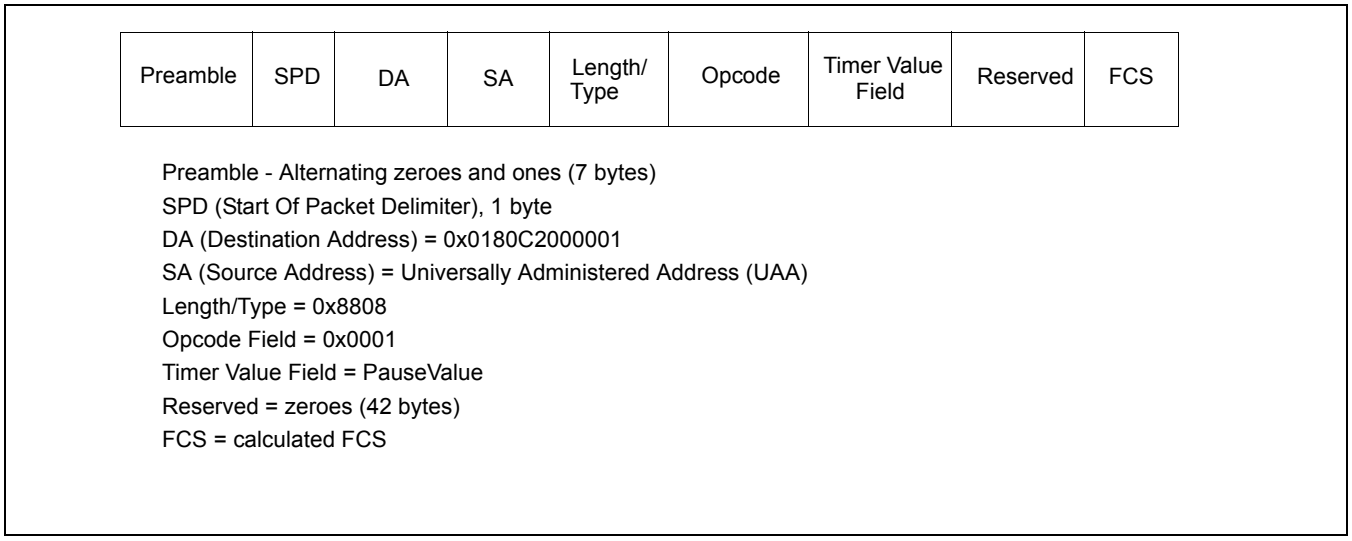
The flow control mechanism enables receive FIFO control logic to automatically notify the node transmitting packets to suspend transmission for a defined period of time. The pause control packet has a fixed length defined as follows:

MinFrameSize – 32 bits (60 bytes)

MAC control packets have a unique value of 0x8808 in the length/type field, and share the same packet format as normal Ethernet packets, except that the data field consists of an opcode field and a parameter field. The opcode field contains an opcode command and the parameter field contains a value associated with the opcode command (0x0001). The only opcode command defined by IEEE 802.3x is the pause opcode; the parameter field for the pause opcode defines the pause time. MAC control packets containing a pause opcode, also called pause packets, can have a destination address equal to a reserved multicast address, or can be the address of the receive station itself. The reserved multicast address is 0x0180C2000001.

Figure 25-11 illustrates the control packet format.

Figure 25-11. Control Packet Format



The timer value field contains the value of the delay interval in resolution of *pause_quanta*, defined in IEEE 802.3x as follows: "MAC Control Parameter[s] (pause_time) is a 2-octet, unsigned integer containing the length of time for which the receiving station is requested to inhibit data packet transmission. The field is transmitted most-significant octet first, and least-significant octet second. The pause_time is measured in units of pause_quanta, equal to 512 bit times. The range of possible pause_time is 0 to 65535 pause_quanta."

25.5.2 Control Packet Transmission

Two options initiate the pause packet transmission from EMAC. The transmitted pause packet forces the node, with the destination address specified, to temporarily suspend the transmission of packets to EMAC.

- Software initiated

  The packet transferred to EMAC by MAL for transmission is a pause packet created by software. EMAC transmits this as a normal packet.

- Automatic flow control initiated

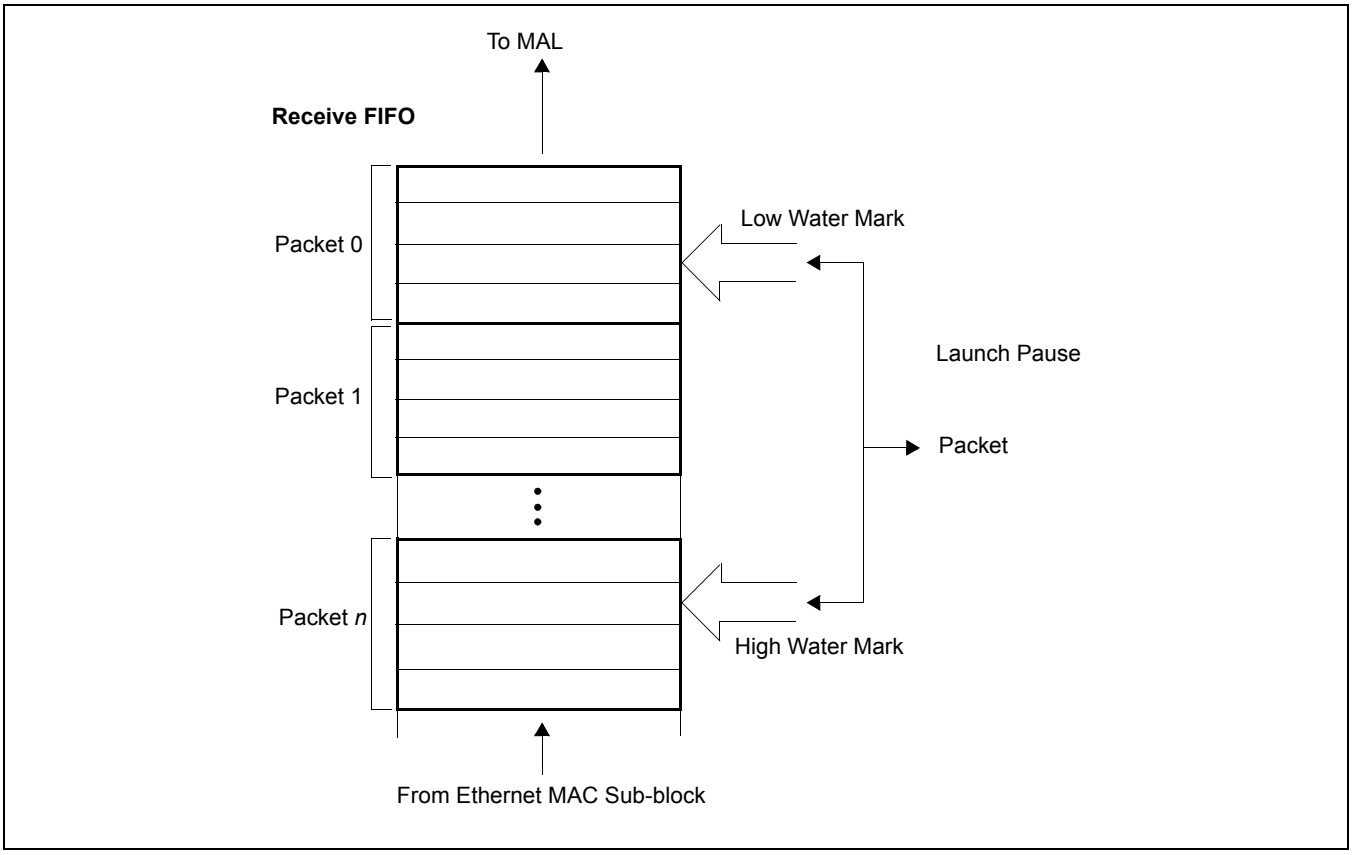
  The EMAC integrated flow control mechanism detects the need for and then transmits a control (pause) packet automatically. When building the control packet, EMAC obtains the SA (source address) field from

EMACx_IAHR and EMACx_IALR, and the timer value from the Pause Timer Register (EMACx_PTR[TVF]). The contents of the other fields in the packet are shown in *Figure 25-11*.

25.5.3 Integrated Flow Control

To enable integrated flow control in full-duplex mode, set EMACx_MR1[EIFC] = 1. When the receive FIFO reaches a predefined threshold level (called a high water mark and specified by EMACx_RWMR[RHWM]), an internal request for control (pause) packet transmission is activated. EMAC sends a control (pause) packet when a new packet enters the receive FIFO and the number of vacant entries in the Receive FIFO is less than the high water mark. When the receive FIFO reaches another predefined threshold level (the low water mark, specified by EMACx_RWMR[RLWM]), a new internal request for a pause packet transmission, with a pause timer value of 0, is activated. EMAC sends a pause packet, with a pause timer value of 0, only once, and only if a pause packet with a non-zero value in the pause timer was transmitted earlier.

Figure 25-12. Integrated Flow Control Mechanism

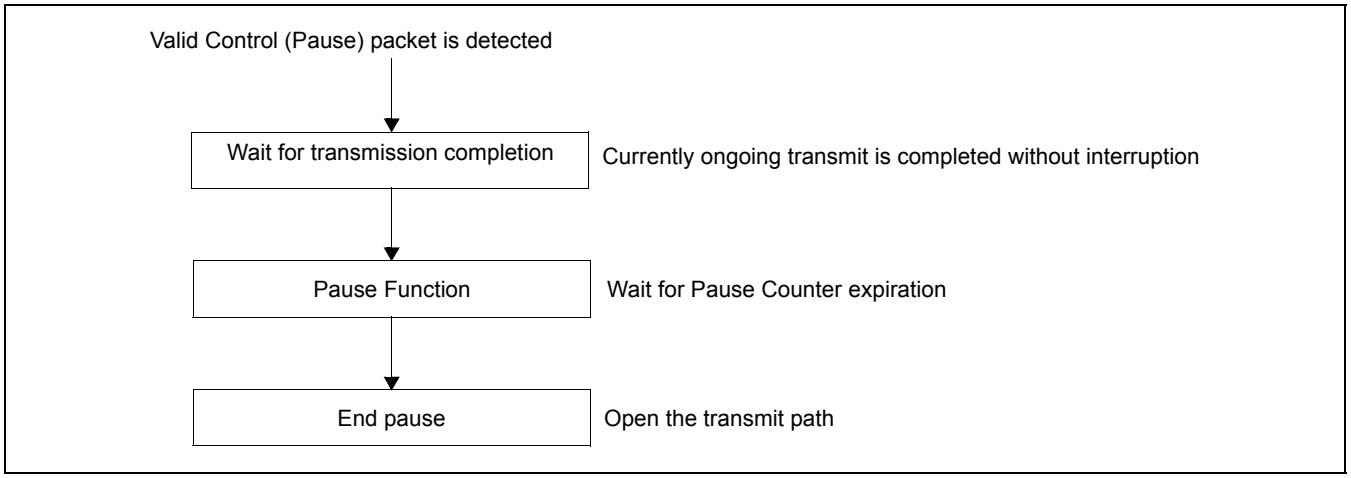


25.5.4 Control Packet Reception

In the receive path, the EMAC can be configured to respond to pause packets, or ignore them, as specified by EMACx_MR1[APP]. When response to pause packets is enabled and EMAC detects a valid MAC control packet with a pause opcode, the EMAC stores the value of the timer value field. The received packet is considered a valid control packet only if no error was detected during the packet reception. If, at the end of packet reception, the packet is considered valid, EMAC launches a pause operation state machine, as specified in the IEEE P802.3x standard. *Figure 25-13* illustrates the pause operation state machine.

If a control (pause) packet is received while another packet is transmitted, the ongoing transmission process is completed and the transmitter is paused. If other packets are in the transmit FIFO, their transmission is delayed until the pause timer expires. EMAC normally does not pass the MAC control packets to MAL unless EMACx_RMR[PPP] = 1.

Figure 25-13. Pause Operation State Machine



In the Pause Function state, EMAC decrements its internal pause timer, which was set to the timer value field of the received control packet.

Note 1: The transmission of control (pause) packets is not affected by the reception of a receive control (pause) packet. Received control (pause) packets inhibit only the transmission of regular packets from the Transmit FIFO.

Note 2: Receipt of a new valid control (pause) packet causes the pause timer of EMAC to be reloaded with the contents of the timer value field of the recently received packet, regardless of the current pause timer setting. This indicates new pause operations take precedence over earlier pause operations.

25.6 VLAN Support

EMAC can handle VLAN tagged packets, as specified in IEEE Draft P802.3ac/D1.0a standard when EMACx_MR1[VLE] = 1.

A VLAN tagged frame is an extension of the standard MAC packet. The extension for VLAN tag support consists of a 4-octet VLAN tag inserted between the end of the source address and the beginning of the length/type fields of the MAC packet.

The VLAN tag consists of two fields:

- 2-octet constant Type field value equal to the VLAN Tag Protocol Identifier (0x8100)
- 2-octet field containing Tag Control Information (TCI)

The MAC client data and FCS fields of the basic MAC packet follow the VLAN tag. The length of the packet is extended by four octets by the VLAN tag (up to 1522 bytes). The FCS is calculated over all fields from the destination address through the end of the MAC client data or pad (if present); that is, all fields except the preamble, SPD, and FCS.

Figure 25-14 illustrates the VLAN tagged frame format.

Figure 25-14. VLAN Tagged Packet Format

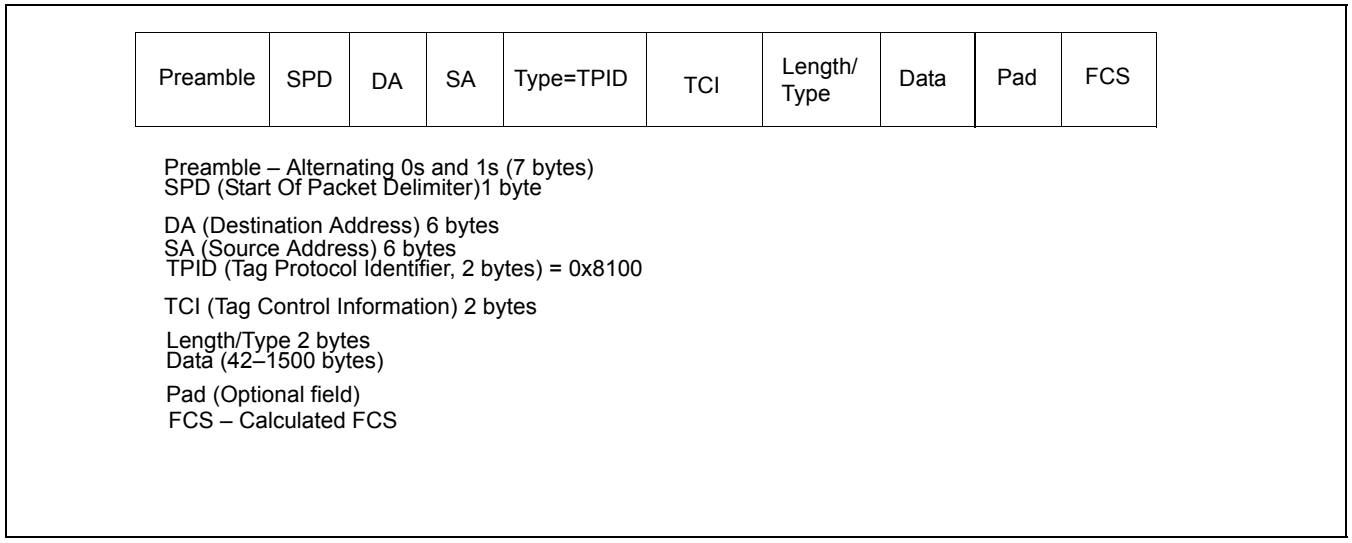
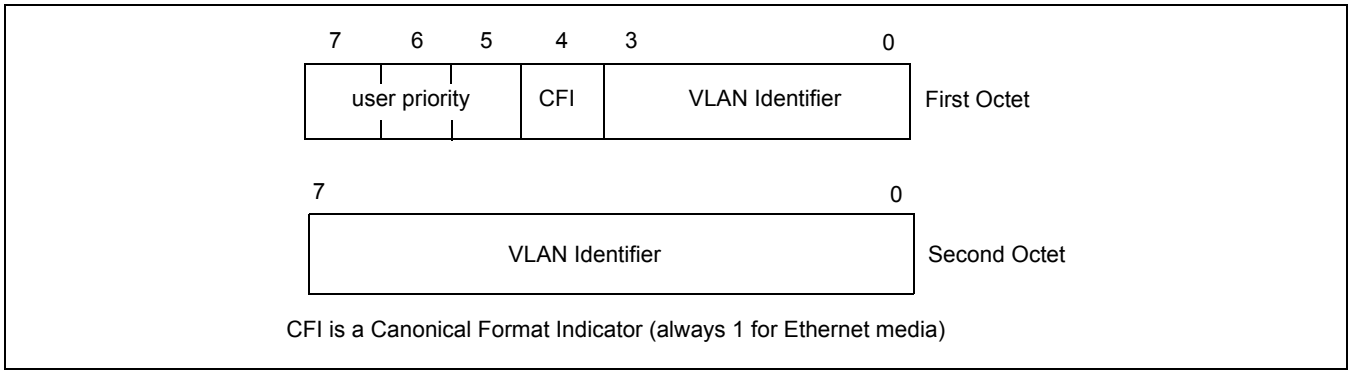


Figure 25-15 illustrates the structure of the TCI field.

Figure 25-15. Tag Control Information Field Structure



25.6.1 VLAN Tagged Packet Transmission

When EMACx_MR1[VLE] = 1, the following configuration options are available, depending on the content of the appropriate bits in the MAL control word (See *MAL TX Descriptor Control/Status Field* on page 594.)

- The Generate FCS bit (bit 6) is not set or both Insert VLAN Tag and Replace VLAN Tag bits (bits 10 and 11, respectively) are not set: EMAC transmits the packet without any changes
- Bit 6 is set and bit 10 is also set: EMAC will insert TPID and TCI for the transmitting packet using the content of EMACx_VTPID and EMACx_VTCI, respectively
- Bit 6 is set and bit 11 is also set: EMAC will replace TPID and TCI for the transmitting packet using the content of EMACx_VTPID and EMACx_VTCI, respectively

25.6.2 VLAN Tagged Packet Reception

If EMACx_MR1[VLE] = 1, the EMAC parses the VLAN Tag unique type/length in the incoming packet during the receive process. If the VLAN Tag is equal to the value stored in EMACx_VTPID, EMAC continues the receive process and allows the received packet to contain up to 1522 octets. Otherwise, the receive process is continued unless the length is greater than 1518 bytes.

25.6.3 Address Match Mechanism

The address match (or filtering) mechanism is a hardware aid that reduces the average amount of CPU cycles required to determine whether an incoming packet should be accepted.

EMAC uses various address filters for incoming packets by using the following address recognition modes.

- Individual mode (also referred to as physical)
- Multicast mode (also referred to as group)
- Broadcast mode (an all-ones group address)
- Promiscuous mode
- Promiscuous multicast mode
- WOL mode

A flowchart for address recognition of received packets is shown in *Figure 25-16* on page 608. If the least significant bit (LSb) of the first byte of the destination address (DA) is 0, the packet is considered individual. If the first bit received is 1, the packet is considered multicast. When the DA field contains all 1s, the packet is broadcast, a special type of multicast.

25.6.3.1 Non-WOL Mode

When EMAC operates in single individual mode (EMACx_RMR[IAE] = 1), the DA of the received packet is compared to the physical address stored in EMACx_IAHR and EMACx_IALR.

When EMAC operates in multiple individual mode (EMACx_RMR[MIAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) to determine whether or not to accept the packet.

When EMAC operates in promiscuous mode (EMACx_RMR[PME] = 1), all properly formed packets are received, regardless of the content of the DA field.

When EMAC operates in multicast promiscuous mode (EMACx_RMR[PMME] = 1), all multicast packets are received, regardless of the content of the DA field.

When EMAC operates in broadcast address mode (EMACx_RMR[BAE] = 1), EMAC performs an address compare on received packets with broadcast addresses.

When EMAC operates in multicast address mode (EMACx_RMR[MAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) as in multiple individual mode, in order to determine whether or not the packet should be accepted.

The logical address filter hardware implements a hash code searching technique commonly used by programmers. The hardware maps the DA of the incoming packet into one of 64 categories corresponding to 64 bits stored in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers. The hardware accepts or rejects the packet, depending on the state of the corresponding bit in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers corresponding to the selected category.

Figure 25-17 on page 609 shows the details of the hardware mapping algorithm. The example depicts multiple individual address mode, but with changes can be used for the multicast address mode.

If the most significant bit (MSb) of an incoming address is 0, the address is individual and is passed to the individual address filter. If the MSb of an incoming address is 1, the address is multicast and is passed to the multicast address filter. The individual/multicast address filter is a 64-bit mask composed of the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers (each register contains 16 bits of a 64-bit mask).

The incoming address is sent through the FCS circuit. After the 48 address bits have gone through the FCS circuit, the high-order 6 bits of the resulting FCS (32-bit CRC) are used to select a the 64-bit positions in the individual/multicast address filter. If the selected filter bit is 1, the address is accepted.

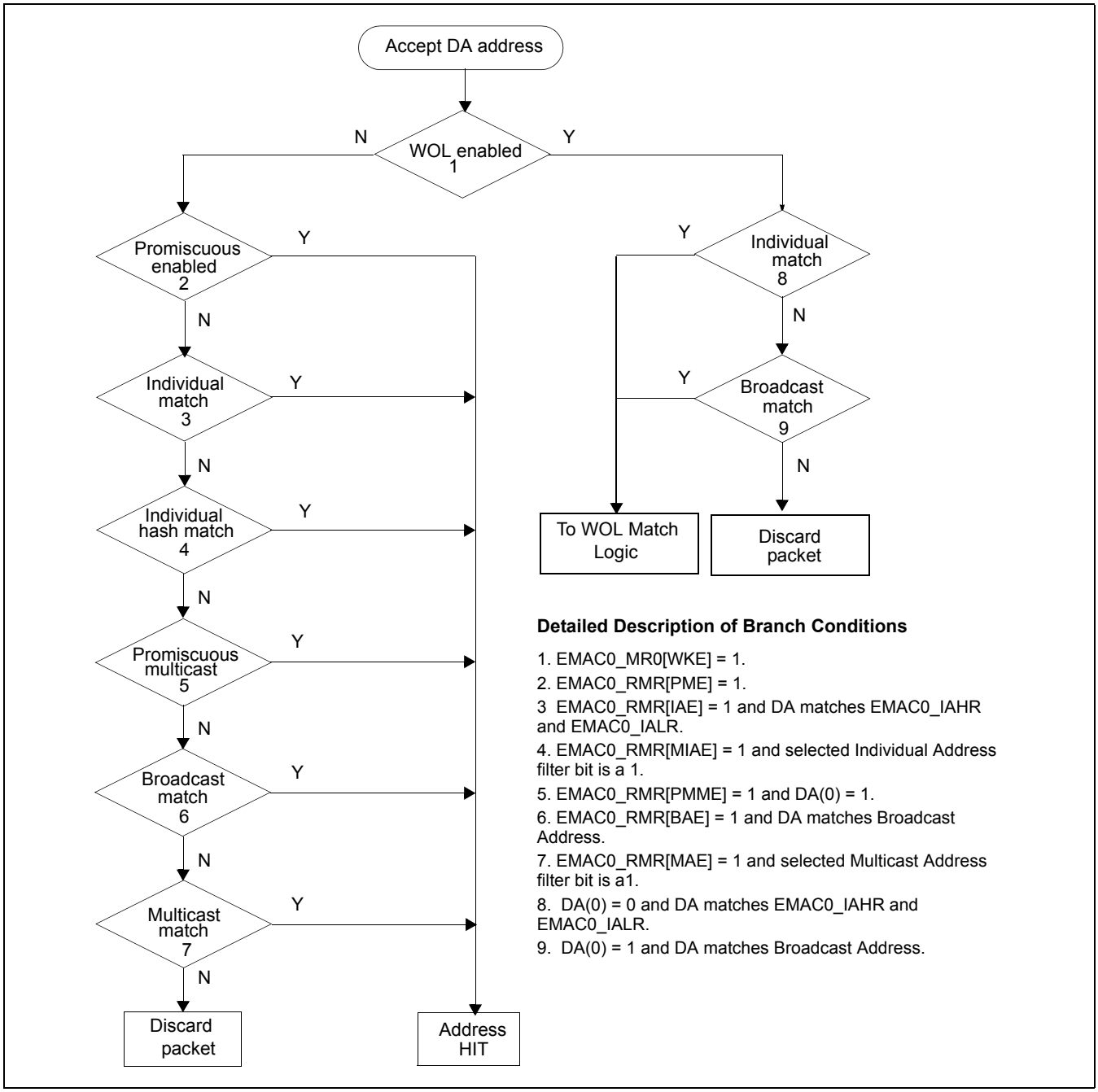
Note: The individual/multicast address filter ensures only that there is a possibility that the incoming packet belongs to this node. To determine if the packet belongs to the node, the incoming individual/multicast address propagated to the main memory is compared by software to the list of logical addresses to be accepted by this node.

For software, the task of mapping an individual/multicast address to one of 64 bit positions requires a program that uses the same CRC algorithm to calculate the hash.

25.6.3.2 WOL Mode

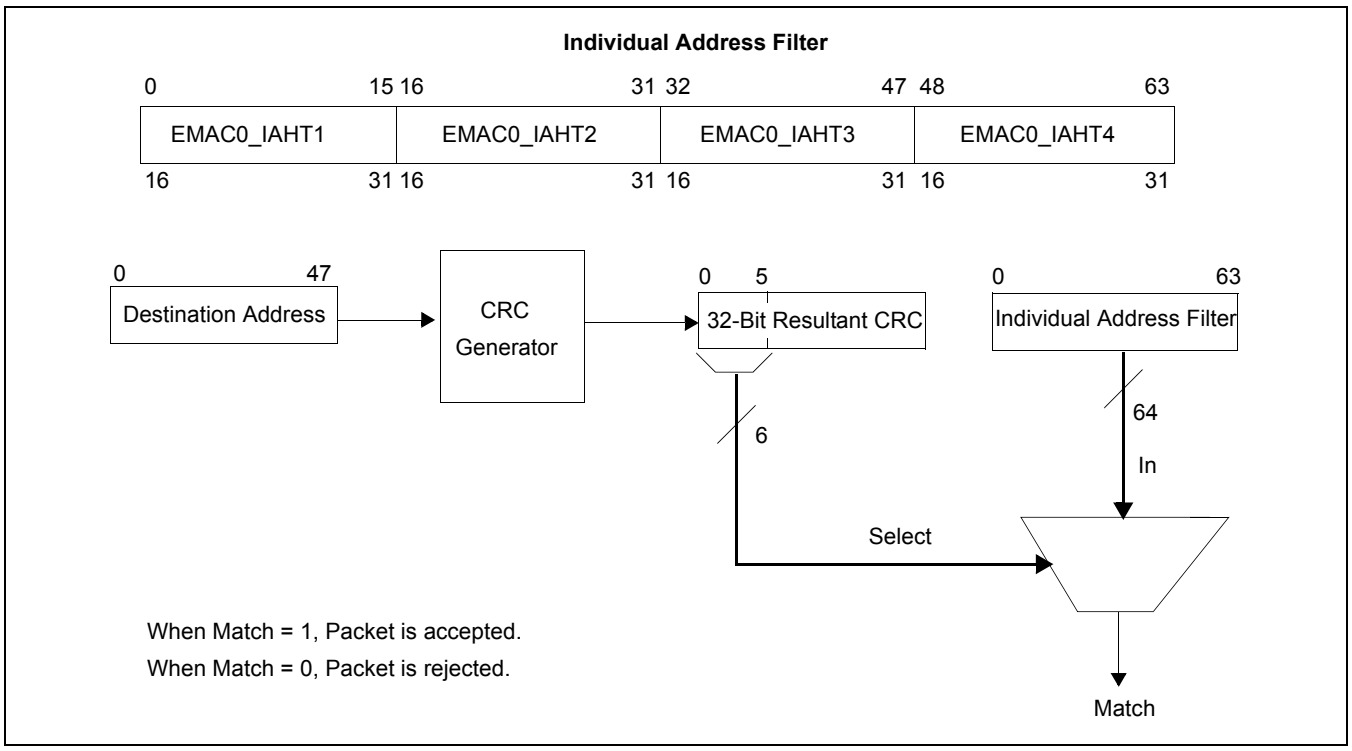
In WOL mode (EMACx_MR0[WKE] = 1), EMAC operates only with the broadcast or individual address in the destination address field.

Figure 25-16. Receive Address Recognition Flowchart



The example in *Figure 25-17* shows that the received individual address maps into category 24, and that bit 8 in EMACx_IAHT2 is set. The match indication is activated and the packet should be accepted.

Figure 25-17. Ethernet Address Filter Operation



25.7 EMAC Registers

This section describes the EMAC registers, which are listed in *Table 25-6* for EMAC0 and in *Table 25-7* for EMAC1. In the register descriptions, the registers are prefixed "EMACx" to denote the identical register implementations in each EMAC. The EMAC registers are accessed using the OPB slave interface. Access to these memory-mapped I/O (MMIO) registers should be word-aligned.

Note: It is required to perform a soft reset of EMAC before initialization as shown in EMACx Mode Register 0 (EMACx_MR0[SRST]).

Table 25-6. EMAC0 Register Summary

Register	Address	Description	Access	Page
EMAC0_MR0	0x0 EF60 0E00	EMAC 0 Mode Register 0	R/W	611
EMAC0_MR1	0x0 EF60 0E04	EMAC 0 Mode Register 1	R/W	612
EMAC0_TMR0	0x0 EF60 0E08	EMAC 0 Transmit Mode Register 0	R/W	613
EMAC0_TMR1	0x0 EF60 0E0C	EMAC 0 Transmit Mode Register 1	R/W	614
EMAC0_RMR	0x0 EF60 0E10	EMAC 0 Receive Mode Register	R/W	615
EMAC0_ISR	0x0 EF60 0E14	EMAC 0 Interrupt Status Register	R/W	616
EMAC0_ISER	0x0 EF60 0E18	EMAC 0 Interrupt Status Enable Register	R/W	618

Table 25-6. EMAC0 Register Summary (continued)

Register	Address	Description	Access	Page
EMAC0_IAHR	0x0 EF60 0E1C	EMAC 0 Individual Address High	R/W	619
EMAC0_IALR	0x0 EF60 0E20	EMAC 0 Individual Address Low	R/W	620
EMAC0_VTPID	0x0 EF60 0E24	EMAC 0 VLAN TPID Register	R/W	620
EMAC0_VTCI	0x0 EF60 0E28	EMAC 0 VLAN TCI Register	R/W	620
EMAC0_PTR	0x0 EF60 0E2C	EMAC 0 Pause Timer Register	R/W	621
EMAC0_IAHT1	0x0 EF60 0E30	EMAC 0 Individual Address Hash Table 1	R/W	621
EMAC0_IAHT2	0x0 EF60 0E34	EMAC 0 Individual Address Hash Table 2	R/W	621
EMAC0_IAHT3	0x0 EF60 0E38	EMAC 0 Individual Address Hash Table 3	R/W	621
EMAC0_IAHT4	0x0 EF60 0E3C	EMAC 0 Individual Address Hash Table 4	R/W	621
EMAC0_GAHT1	0x0 EF60 0E40	EMAC 0 Group Address Hash Table 1	R/W	621
EMAC0_GAHT2	0x0 EF60 0E44	EMAC 0 Group Address Hash Table 2	R/W	621
EMAC0_GAHT3	0x0 EF60 0E48	EMAC 0 Group Address Hash Table 3	R/W	621
EMAC0_GAHT4	0x0 EF60 0E4C	EMAC 0 Group Address Hash Table 4	R/W	621
EMAC0_LSAH	0x0 EF60 0E50	EMAC 0 Last Source Address Low	R	621
EMAC0_LSAL	0x0 EF60 0E54	EMAC 0 Last Source Address High	R	622
EMAC0_IPGVR	0x0 EF60 0E58	EMAC 0 Inter-Packet Gap Value Register	R/W	622
EMAC0_STACR	0x0 EF60 0E5C	EMAC 0 STA Control Register	R/W	622
EMAC0_TRTR	0x0 EF60 0E60	EMAC 0 Transmit Request Threshold Register	R/W	623
EMAC0_RWMR	0x0 EF60 0E64	EMAC 0 Receive Low/High Water Mark Register	R/W	624
EMAC0_OCTX	0x0 EF60 0E68	EMAC 0 Number of Octets Transmitted	R	625
EMAC0_OCRX	0x0 EF60 0E6C	EMAC 0 Number of Octets Received	R	625

Table 25-7. EMAC1 Register Summary

Register	Address	Description	Access	Page
EMAC1_MR0	0x0 EF60 0F00	EMAC 1 Mode Register 0	R/W	611
EMAC1_MR1	0x0 EF60 0F04	EMAC 1 Mode Register 1	R/W	612
EMAC1_TMR0	0x0 EF60 0F08	EMAC 1 Transmit Mode Register 0	R/W	613
EMAC1_TMR1	0x0 EF60 0F0C	EMAC 1 Transmit Mode Register 1	R/W	614
EMAC1_RMR	0x0 EF60 0F10	EMAC 1 Receive Mode Register	R/W	615
EMAC1_ISR	0x0 EF60 0F14	EMAC 1 Interrupt Status Register	R/W	616
EMAC1_ISER	0x0 EF60 0F18	EMAC 1 Interrupt Status Enable Register	R/W	618
EMAC1_IAHR	0x0 EF60 0F1C	EMAC 1 Individual Address High	R/W	619
EMAC1_IALR	0x0 EF60 0F20	EMAC 1 Individual Address Low	R/W	620
EMAC1_VTPID	0x0 EF60 0F24	EMAC 1 VLAN TPID Register	R/W	620
EMAC1_VTCI	0x0 EF60 0F28	EMAC 1 VLAN TCI Register	R/W	620

Table 25-7. EMAC1 Register Summary (continued)

Register	Address	Description	Access	Page
EMAC1_PTR	0x0 EF60 0F2C	EMAC 1 Pause Timer Register	R/W	621
EMAC1_IAHT1	0x0 EF60 0F30	EMAC 1 Individual Address Hash Table 1	R/W	621
EMAC1_IAHT2	0x0 EF60 0F34	EMAC 1 Individual Address Hash Table 2	R/W	621
EMAC1_IAHT3	0x0 EF60 0F38	EMAC 1 Individual Address Hash Table 3	R/W	621
EMAC1_IAHT4	0x0 EF60 0F3C	EMAC 1 Individual Address Hash Table 4	R/W	621
EMAC1_GAHT1	0x0 EF60 0F40	EMAC 1 Group Address Hash Table 1	R/W	621
EMAC1_GAHT2	0x0 EF60 0F44	EMAC 1 Group Address Hash Table 2	R/W	621
EMAC1_GAHT3	0x0 EF60 0F48	EMAC 1 Group Address Hash Table 3	R/W	621
EMAC1_GAHT4	0x0 EF60 0F4C	EMAC 1 Group Address Hash Table 4	R/W	621
EMAC1_LSAH	0x0 EF60 0F50	EMAC 1 Last Source Address Low	R	621
EMAC1_LSAL	0x0 EF60 0F54	EMAC 1 Last Source Address High	R	622
EMAC1_IPGVR	0x0 EF60 0F58	EMAC 1 Inter-Packet Gap Value Register	R/W	622
EMAC1_STACR	0x0 EF60 0F5C	EMAC 1 STA Control Register	R/W	622
EMAC1_TRTR	0x0 EF60 0F60	EMAC 1 Transmit Request Threshold Register	R/W	623
EMAC1_RWMR	0x0 EF60 0F64	EMAC 1 Receive Low/High Water Mark Register	R/W	624
EMAC1_OCTX	0x0 EF60 0F68	EMAC 1 Number of Octets Transmitted	R	625
EMAC1_OCRX	0x0 EF60 0F6C	EMAC 1 Number of Octets Received	R	625

25.7.1 Mode Register 0 (EMACx_MR0)

EMACx_MR0 defines the operating modes of the EMAC, which can be changed at any time during EMAC operation.

Figure 25-18. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.

3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

25.7.2 Mode Register 1 (EMACx_MR1)

EMACx_MR1 defines the EMAC operating modes, which can be changed only after a reset.

Software can use EMACx_MR1[FDE] and proper programming of the attached PHY to activate external loop-back mode.

When EMACx_MR1[FDE, ILE] = 1, EMAC wraps transmitted packets back to the receive FIFO without accessing the MII.

Figure 25-19. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MR1[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to *Flow Control* on page 601 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.
4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 Reserved 11 Reserved	Defines the possible operational frequency on the MII interface.

10:11	RFS	Receive (RX) FIFO Size 00 512 bytes 01 1 KB 10 2 KB 11 4 KB	
12:13	TFS	Transmit (TX) FIFO Size 00 Reserved 01 1 KB 10 2 KB 11 Reserved	
14		Reserved	Always 0
15:16	TR0	Transmit Request 0 00 Single packet 01 Multiple packets 10 Dependent mode (bits 17:18 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 0 of EMAC.
17:18	TR1	Transmit Request 1 00 Single packet 01 Multiple packets 10 Dependent mode (bits 15:16 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 1 of EMAC.
19	JE	Jumbo Enable	Must be set to 0.
20:31		Reserved	

25.7.3 Transmit Mode Register 0 (EMACx_TMR0)

EMACx_TMR0 defines EMAC operating modes during transmit operations (see *EMAC Transmit Operation* on page 593).

EMACx_TMR0[GNP0, GNP1, GNPD] are self-clearing. Writing 0 to these fields has no effect.

Figure 25-20. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP	Get New Packet 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel	EMACx_TMR0[GNP0] = 0 if EMAC is programmed in dependent mode.
1	GNP1	Get New Packet 1 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 1	EMACx_TMR0[GNP1] = 0 if EMAC is programmed in dependent mode.
2	GNPD	Get New Packet for Dependent Mode 0 Writing 0 to this bit has no effect 1 Packet ready for transmission in dependent mode	EMACx_TMR0[GNPD] = 0 if EMAC is not programmed in dependent mode. EMACx_TMR0[GNPD] = 1 activates the EMAC transmit path in dependent mode.
3	FC	First Channel 0 Activate TX Channel 0 first when GNPD is 1 1 Activate TX Channel 1 first when GNPD is 1	EMACx_TMR0[FC] is only meaningful in dependent mode, after resetting EMACx_ISR[DB DM]. EMACx_TMR0[FC] = 0 if EMAC is not programmed in dependent mode.
4:31		Reserved	

25.7.4 Transmit Mode Register 1 (EMACx_TMR1)

EMACx_TMR1 shown in *Figure 25-21* defines conditions for activation of MAL service requests during transmit operations (see *EMAC Transmit Operation* on page 593).

25.7.4.1 Low-Priority Requests

EMAC requests low priority service from the MAL when the number of vacant entries in the transmit FIFO exceeds the transmit low request (TLR) value.

To avoid a deadlock, the sum of EMACx_TMR1[TLR] and EMACx_TRTR[TRT] must be at least 4 smaller than the transmit FIFO size specified by the FIFO value in EMACx_MR1[TFS].

25.7.4.2 Urgent-Priority Requests

EMAC requests urgent priority service from MAL if both of the following conditions occur:

- EMAC begins transmitting the packet to the media before the entire packet is placed in the TX FIFO
- The number of vacant entries in the FIFO for the currently transmitting packet exceeds the TUR value

Software must coordinate the value of EMACx_TMR1[TUR] with the value of EMACx_MR1[TFS]. The value of EMACx_TMR1[TUR] must be smaller than the number of FIFO entries specified by EMACx_MR1[TFS].

The value of EMACx_TMR1[TUR] must be greater than that of EMACx_TMR1[TLR].

The EMACx_TMR1 contents can be changed only when EMACx_TMR0[GNP0, GNP1, GNPD] = 0.

Figure 25-21. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request	
5:7		Reserved	
8:15	TUR	Transmit Urgent Request	
16:31		Reserved	

25.7.5 Receive Mode Register (EMACx_RMR)

EMACx_RMR defines EMAC operating modes during receive operations.

Figure 25-22. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.	
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.	
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.	
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.	
4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.	
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.	
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.	
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.	
8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.	
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.	
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.	
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.	
12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.	

13:31		Reserved	

25.7.6 Interrupt Status Register (EMACx_ISR)

Each EMAC generates a distinct interrupt event indication. The event indication signal is driven out of the EMAC to UIC1 (interrupt 28 for EMAC0 and interrupt 30 for EMAC1). This interrupt is generated from the content of the EMACx_ISR. The content of the EMACx_ISR is first ANDed with the corresponding mask bits in the EMACx_ISER; the resulting bits are then logically ORed to produce the interrupt signal. Thus, if any of the resulting bits is a 1, an interrupt is generated.

Note: EMAC activates its interrupt signal only after an indication that status for the current packet was accepted by MAL (with the exception of "MMA Operation Succeed/MMA Operation Failed," which causes unconditional activation of interrupt, if it is not masked).

The interrupt indication is cleared by writing 1 to the related bit in the EMACx_ISR; writing 0 has no effect.

The event indication signal is cleared when all non-masked event indication bits are cleared.

Figure 25-23. Interrupt Status Register (EMACx_ISR)

0:5		Reserved	
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet	
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet	
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error	
9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than Short-EventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.
13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: • 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) • 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled)

14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:21		Reserved	
22	DBDM	Dead Bit Dependent Mode 0 No transmit error or SQE in dependent mode 1 Transmit error or SQE has occurred while in dependent mode	If EMACx_ISR[DBDM] = 1, EMAC does not request MAL service, even if EMACx_TMR0[GNPD] = 1. EMACx_ISR[DBDM] does not affect EMAC interrupt.
23	DB	Dead Bit 0 No transmit error or SQE for TX Channel 1 Transmit error or SQE has occurred for TX Channel	If EMACx_ISR[DB] = 1, EMAC does not request service for TX Channel from MAL, even if EMACx_TMR0[GNP] = 1. EMACx_ISR[DB] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 No SQEs on TX Channel 1 SQE test failure during transmission of a packet from TX Channel	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
25	TE0	Transmit Error 0 TX Channel transmission OK 1 TX Channel transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
26	DB1	Dead Bit 1 0 No transmit error or SQE for TX Channel 1 while not in dependent mode 1 Transmit error or a SQE has occurred for TX Channel 1 while not in dependent mode	If this bit is set, EMAC does not request MAL service for TX Channel 1 even if EMACx_TMR1[GNP1] = 1. EMACx_ISR[DB1] does not affect EMAC interrupt.
27	SE1	Signal Quality Error 1 0 No SQE on TX Channel 1 1 SQE test failure during transmission of a packet from TX Channel 1	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
28	TE1	Transmit Error 1 0 TX Channel 1 transmission OK 1 TX Channel 1 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer *not* valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

25.7.7 Interrupt Status Enable Register (EMACx_ISER)

EMACx_ISER indicates which conditions in the EMACx_ISR can generate an interrupt.

Each masking bit in the EMACx_ISER corresponds to a related bit in the EMACx_ISR. If a mask bit is set to 1, the corresponding status bit, when set, causes an interrupt to be generated. Setting a mask bit to 0 suppresses interrupt generation for the associated condition.

Mask bits for reserved bits in the EMACx_ISR are not implemented, have no effect on write, and return 0 on read.

Figure 25-24. Interrupt Status Enable Register (EMACx_ISER)

0:5		Reserved	
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.	
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.	
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.	
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.	
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.	
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.	
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.	
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.	
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.	

15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.	
16:21		Reserved	
22	DBDM	Dead Bit Dependent Mode 0 Dead bit dependent mode will not generate an interrupt 1 Dead bit dependent mode will generate an interrupt	
23	DB0	Dead Bit 0 0 Dead bit 0 will not generate an interrupt 1 Dead bit 0 will generate an interrupt	
24	SE0	SQE Error 0 SQE error on TX Channel will not generate an interrupt. 1 SQE error on TX Channel will generate an interrupt.	
25	TE0	Transmit Error 0 TX error on TX Channel will not generate an interrupt. 1 TX error on TX Channel will generate an interrupt.	
26	DB1	Dead Bit 1 0 Dead bit 1 will not generate an interrupt 1 Dead bit 1 will generate an interrupt	
27	SE1	SQE Error 1 0 SQE error on TX Channel 1 will not generate an interrupt. 1 SQE error on TX Channel 1 will generate an interrupt.	
28	TE1	Transmit Error 1 0 TX error on TX Channel 1 will not generate an interrupt. 1 TX error on TX Channel 1 will generate an interrupt.	
29		Reserved	
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.	
31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.	

25.7.8 Individual Address High (EMACx_IAHR)

EMACx_IAHR contains the high-order halfword of the station unique individual address.

During packet reception, if EMAC is programmed in individual address match mode (EMACx_RMR[IAE] = 1), the contents of EMACx_IAHR are concatenated with the content of EMACx_IALR to form a composite address that is compared with the destination address of the received packet. If addresses match, the packet is transferred to MAL.

During packet transmission, EMACx_IAHR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

Figure 25-25. Individual Address High Register (EMACx_IAHR)			
0:15		Reserved	
16:31	HO	Receive and Transmit High Order High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

25.7.9 Individual Address Low (EMACx_IALR)

EMACx_IALR contains the low-order word of the station unique individual address.

During packet reception, EMACx_IALR is compared with the corresponding address bits of the received packet.

During packet transmission, EMACx_IALR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

Figure 25-26. Individual Address Low Register (EMACx_IALR)			
0:15		Reserved	
16:31	LO	Receive and Transmit Low Order Low-order bits of Receive Individual Address or Transmit Source Address	

25.7.10 VLAN TPID Register (EMACx_VTPID)

EMACx_VTPID contains the value of the VLAN TPID (Tag Protocol Identifier) field.

During packet reception, packet bytes 13 and 14 are compared to the content of this register to check whether the packet is tagged with a VLAN ID.

During packet transmission, EMAC uses EMACx_VTPID when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

The value of this register must be a Type field (8100).

Figure 25-27. VLAN TPID Register (EMACx_VTPID)			
0:15		Reserved	
16:31	VIDT	VLAN ID tag	

25.7.11 VLAN TCI Register (EMACx_VTCI)

EMACx_VTCI contains the value of the VLAN TCI (Tag Control Information) field.

During packet transmission, EMAC uses EMACx_VTCI when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

Figure 25-28. VLAN TCI Register (EMACx_VTCI)			
0:15		Reserved	
16:31	VTCI	VLAN TCI tag	

25.7.12 Pause Timer Register (EMACx_PTR)

EMACx_PTR defines the time period for which the pause function is enabled. EMAC uses EMACx_PTR[TVR] as the timer value field of control (pause) packets (see *Control Packet Transmission* on page 602). Each bit corresponds to 512 bit times.

Figure 25-29. Pause Timer Register (EMACx_PTR)			
0:15		Reserved	
16:31	TVF	Timer Value Field	

25.7.13 Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

These registers are used in the hash table function of the multiple individual addressing mode.

See *Address Match Mechanism* on page 606 for more information. See *Figure 25-17* on page 609 for bit mapping information.

Figure 25-30. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)			
0:15		Reserved	
16:31	IAHN	Individual Address Hash Number	

25.7.14 Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

These registers are used in the hash table function of the group addressing mode.

See *Address Match Mechanism* on page 606 for more information. See *Figure 25-17* on page 609 for bit mapping information.

Figure 25-31. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)			
0:15		Reserved	
16:31	GAHN	Group Address Hash Number	

25.7.15 Last Source Address High (EMACx_LSAH)

EMACx_LSAH contains the high-order halfword of the source address of the last "good" received packet. The packet is considered to be "good" if EMAC is programmed to provide this packet to MAL.

Figure 25-32. Last Source Address High Register (EMACx_LSAH)			
0:15		Reserved	
16:31	LSA	Last Source Address High-Order Halfword	

25.7.16 Last Source Address Low (EMACx_LSAL)

EMACx_LSAL contains the low-order word of the source address of the last "good" received packet. The packet is considered "good" if EMAC is programmed to provide this packet to MAL.

Figure 25-33. Last Source Address Low Register (EMACx_LSAL)			
0:15		Reserved	
16:31	LSA	Last Source Address Low-Order Word	

25.7.17 Inter-Packet Gap Value Register (EMACx_IPGVR)

EMACx_IPGVR contains the value of one-third of the inter-packet gap (IPG) for the next packet to be transmitted. ("Frame" is synonymous with "packet.")

The resolution of each bit is 8-bit times. The minimum value in the register is four, causing a minimum.

Figure 25-34. Inter-Packet Gap Value Register (EMACx_IPGVR)			
0:25		Reserved	
26:31	IFG	Inter-Packet Gap	

25.7.18 STA Control Register (EMACx_STACR)

EMACx_STACR controls the MII Management interface. The software must follow the following steps during access to the EMACx_STACR:

1. Software polls EMACx_STACR[OC], waiting for it to be set by EMAC.

   EMAC sets EMACx_STACR[OC] = 0 when the EMACx_STACR is written to.

   EMAC then sets EMACx_STACR[OC] = 1 to indicate that the data has been written to the PHY, or the data read from the PHY is valid. The device driver should poll for EMACx_STACR[OC] = 1 before issuing a new command, or before using data read from the PHY.
2. The software can perform read/write access to the EMACx_STACR.
3. EMAC clears EMACx_STACR[OC] (sets EMACx_STACR[OC] = 0) and starts activity on the MII management interface.
4. Return to step 1.

Figure 25-35. STA Control Register (EMACx_STACR)			
0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21	OPBC	OPB Bus Clock Frequency 00 50 MHz 01 66 MHz 10 83 MHz 11 100 MHz	EMACx_STACR[OPBC] is used to generate the Management Data Clock (EMCMDCIk). When the operational frequency differs from those in the list, then the next greater frequency should be chosen.
22:26	PCDA	PHY Command Destination Address	This field contains the address of the PHY intended to receive the command
27:31	PRA	PHY Register Address	This field contains the PHY register address.

25.7.19 Transmit Request Threshold Register (EMACx_TRTR)

EMACx_TRTR defines the conditions that cause EMAC to initiate transmission to the Ethernet MAC sub-block, and for requesting service from the MAL.

EMACx_TRTR[TRT] defines the number of occupied entries in the transmit FIFO that should be filled before the transmit FIFO control logic initiates a transmit request to the Ethernet MAC sub-block.

If an entire packet is already located in the transmit FIFO, EMAC initiates a transmit regardless of the programmed value.

The software must coordinate the value of EMACx_TRTR[TRT] with the transmit FIFO size specified in EMACx_MR1[TFS].

To avoid deadlock, the sum of EMACx_ TMR1[TLR] and the number of FIFO entries represented by EMACx_TRTR[TRT] must be smaller, by at least 4, than the transmit FIFO size specified by the FIFO value in EMACx_MR1[TFS].

If this value is set too low, transmit FIFO underruns can occur.

In half-duplex mode, in case of collision, to allow packet re-transmission without involving MAL, EMAC preserves the necessary space in the Transmit FIFO unless it gets an indication that the collision window has elapsed.

The EMACx_TRTR can be written only while EMACx_MR0[TXI] = 1.

Figure 25-36. Transmit Request Threshold Register (EMACx_TRTR)			
0:4	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 00000 64 bytes 00001 128 bytes 00010 192 bytes 00011 256 bytes . . . 11111 2048 bytes	
5:31		Reserved	

25.7.20 Receive Low/High Water Mark Register (EMACx_RWMR)

The EMACx_RWMR defines the conditions that cause the EMAC to activate a low or urgent priority MAL request, and that manage flow control.

EMAC activates a low priority request if the number of occupied entries in the receive FIFO is greater than or equal to the content of EMACx_RWMR[RLWM] (the receive low water mark is reached). A request for a pause packet with a pause_value of 0 is also issued when the receive low water mark is reached.

Software must coordinate the value of EMACx_RWMR[RLWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RLWM] should be smaller than the receive FIFO size specified by the FIFO value in EMACx_MR1[RFS] and larger than the MAL burst length in FIFO entries.

If the entire packet is already in the receive FIFO, EMAC initiates a low priority request regardless of the programmed value.

EMAC activates an urgent priority request if the number of occupied entries in the Receive FIFO is greater than or equal to EMACx_RWMR[RHWM] (the receive high water mark is reached). A request for a pause packet is also issued when the receive high water mark is reached.

Software must coordinate the value of EMACx_RWMR[RHWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RHWM] should be greater than the value of EMACx_RWMR[RLWM] and less then the size of the receive FIFO represented in FIFO entries.

Figure 25-37. Receive Low/High Water Mark Register (EMACx_RWMR)			
0:8	RLWM	Receive Low Water Mark	
9:15		Reserved	
16:24	RHWM	Receive High Water Mark	
25:31		Reserved	

25.7.21 Transmitted Octects (EMACx_OCTX)

The read-only EMACx_OCTX register contains the number of transmitted octets.

Figure 25-38. Number of Octets Transmitted (EMACx_OCTX)			
0:31	OCTX	Number of octets (bytes) transmitted.	

25.7.22 Received Octects Register (EMACx_OCRX)

The read-only EMACx_OCRX register contains the number of received octets.

Figure 25-39. Number of Octets Received (EMACx_OCRX)			
0:31	OCRX	Number of octets (bytes) received.	

25.8 MII Interface

EMAC implements all MII functionality in accordance with Clause 22 in the IEEE Std. 802.3u.

The MII interface is a reconciliation sublayer interface which allows a variety of PHYs to be attached to the EMAC Ethernet MAC without future upgrade problems.

25.8.1 MII Station Management Interface

The EMAC MII station management unit (STA) implements a specific protocol and a special packet format to exchange management packets with the registers of the attached PHY. EMAC automatically generates MII management packets, which conform to Clause 22 in IEEE Std. 802.3u. EMAC uses the EMACx_STACR for generation of the management packet. illustrates the interface.

25.9 MAL—EMAC Packet Transfer Flow

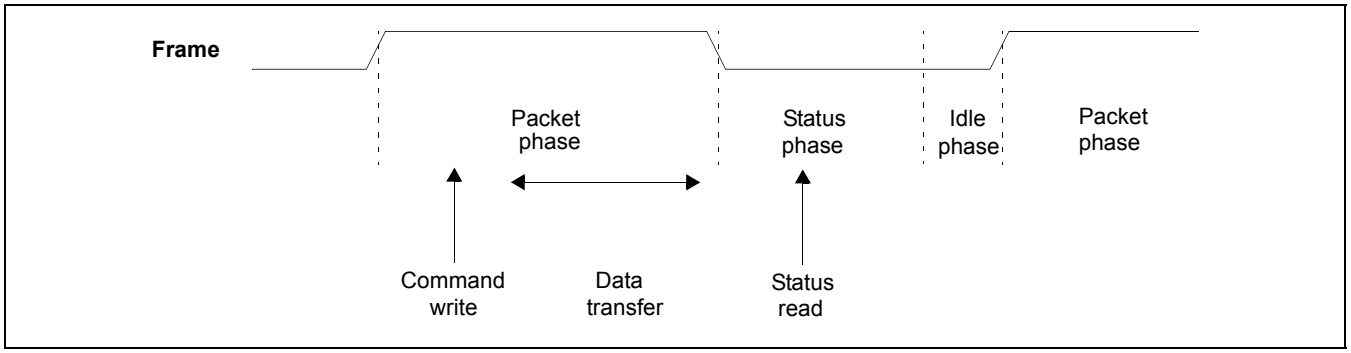
The packet transfer flow consists of three phases. These three phases are used to define the details of the EMAC-MAL protocol.

1. Packet phase - EMAC initiates a packet transfer operation. The packet transfer is started by a command write. During command write MAL provides control information for EMAC on a per-packet basis. Following the command write, MAL begins the data transfer, during which MAL transfers data between the buffers located in the system's memory and EMAC. In transmit, the data is transferred from the system's memory to EMAC, while in receive, the data is transferred from EMAC to the system's memory buffers.
   - EMAC remains in the packet phase until the data transfer has been completed or a ready status can be returned to MAL. The packet phase ends when EMAC deasserts the FRAME signal associated with the related channel (receive/transmit).
   - The packet phase is defined by activity of an appropriate FRAME signal.
2. Status phase - This is the second phase of the packet transfer. Following the de-assertion of the FRAME signal, EMAC switches to the status phase. At this stage, EMAC uses an appropriate signal as a request for service which is interpreted by MAL as a request for status read.

3. Idle phase - EMAC moves into the idle phase following a reset or after status was transferred (end of status phase). During the idle phase, EMAC cannot send any signals to MAL, nor can MAL send any active signals to EMAC. EMAC exits the idle phase by asserting the FRAME signal (and entering the packet phase described above). Idle phase can be skipped when EMAC operates in multiple transfer mode.

Figure 25-40 illustrates the different phases in the EMAC-MAL communication.

Figure 25-40. EMAC-MAL Communication Phases



During the packet and status phases EMAC signals a request for service by driving its arbitration level signal to a non-idle level.

25.10 Programming Notes

Certain combinations in device drivers are not allowed when writing to EMAC registers. When creating device drivers, ensure that the following guidelines are used:

- In dependent mode, EMACx_MR1[TR0] must be equal EMACx_MR1[TR1]
- When internal loopback is enabled (EMACx_MR1[ILE] = 1), EMAC must be configured in full-duplex mode (EMACx_MR1[FDE] =1)
- EMACx_MR1[IST] =0 only when EMACx_MR1[MF] = 0:10 and EMACx_MR1[FDE] =0
- In dependent mode, EMACx_ISER[SE0, TE0] must equal EMACx_ISER[SE1, TE1]
- EMACx_MR1[EIFC] =0 if EMACx_MR1[FDE] =0
- EMACx_TMR1[TLR] must be greater than the MAL burst size in entities (6 for MAL)
- EMACx_TMR1[TUR] must be greater or equal to EMACx_TMR1[TLR] and less than the Transmit FIFO size in entries (EMACx_MR1[TFS])
- To avoid deadlock, the sum of EMACx_TMR1[TLR] and the EMACx_TRTR[TRT] must be at least four less than the Transmit FIFO size specified in EMACx_MR1[TFS]
- EMACx_RWMR[RLWM] must be greater than the MAL burst size in entities (six for MAL)
- EMACx_RWMR[RHWM] must be greater than EMACx_RWMR[RLWM]
- EMACx_RWMR[RHWM] must be less than the Receive FIFO size in entities (EMACx_MR1[RFS])

25.10.1 Reset and Initialization

The EMAC must be initialized after a reset, or before performing configuration changes. The following types of reset operations can be applied to EMAC.

- Hard Reset. When RESET input is asserted, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. To be recognized, the

reset signal must be asserted for at least two cycles of the slowest clock domain inside EMAC (indicating that the hard reset must be at least 800 ns).

- Soft Reset. Software first should reset the appropriate MAL channels and then begin a soft reset by setting EMACx_MR0[SRST] = 1. In response to the soft reset, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. After EMAC finishes all activities related to the soft reset processing, EMACx_MR0[SRST] = 0.
- Smart Reset. The software initializes smart reset mode by writing 0 to EMACx_MR0[TXE] or EMACx_MR0[RXE], or to both. In this case, the Ethernet MAC sub-block completes on-going activity (receive, transmit, or both) and then goes to the related Idle state (indicated by setting either EMACx_MR0[TXI] = 1 or EMACx_MR0[RXI] = 1, or both). In this case, the control logic sub-block of EMAC is still accessible for OPB and MAL transactions.

Before performing the necessary configuration changes in EMAC, the software must follow one of the following scenarios. Then the EMAC can be properly configured.

25.10.1.1 Scenario 1

- Hard/soft reset was activated.
- During hard/soft reset, EMACx_MR0[TXE] and EMACx_MR0[RXE] are reset.
- Software detects that the EMACx_MR0[SRST] is reset (after soft reset only).
- Software keeps EMACx_TMR0[GNP0, GNP1] = 0.
- The software can change one or more fields in registers marked with a Reset write access mode in *Table 25-6 EMAC0 Register Summary* on page 609 (actually, all EMAC registers are accessible in this scenario).
- The software initializes EMACx_TMR0[GNP0, GNP1] as appropriate.
- The software configures EMACx_MR0[TXE, RXE].

25.10.1.2 Scenario 2

- Software sets EMACx_MR0[TXE] = 0.
- The TXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[TXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[TXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[TXE] = 0. The software can access only part of the EMAC registers marked with write access mode T in *Table 25-6 EMAC0 Register Summary* on page 609.
- After all configuration is done, software can set EMACx_MR0[TXE] = 1.

Note: When Scenario 2 occurs, EMAC can still receive packets if EMACx_MR0[RXE] = 1. Scenarios 2 and 3 can occur simultaneously.

25.10.1.3 Scenario 3

- Software sets EMACx_MR0[RXE] = 0.
- The RXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[RXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[RXI] = 1.

- Software performs the necessary EMAC configuration, keeping EMACx_MR0[RXE] = 0. The software can access only part of EMAC registers marked with write access mode R in *Table 25-6 EMAC0 Register Summary* on page 609.
- After all configuration is done, software can set EMACx_MR0[RXE] = 1.

Note: When Scenario 3 occurs, EMAC can still transmit packets if EMACx_MR0[TXE] = 1. Scenarios 2 and 3 can occur simultaneously.

26. Serial Port Operations

The PPC440EP contains four universal asynchronous receiver/transmitters (UARTs) to support communications with serial peripheral devices. Each UART includes a 64-byte send and a 64-byte receive FIFO.

Features of the UART include:

- 64-byte send FIFO, 64-byte receive FIFO
- Full duplex operation
- Programmable baud rate generator
- Supports 5- to 8-bit word size, 1 or 2 stop bits, even, odd, or no parity
- One 8-wire interface, two 4-wire interfaces, one 4-wire interface and two 2-wire interfaces, or four 2-wire interfaces
- Hardware flow control is selectable

The UART performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions, such as parity, overrun, framing, and break interrupt.

This UART powers up in character mode and can be put into FIFO mode to relieve the processor of excessive software overhead. Here, internal FIFOs are activated allowing 64 bytes (plus 3 bits per byte of error data in the RCVR FIFO) to be stored in both receive and transmit modes.

The source of the UART serial clock input is selected in SDR0_UART0:3[U0EC:U3EC] bits 8. Either the internal serial clock or an external serial clock can be selected. A programmable baud rate generator is included that is capable of dividing the UART serial clock input by a divisor of 1 to (2^{16} – 1) and producing the 16× clock required for driving the UART internal transmitter/receiver logic. The internal serial clock input is derived from the PLB clock by a divisor specified in SDR0_UART0, 1, 2 and 3[UDIV]. See *Clocking* on page 283 for additional information.

The UART has an interrupt system that can be programmed to the user's requirements, helping to minimize the computing required to handle the communications link. UART interrupts are capable of triggering an interrupt request to the PPC440EP interrupt controller.

26.1 Functional Description

- Equipped with complete status reporting capability
- Transmitter and receiver are each buffered with 64-byte FIFOs when FIFO mode selected
- Can add/delete standard asynchronous communication bits such as start, stop, and parity to/from the serial data
- When in character mode, holding and shift registers eliminate the need for precise synchronization between the processor and serial data
- Full prioritized interrupt system controls
- Independently controlled transmit, receive, line status, and data set interrupts
- Programmable baud rate generator divides the UART serial clock input by 1 to (2^{16}–1) and generates the 16x clock:

  Baud rate (bps) = (Serial Clock Input) / (16 x Decimal Divisor)

- Receiver uses 5-way oversampling as follows: it samples each serial bit five times, and if at least three of the samples are 1's, the bit is determined to be a 1, otherwise it is a 0
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no parity bit generation and detection
  - 1-,1.5-, or 2-stop bit generation
  - Variable baud rate
- Line break generation and detection, and false start bit detection
- Internal diagnostic capability:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation

26.2 Serial Port Clocking

Each of the four PPC440EP UARTs can be clocked individually from an external serial clock or from a single internally generated serial clock. The internally generated serial clock is derived from the PLB clock, and can only be an integer (n) fraction of that clock where n is in the range from 1 to 256.

The choice of serial clock frequency affects the serial communications error rate. If an external clock of 1.8432 MHz (or some multiple of this frequency) is used, the error rate approaches zero. However the internally-generated clock operates only at certain clock frequencies, all of which result in a small, non-zero error rate.

The optimum serial clock frequency is determined from the following relationship:

$$\text{Serial Clock} = \text{Baud Rate} \times 16 \times \text{UART Divisor}$$

Acceptable baud rates are always integral multiples of 300 (for example, 1200 = 4 × 300). *Table 26-1* shows optimum UART divisor and divide ratios for a range of possible baud rates. This information is provided for various clock frequencies. Note that the serial clock frequency must be less than half the OPB frequency. The UART divisor is programmed in UARTx_DLM and UARTx_DLL (see *Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)* on page 639). The value range is 1 to (2^16-1)=65535.

Table 26-1. Baud Rate Settings

Desired Baud Rate (bps)	PLB:UART Divide Ratio	Serial Clock Frequency (MHz)	Minimum OPB Frequency (MHZ)	UART Divisor	Actual Baud Rate (bps)	Error (%)
PLB Clock = 100 MHz						
1200	12	8.3333	16.9491	434	1200.08	0.006
2400	14	7.1429	14.4927	186	2400.15	0.01
4800	14	7.1429	14.4927	93	4800.31	0.01
9600	7	14.2860	29.4117	93	9600.61	0.01
19200	13	7.6923	15.6250	25	19230.77	0.16
28800	31	3.2258	6.4935	7	28801.84	0.01
33600	31	3.2258	6.4935	6	33602.15	0.01
38400	27	3.7037	7.4626	6	38580.25	0.47
57600	27	3.7037	7.4626	4	57870.37	0.47
115200	27	3.7037	7.4626	2	115740.74	0.47
307200	10	10.0000	20.4081	2	312500.00	1.73[1]
PLB Clock = 125 MHz						
1200	31	4.0322	8.1301	210	1200.08	0.006
2400	31	4.0322	8.1301	105	2400.15	0.01
4800	22	5.6818	11.4943	74	4798.83	-0.02
9600	22	5.6818	11.4943	37	9597.67	-0.02
19200	11	11.3636	23.2558	37	19195.33	-0.02
28800	16	7.3529	14.9254	16	28722.43	0.27
33600	29	4.3103	8.6956	8	33674.57	0.22
38400	29	4.3103	8.6956	7	38485.22	0.22
57600	17	7.3529	14.9254	8	57444.85	-0.27
115200	17	7.3529	14.9254	4	114889.70	-0.27
307200	25	5.0000	10.1010	1	312500	1.73[1]
PLB Clock =133 MHz						
1200	14	9.5238	19.4175	496	1200.07	0.006
2400	14	9.5238	19.4175	248	2400.15	0.01
4800	14	9.5238	19.4175	124	4800.31	0.01
9600	14	9.5238	19.4175	62	9600.61	0.01
19200	14	9.5238	19.4175	31	19201.23	0.01
28800	17	7.8431	15.0376	17	28835.06	0.12
33600	31	4.3011	8.4033	8	33602.15	0.01

Table 26-1. Baud Rate Settings (continued)

Desired Baud Rate (bps)	PLB:UART Divide Ratio	Serial Clock Frequency (MHz)	Minimum OPB Frequency (MHZ)	UART Divisor	Actual Baud Rate (bps)	Error (%)
38400	31	4.3011	8.4033	7	38402.46	0.01
57600	29	4.5977	9.2807	5	57471.26	-0.22
115200	24	5.5556	11.2359	3	115740.74	0.47
307200	27	4.9383	9.9751	1	308641.97	0.47

1. This amount of error may cause framing errors.

26.3 UART Registers

UART registers are accessed via memory to 0x0 EF60_0XYY where X=3 for UART0, X=4 for UART1, X=5 for UART2 and X=6 for UART3.

Table 26-2. UART Configuration Registers

Register	Address	Access	Description	Page
UARTx_RBR	0x0 EF60_0X00 [1]	R	UART x Receiver Buffer Register	633
UARTx_THR	0x0 EF60_0X00 [1]	W	UART x Transmitter Holding Register	633
UARTx_IER	0x0 EF60_0X01 [1]	R/W	UART x Interrupt Enable Register	633
UARTx_IIR	0x0 EF60_0X02	R	UART x Interrupt Identification Register	634
UARTx_FCR	0x0 EF60_0X02	W	UART x FIFO Control Register	635
UARTx_LCR	0x0 EF60_0X03	R/W	UART x Line Control Register	635
UARTx_MCR	0x0 EF60_0X04	R/W	UART x Modem Control Register	636
UARTx_LSR	0x0 EF60_0X05	R/W	UART x Line Status Register	637
UARTx_MSR	0x0 EF60_0X06	R/W	UART x Modem Status Register	638
UARTx_SCR	0x0 EF60_0X07	R/W	UART x Scratch Register	639
UARTx_DLL	0x0 EF60_0X00 [1]	R/W	UART x Divisor Latch (LSB)	639
UARTx_DLM	0x0 EF60_0X01 [1]	R/W	UART x Divisor Latch (MSB)	639

1. UARTx_LCR[DLAB] controls the function accessed through registers 0x0 EF60_0X00 and 0x0 EF60_0X01. When UARTx_LCR[DLAB] is 0, access is enabled to the Receiver/Transmitter registers and the Interrupt Enable register. When UARTx_LCR[DLAB] is a 1, access is enabled to the Divisor Latch registers.

The system programmer can access any of the UART registers via the processor. These registers control all UART operations including transmission and reception of data. In 440EP there are four UARTs, designated 0 through 3. In the following sections, the registers are specified with a generic name where x represents 0, 1, 2 or 3. For example, the Line Control Register appears as a UARTx_LCR.

In the four wire interfaces, two of the four wires are TX and RX. The remaining two wires can be programmed as a combination of DTR and DSR, or CTS and RTS. The selection is made in SDR0_PFC1 bits U0ME and U1ME respectively. DCD and RI are not available on the 4-wire interface.

26.3.1 Receiver Buffer Registers (UARTx_RBR)

Figure 26-1. UART Receiver Buffer Registers (UARTx_RBR)

0:7		Data bit	

Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.

26.3.2 Transmitter Holding Registers (UARTx_THR)

Figure 26-2. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit	

Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.

26.3.3 Interrupt Enable Registers (UARTx_IER)

Five UART interrupts on four priority levels are enabled via the Interrupt Enable Register, UARTx_IER. Any of the five interrupts can be used to surface a UART interrupt to the PPC440EP interrupt controller. Each interrupt can be enabled by setting its appropriate bit. Resetting UARTx_IER[4:7] totally disables the UART interrupt system. Disabling an interrupt prevents it from being shown as active in the UARTx_IIR and prevents it from signaling a UART interrupt to the PPC440EP interrupt controller. See *Table 26-3 Interrupt Priority Level* on page 634.

Figure 26-3. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/ disable state of ERBFI.

Note: UART0_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.

26.3.4 Interrupt Identification Registers (UARTx_IIR)

The UART prioritizes interrupts into four levels which are recorded in the Interrupt Identification Register. The interrupt types in the order of their priority are as follows:

1. Receiver line status
2. Received data available and character timeout indication
3. Transmitter holding register empty
4. Modem status

Table 26-3. Interrupt Priority Level

IIR Bit 4	IIR Bit 5	IIR Bit 6	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	1	Receiver Line Status	Overrun, Parity or Framing Error, or Break Interrupt.	Read LSR.
0	1	0	2	Received Data Available	Receiver data available or trigger level reached.	Read RBR, or FIFO drops below trigger level.
1	1	0	2	Character Timeout Indication	No characters have been removed from or input to the receiver FIFO during the last four character times and it contains at least one character during this time.	Read RBR.
0	0	1	3	Transmitter Holding Register Empty	Transmitter Holding Register Empty.	Read IIR (if source of interrupt) or write THR.
0	0	0	4	Modem Status	Clear to Send, Data Set Ready, Ring Indicator or Data Carrier Detect.	Read MSR.

When the processor accesses UARTx_IIR, the UART records new interrupts, but does not change its current contents until the access by the processor is complete. The UART indicates the highest priority interrupt pending to the PPC440EP interrupt controller via the IIR.

Figure 26-4. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FE	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)	
2:3		Reserved	
4:6	INTID	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved	**Note:** Priority 1 is highest priority.

7	INTP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending	When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.5 FIFO Control Registers (UARTx_FCR)

The FIFO control register has the same address as the IIR and is a write-only register. This register is used to perform FIFO control operations such as selecting the type of DMA signaling, setting the receiver FIFO trigger levels, clearing the FIFOs, and enabling the FIFO.

Figure 26-5. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.6 Line Control Registers (UARTx_LCR)

The system programmer uses the line control register (LCR) to specify the format of the asynchronous data communications exchange and to set the Divisor Latch Access bit. The contents of the LCR can also be read by the processor. The read capability simplifies system programming, and eliminates the need for separate storage of the line characteristics in system memory.

Figure 26-6. UART Line Control Registers (UARTx_LCR)			
0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR [EPS] = 0 and UARTx_LCR[PEN] = 1,the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
Note: UART0_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.7 Modem Control Registers (UARTx_MCR)

The interface between the modem, data set, or peripheral device emulating a modem, and the UART, is controlled by the Modem Control Register (UARTx_MCR).

Figure 26-7. UART Modem Control Registers (UARTx_MCR)			
0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	

3	LOOP	Loopback Mode 0 Disabled 1 Enabled	Provides a local loop back feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{DSR}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DCD}$ are disconnected. 4. The four modem control outputs $\overline{DTR}$, $\overline{RTS}$, $\overline{OUT1}$, and $\overline{OUT2}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.
4	OUT2	User Output 2 0 $\overline{OUT2}$ inactive (1) 1 $\overline{OUT2}$ active (0)	The $\overline{OUT2}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{OUT1}$ inactive (1) 1 $\overline{OUT1}$ active (0)	The $\overline{OUT1}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 $\overline{RTS}$ inactive (1) 1 $\overline{RTS}$ active (0)	
7	DTR	Data Terminal Ready 0 $\overline{DTR}$ inactive (1) 1 $\overline{DTR}$ active (0)	
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.8 Line Status Registers (UARTx_LSR)

Information concerning the data transfer is held for the processor in this register. Bits 3 through 6 are conditions that produce a receiver line status interrupt whenever the condition corresponding to the active bit is detected and the interrupt is enabled. This register is intended for read operations only and writing is not recommended.

Figure 26-8. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character. 1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.	

2	THRE	Transmitter Holding Register Empty Indicator 0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO. 1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the PPC440EP interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.
3	BI	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	Framing Error Indicator. 0 Reset to 0 whenever processor reads LSR. 1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level). Indicates that a valid stop bit was not found in the received character.	Error causes a Receiver Line Status Interrupt.
5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR.[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.
6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.9 Modem Status Registers (UARTx_MSR)

The processor can monitor the present state of the modem (or peripheral device) control lines by reading the Modem Status Register (UARTx_MSR). In addition, the UARTx_MSR has four bits to indicate if any of the modem (or peripheral device) control lines have changed state.

Figure 26-9. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].

2	DSR	Data Set Ready	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{DCD}$ input changed state	Indicates that the $\overline{DCD}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{RI}$ input changed from 0 to 1	Indicates that the $\overline{RI}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 $\overline{DSR}$ input changed state	Indicates that the $\overline{DSR}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{CTS}$ input changed state	Indicates that the $\overline{CTS}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.10 Scratchpad Registers (UARTx_SCR)

A scratchpad register intended for use by the programmer as a temporary data location is provided in this UART. It does not control the UART operation in any way.

Figure 26-10. UART Scratchpad Registers (UARTx_SCR)

0:7		Data bits	
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

26.3.11 Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)

The divisor latches are used to program the UART divisor used in generating the baud clock. A 16-bit divisor may be programmed through these registers. Access to these registers is provided by setting UARTx_LCR[DLAB] = 1. These registers have a power-on reset value of 0.

Figure 26-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits	
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

Figure 26-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits	
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

The UART divisor is calculated using the following formula:

UART Divisor = Serial Input Clock/(16 x Baud Rate)

For example, if the serial input clock= 11.0592MHz and a baud rate of 9600 bps is required:

UART Divisor = Serial Input Clock/(16 x Baud Rate)

= 11,059,200/(16x9600)

= 72 = 0x48

For this example, UARTx_DLM should be programmed to 0 and UARTx_DLL register should be programmed to 0x48. Due to the error introduced by rounding, some baud rates cannot be generated at certain serial input clock frequencies. *Table 26-4* lists some common baud rates and their corresponding Divisor Latch register values with a serial input clock of 11.0592MHz.

Table 26-4. Divisor Latch Settings for Certain Baud Rates

Baud Rate (bps)	Divisor Latch MSB	Divisor Latch LSB
9600	0x00	0x48
19200	0x00	0x24
28800	0x00	0x18
38400	0x00	0x12
57600	0x00	0x0C

26.3.12 UART Configuration Register 0 (SDR0_UART0))

Figure 26-13 describes SDR0_UART0 register bits.

Reset value = 0x20000026

Figure 26-13. UART Configuration Register 0 (SDR0_UART0)

0:3	U0ICS	UART 0 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART 0 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock	
9	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled	
10	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled	

11	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U0DIV	UART 0 Divider 0000_0000 - divider = 256 0000_0001 - divider = 1 0000_0010 - divider = 2 ..... 1111_1111 - divider = 255	The source for SDR0_UART0[U0DIV] is indicated by SDR0_UART0[U0ICS]

26.3.13 UART Configuration Register 1 (SDR0_UART1))

Figure 26-14 describes SDR0_UART1 register bits.

Reset value = 0x20000026

Figure 26-14. UART Configuration Register 1 (SDR0_UART1)

0:3	U1ICS	UART 1 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART1 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock	
9	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled	
10	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled	
11	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U1DIV	UART 1 Divider 0000_0000 - UART1 divisor = 256 0000_0001 - UART1 divisor = 1 0000_0010 - UART1 divisor = 2 ..... 1111_1111 - UART1 divisor = 255	The source for SDR0_UART1[U1DIV] is indicated by SDR0_UART1[U1ICS]

26.3.14 UART Configuration Register 2 (SDR0_UART2))

Figure 26-15 describes SDR0_UART2 register bits.

Reset value = 0x20000026

Figure 26-15. UART Configuration Register 2 (SDR0_UART2)

0:3	U2ICS	UART 2 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART2 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U2EC	UART 2 EXT Clock Enable 0 UART2 uses the internal serial clock 1 UART2 uses the external serial clock	
9	U2DTE	UART 2 DMA Transmit Enable 0 UART2 DMA transmit is disabled 1 UART2 DMA transmit is enabled	
10	U2DRE	UART 2 DMA Receive Enable 0 UART2 DMA receive is disabled 1 UART2 DMA receive is enabled	
11	U2DC	UART 2 DMA Clear 0 U2DTE and U2DRE are not cleared when UART receives a corresponding terminal count. 1 U2DTE and U2DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U2DIV	UART 2 Divider 0000_0000 - UART2 divisor = 256 0000_0001 - UART2 divisor = 1 0000_0010 - UART2 divisor = 2 ..... 1111_1111 - UART2 divisor = 255	The source for SDR0_UART2[U2DIV] is indicated by SDR0_UART2[U2ICS]

26.3.15 UART Configuration Register 3 (SDR0_UART3))

Figure 26-16 describes SDR0_UART3 register bits.

Reset value = 0x20000026

Figure 26-16. UART Configuration Register 3 (SDR0_UART3)

0:3	U3ICS	UART 3 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART3 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U3EC	UART 3 EXT Clock Enable 0 UART3 uses the internal serial clock 1 UART3 uses the external serial clock	

9	U3DTE	UART 3 DMA Transmit Enable 0 UART3 DMA transmit is disabled 1 UART3 DMA transmit is enabled	
10	U3DRE	UART 3 DMA Receive Enable 0 UART3 DMA receive is disabled 1 UART3 DMA receive is enabled	
11	U3DC	UART 3 DMA Clear 0 U3DTE and U3DRE are not cleared when UART receives a corresponding terminal count. 1 U3DTE and U3DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U3DIV	UART 3 Divider 0000_0000 - UART3 divisor = 256 0000_0001 - UART3 divisor = 1 0000_0010 - UART3 divisor = 2 ..... 1111_1111 - UART3 divisor = 255	The source for SDR0_UART3[U3DIV] is indicated by SDR0_UART3[U3ICS]

26.4 FIFO Operation

This section discusses the various modes of operation including interrupt mode, polled mode, and sleep mode.

26.4.1 Interrupt Mode

The following sections describe the operation of the UART in interrupt mode.

26.4.1.1 Receiver

Receiver interrupts occur as described below when the receiver FIFO and receiver interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ERBFI] = 1.

The received data available interrupt is issued when the number of characters in the FIFO has reached the trigger level programmed into UARTx_FCR. This interrupt is reset to 0 when the FIFO character count drops below this trigger level.

The received data available indicator is issued when the number of characters in the FIFO has reached the trigger level programmed into UARTx_FCR. This indicator is reset to 0 when the FIFO character count drops below this trigger level.

The receiver line status interrupt (UARTx_IIR = 0xC6) is a top priority interrupt, whereas the received data available interrupt (UARTx_IIR = 0xC4) is a second priority interrupt.

Data Ready (UARTx_LSR[DR]) is set as soon as a character is transferred from the shift register to the receiver FIFO. This bit is reset when the FIFO is empty.

Receiver timeout interrupts will occur as described below when the receiver FIFO and receiver interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ERBFI] = 1.

A FIFO timeout will occur when:

At least one character is in the receiver FIFO, no serial characters have been received for four serial character time periods, and the processor has not read the FIFO for four serial character time periods. A serial character time period is as follows:

1/(baud rate) x (# start bits + word length + # parity bits + # stop bits)

For example, the serial character time period for an 8-bit word with one parity bit, two stop bits at 56K baud is as follows:

1/(56000) x (1 + 8 + 1 + 2) = 214.3μs

So the timeout would occur after 857.1μs, if the above conditions hold.

When a timeout interrupt has occurred, it is cleared and the timer is reset when the processor reads one character from the receiver FIFO.

When a timeout interrupt has not occurred, the timer is reset after a new serial character is received or the processor reads the receiver FIFO.

26.4.1.2 Transmitter

Transmitter interrupts occur, as described below, when the transmitter FIFO and transmitter interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ETBEI] = 1.

The transmitter holding register interrupt (UARTx_IIR = 0xC2) occurs when transmit FIFO is empty, and is cleared as soon as the transmitter holding register is written to or the IIR is read. One to 16 characters may be written to the transmitter FIFO while servicing this interrupt.

The transmitter FIFO empty indications are delayed by one character time minus the last stop bit time whenever the following event occurs: UARTx_LSR[THRE] = 1 and there were less than two bytes simultaneously present in the transmit FIFO since the last UARTx_LSR[THRE] = 1. If UARTx_FCR[FE] = 1 (FIFOs enabled), the first transmitter interrupt after changing UARTx_FCR[FE] is immediate.

Receiver FIFO trigger level interrupts, received data available interrupts, and character timeouts all have equivalent second interrupt priority. Current transmitter holding register empty interrupt and Transmit FIFO empty have equivalent third interrupt priority.

26.4.2 Polled Mode

When UARTx_FCR[FE] = 1 (FIFOs enabled), and UARTx_IER[5:7] are all set to 0 (interrupts disabled), the UART is in FIFO polled mode of operation. The receiver and transmitter are controlled separately, so either can be in polled mode of operation. In polled mode, the user program must check the UARTx_LSR to see the status of the receiver and/or transmitter.

UARTx_LSR[3:6] specifies which errors (if any) have occurred. Character status errors are handled in the same way as in interrupt mode. Since UARTx_IER[ELSI] = 0, the IIR is not affected. UARTx_LSR[DR] is set as long as there is at least one character in the receiver FIFO. UARTx_LSR[THRE] indicates if the transmitter FIFO is empty. UARTx_LSR[TEMT] indicates if the transmitter FIFO and the transmitter shift register are empty. UARTx_LSR[RFE] indicates if there are any errors in the receiver FIFO.

In FIFO polled mode, there are no character timeout or trigger levels; however, the FIFOs are still capable of holding characters.

26.4.3 UART and Sleep Mode

All UARTs can be placed in sleep mode via the UART sleep bits in the CPM0_ER register (CPM0_ER[UART0:UART1:UART2_UART3]). The most common usage would be to save a little power if one or any of the UARTs were not going to be used.

Using sleep mode dynamically requires careful software control to make sure the UARTs are idle before putting them to sleep.

26.5 DMA Operation

PPC440EP implements two DMAs: DMA to PLB4 (DMA2P40) and DMA to PLB3 (DMA2P30) (see *Direct Memory Access Controllers* on page 509). The DMA controller can be configured to perform DMA operations using UART, which appears as an 8-bit peripheral to the DMA controller. When selected, the UART receiver is internally wired to the DMAReq and DMAAck signals of DMA channel 2, and the transmitter is internally wired to the DMAReq and DMAAck signals of DMA channel 3.

The UART can be operated in FIFO mode or non-FIFO mode. In FIFO mode, the transfers can be done as single transfers (DMA mode 0) or multiple transfers (DMA mode 1), depending on the setting of the DMS field in the FIFO Control Register (FCR). In non-FIFO mode, DMA transfers are performed using single transfers, using the UART's DMA mode 0. This section describes proper UART DMA programming. For more information on general DMA programming, see *Direct Memory Access Controllers* on page 509.

DMA2P40 is attached to the 128-bit processor local bus.

DMA2P30 is attached to the 64-bit processor local bus. *Table 26-5* lists the DMA to PLB3 channel assignments in The PPC440EP.

Table 26-5. DMA to PLB3 Channel Assignments

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 0	Sub-Channel 0	UART 0 Receive	External Peripheral DMA 0
	Sub-Channel 1	UART 2 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

Table 26-5. DMA to PLB3 Channel Assignments (continued)

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 1	Sub-Channel 0	UART 0 Transmit	External Peripheral DMA 1
	Sub-Channel 1	UART 2 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 2	Sub-Channel 0	UART 1 Receive	External Peripheral DMA 2
	Sub-Channel 1	UART 3 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 3	Sub-Channel 0	UART 1 Transmit	External Peripheral DMA 3
	Sub-Channel 1	UART 3 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

26.5.1 Transmitter DMA Mode

The four DMA channels on the PPC440EP can be used to move data to and from the four UARTs. The UART0 and UART2 transmit channel is tied to DMA channel 1, the UART1 and UART3 transmit channel is tied to DMA channel 3. Use of the DMA with the UARTs is controlled by the UxDTE bits of the SDR0_UART0:3 registers.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when the FIFOs are disabled or the FIFOs are enabled and there are no characters in the TX FIFO or Transmit Holding Register (THR). Once activated, the DMA request goes inactive after the first character is loaded into the TX FIFO or THR.

When UARTx_FCR[DMS] is 1 the DMA request goes active, when FIFOs are enabled and there is at least one unfilled position in the TX FIFO. DMA request goes inactive when the TX FIFO is completely full. To operate in this mode the corresponding PPC440EP DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA2P30_CRx to 1 configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 26-6 lists required register settings for UART0 transmit transfers. The UART0 transmit channel is tied to DMA channel 1. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-6. UART 0 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART0[U0DTE] = 1	UART0 DMA transmit channel is enabled using DMA channel 1
SDR0_UART0[U0DC]	Set to 0 to not clear SDR0_UART0[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR1[TD] = 0	DMA channel 1 transfer direction is from memory to peripheral.
DMA2P30_CR1[PL] = 1	DMA channel 1 peripheral is on the OPB (UART0).
DMA2P30_CR1[PW] = 000	Peripheral width is byte (8 bits).
DMA2P30_CR1[TM] = 00	DMA Channel 1 is in peripheral mode.
DMA2P30_CR1[PWC] = 000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR1[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR1[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC1[SCID] = 000	DMA Channel 1 sub-channel 0 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 1 for UART0 transmitter transfers, external DMA transfers cannot be performed on this channel.	

Table 26-7 lists required register settings for UART 1 transmit transfers. The UART1 transmit channel is tied to DMA channel 3. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-7. UART 1 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART1[U1DTE] = 1	UART 1 DMA transmit channel is enabled using DMA channel 3.
SDR0_UART1[U1DC]	Set to 0 to not clear SDR0_UART1[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR3[TD] = 0	DMA channel 3 transfer direction is from memory to peripheral.
DMA2P30_CR3[PL] = 1	DMA channel 3 peripheral is on the OPB (UART0).
DMA2P30_CR3[PW] = 000	Peripheral width is byte (8 bits).
DMA2P30_CR3[TM] = 00	DMA channel 3 is in peripheral mode.
DMA2P30_CR3[PWC] = 000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR3[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR3[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC3[SCID] = 000	DMA Channel 3 sub-channel 0 selected.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 3 for UART 1 transmitter transfers, external DMA transfers cannot be performed on this channel.	

Table 26-8 lists required register settings for UART2 transmit transfers. The UART2 transmit channel is tied to DMA channel 1. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-8. UART2 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART2[U2DTE] = 1	UART2 DMA transmit channel is enabled using DMA channel 1
SDR0_UART2[U2DC]	Set to 0 to not clear SDR0_UART2[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR1[TD] = 0	DMA channel 1 transfer direction is from memory to peripheral.
DMA2P30_CR1[PL] = 1	DMA channel 1 peripheral is on the OPB (UART0).
DMA2P30_CR1[PW] = 000	Peripheral width is byte (8 bits).
DMA2P30_CR1[TM] = 00	DMA Channel 1 is in peripheral mode.
DMA2P30_CR1[PWC] = 000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR1[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR1[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC1[SCID] = 001	DMA Channel 1 sub-channel 1 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 1 for UART0 transmitter transfers, external DMA transfers cannot be performed on this channel.	

Table 26-9 lists required register settings for UART3 transmit transfers. The UART3 transmit channel is tied to DMA channel 3. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-9. UART3 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART3[U3DTE] = 1	UART 3 DMA transmit channel is enabled using DMA channel 3.
SDR0_UART3[U3DC]	Set to 0 to not clear SDR0_UART3[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR3[TD] = 0	DMA channel 3 transfer direction is from memory to peripheral.
DMA2P30_CR3[PL] = 1	DMA channel 3 peripheral is on the OPB (UART0).
DMA2P30_CR3[PW] = 000	Peripheral width is byte (8 bits).
DMA2P30_CR3[TM] = 00	DMA channel 3 is in peripheral mode.
DMA2P30_CR3[PWC] = 000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR3[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR3[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC3[SCID] = 001	DMA Channel 3 sub-channel 1 selected.
UART3_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 3 for UART 3 transmitter transfers, external DMA transfers cannot be performed on this channel.	

26.5.2 Receiver DMA Mode

The four DMA channels on the PPC440EP can be used to move data to and from the two UARTs. The UART0 and UART2 receive channel is tied to DMA channel 0 and the UART1 and UART3 receive channel is tied to DMA channel 2. Use of the DMA with the UARTs is controlled by the UxDRE bits of the SDR0_UART0 register.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when there is at least one character in the RX FIFO or Receive Buffer Register (RBR). Once activated, the DMA request goes inactive when there are no more characters in the FIFO or RBR.

When UARTx_FCR[DMS] is 1 the DMA request goes active when the FIFOs are enabled and the trigger level or the timeout has been reached. DMA request goes inactive when there are no more characters in the FIFO or RBR. To operate in this mode the corresponding PPC440EP DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA2P30_CRx to 1configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 26-10 lists required register settings for UART0 receive transfers. The UART0 receive channel is tied to DMA channel 0. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-10. UART0 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART0[U0DRE] = 1	UART0 DMA receiver channel is enabled using DMA channel 0.
SDR0_UART0[U0DC]	Set to 0 to not clear SDR0_UART0[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR0[TD] = 1	DMA channel 0 transfer direction is from peripheral to memory.
DMA2P30_CR0[PL] = 1	DMA channel 0 peripheral is on the OPB (UART0).
DMA2P30_CR0[PW] = 000	Peripheral width is byte (8 bits).
DMA2P30_CR0[TM] = 00	DMA channel 0 is in peripheral mode.
DMA2P30_CR0[PWC] = 000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR0[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR0[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC0[SCID] = 000	DMA Channel 0 sub-channel 0 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 0 for UART0 receiver transfers, external DMA transfers cannot be performed on this channel.	

Table 26-11 lists required register settings for UART1 receive transfers. The UART1 receive channel is tied to DMA channel 2. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-11. UART1 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART1[U1DRE] = 1	UART1 DMA receiver channel is enabled using DMA channel 2.
SDR0_UART1[UIDC]	Set to 0 to not clear SDR0_UART1[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR2[TD] = 1	DMA Channel 2 transfer direction is from peripheral to memory.
DMA2P30_CR2[PL] = 1	DMA Channel 2 peripheral is on the OPB (UART0).
DMA2P30_CR2[PW] = 00	Peripheral width is byte (8 bits).
DMA2P30_CR2[TM] = 00	DMA Channel 2 is in peripheral mode.
DMA2P30_CR2[PWC] = 000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR2[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR2[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC2[SCID] = 000	DMA Channel 2 sub-channel 0 selected.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 2 for UART1 receiver transfers, external DMA transfers cannot be performed on this channel.	

Table 26-12 lists required register settings for UART2 receive transfers. The UART2 receive channel is tied to DMA channel 0. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-12. UART2 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART2[U2DRE] = 1	UART2 DMA receiver channel is enabled using DMA channel 0.
SDR0_UART2[U2DC]	Set to 0 to not clear SDR0_UART2[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR0[TD] = 1	DMA channel 0 transfer direction is from peripheral to memory.
DMA2P30_CR0[PL] = 1	DMA channel 0 peripheral is on the OPB (UART0).
DMA2P30_CR0[PW] = 000	Peripheral width is byte (8 bits).
DMA2P30_CR0[TM] = 00	DMA channel 0 is in peripheral mode.
DMA2P30_CR0[PWC] = 000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR0[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR0[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC0[SCID] = 001	DMA Channel 0 sub-channel 1 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 0 for UART2 receiver transfers, external DMA transfers cannot be performed on this channel.	

Table 26-13 lists required register settings for UART3 receive transfers. The UART3 receive channel is tied to DMA channel 2. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 509 for more information.

Table 26-13. UART3 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART3[U3DRE] = 1	UART3 DMA receiver channel is enabled using DMA channel 2.
SDR0_UART3[U3DC]	Set to 0 to not clear SDR0_UART3[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR2[TD] = 1	DMA Channel 2 transfer direction is from peripheral to memory.
DMA2P30_CR2[PL] = 1	DMA Channel 2 peripheral is on the OPB (UART0).
DMA2P30_CR2[PW] = 00	Peripheral width is byte (8 bits).
DMA2P30_CR2[TM] = 00	DMA Channel 2 is in peripheral mode.
DMA2P30_CR2[PWC] = 000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR2[PHC] = 000	Peripheral hold cycles are 0.
DMA2P30_CR2[ETD] = 1	EOT/TC is programmed as terminal count output.
DMA2P30_SC2[SCID] = 001	DMA Channel 2 sub-channel 1 selected.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 2 for UART1 receiver transfers, external DMA transfers cannot be performed on this channel.	

27. Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a full-duplex, synchronous, character-oriented (byte) port that enables the exchange of data with other serial devices. The SPI is master on the serial port supporting a 4-wire interface (receive, transmit, clock, and slave select). The SPI is also a slave device to the on-chip peripheral bus (OPB) for communication to the processor.

The Serial Peripheral Interface has the following features:

- 4-wire serial port interface
- Full-duplex synchronous operation
- SPI bus master
- OPB bus slave
- Programmable clock rate divider
- Clock inversion
- Optional clock phase
- Reverse data
- Local data loop back for test

27.1 Functional Description

Figure 27-1 shows the functional diagram of the port. The SPI accepts instructions from the OPB to read/write the control registers, or to read/write data from/to the data buffers, RxD and TxD. The SPI can then accept an instruction to initiate a transmit/receive cycle when SPI control register serial data transmit/receive bit is enabled (SPI0_CR[STP] = 1) (see *SPI Control Register (SPI0_CR)* on page 656). This activity is controlled by the OPB interface circuit.

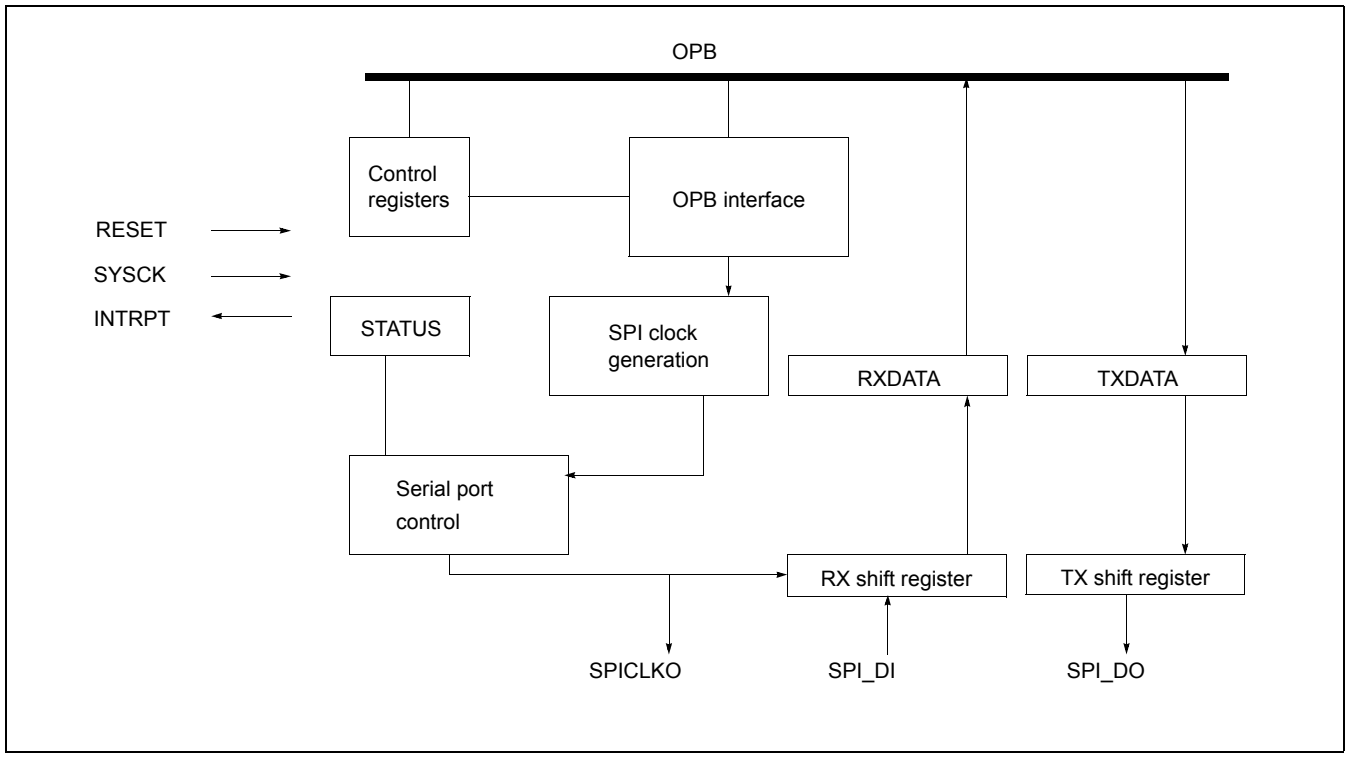


Figure 27-1. SPI Functional Diagram

If a send/receive cycle is initiated by the OPB, then the serial port control circuit enables the SPI clock for a series of exactly eight clock pulses, shifting eight bits into and eight bits out of the RX and TX shift registers. When this is complete, the SPI clock is again disabled, and an interrupt is generated so the port can be serviced by the interrupt controller. The SPI is always in full duplex mode and always sends and receives one byte of data. The SPI clock is generated internally from the OPB clock (OPBClk) using a clock divider which can be programmed via the SPI0_CDM register (see *SPI Clock Divisor Modulus Register (SPI0_CDM)* on page 657).

The OPB interface uses 8-bit input and output data buses, referred to as OPB_DBUS[0:7] for input and SL_DBUS[0:7] for output. If an SPI register is addressed for write, then its bits are replaced by the respective bits in the OPB_DBUS. Unused control register bits are reserved and should always be written with 0.

If the SPI is addressed for a register read, then the register bits are placed onto the SL_DBUS and the unused or reserved bits are set to 0. For example, if the OPB reads the SPI0_MODE register, then the SL_DBUS is loaded with 00034567 where 34567 is the content of bits 3, 4, 5, 6, and 7 of the SPI0_MODE register. Reserved bits are set to 0 on a register read.

27.1.1 SPI Interrupt Operation

A single interrupt is generated when the send/receive cycle is complete (RxByteReady). The interrupt is cleared when the received data is read by the OPB. Therefore, even if the received data is not used by the processor, a read operation is still required to clear the interrupt.

27.1.2 Loopback

In loop back operation, the output data is inverted and received into the input receiver.

27.1.3 Typical Operation (Exchange of Data)

The SPI operates as a slave to the OPB. In a typical exchange of data, the processor initiates a data exchange with an external peripheral (an SPI slave). The processor activates the enable signal for that slave using one of the general purpose I/O pins.

To begin the exchange of data, the processor writes the data to be transmitted into the SPI0_TxD register, and then writes the start SPI0_SR[STR] register transmit bit to start the transmission of data. SPI0_SR[STR] is cleared by the SPI after transmission has begun on the serial port.

When the serial port controller recognizes the STR bit, it loads the TxData into the transmit shift register and generates eight clock pulses that simultaneously transmit and receive eight bits of data. The received data is then transferred to the RxData buffer, and an interrupt is issued to the interrupt controller. When the OPB reads the data from the SPI, the interrupt is cleared.

TxData is loaded and SPI0_SR[STR] is cleared when the send/receive cycle is started. Therefore, if SPI0_SR[STR] is written again before this register can be cleared, the new request is lost. If new transmit data is written to the TxData buffer before SPI0_SR[STR] is cleared (that is, old data not yet sent), then the old data in the TxData buffer is overwritten.

27.2 SPI Registers

SPI registers listed on *Table 27-1* are 8-bit wide memory mapped registers. Some registers have reserved bits which should be written as 0, and return 0 when read.

Table 27-1. SPI Registers

Mnemonic	R/W Mode	Offset[28:31]	Register	Page
SPI0_MODE	R/W	0x0 EF60 0900	SPI Mode Register	658
SPI0_RxD	R	0x0 EF60 0901	SPI Receive Data Register	656
SPI0_TxD	R/W	0x0 EF60 0902	SPI Transmit Data Register	656
SPI0_CR	R/W	0x0 EF60 0903	SPI Control Register	656
SPI0_SR	R	0x0 EF60 0904	SPI Status Register	657
SPI0_CDM	R/W	0x0 EF60 0906	SPI Clock Divisor Modulus Register	657

27.2.1 SPI Receive Data Register (SPI0_RxD)

The read-only SPI0_RxD register holds data received on the serial port.

Figure 27-2. SPI Receive Data Register (SPI0_RxD)

0	Rx0	Receive data 0	
1	Rx1	Receive data 1	
2	Rx2	Receive data 2	
3	Rx3	Receive data 3	
4	Rx4	Receive data 4	
5	Rx5	Receive data 5	
6	Rx6	Receive data 6	
7	Rx7	Receive data 7	

27.2.2 SPI Transmit Data Register (SPI0_TxD)

The read/write SPI0_TxD register holds data transmitted on the serial port.

Figure 27-3. SPI Transmit Data Register (SPI0_TxD)

0	Tx0	Transmit data 0	
1	Tx1	Transmit data 1	
2	Tx2	Transmit data 2	
3	Tx3	Transmit data 3	
4	Tx4	Transmit data 4	
5	Tx5	Transmit data 5	
6	Tx5	Transmit data 6	
7	Tx7	Transmit data 7	

27.2.3 SPI Control Register (SPI0_CR)

Figure 27-4 describes the read/write SPI0_CR register. Asserting SPI0_CR[STR] is a request that starts a new transmit/receive cycle on the serial port. When SPI0_CR[STR]=1, and the SPI is idle, a new byte of data is loaded into the output shift register and a new transmit/receive cycle begins. STR is cleared by the SPI when the transmit/receive begins. The delay between the setting of SPI0_CR[STR]=1 and the beginning of transmit is variable and depends on the SPI port frequency.

If the port is busy, SPI0_CR[STR] is a pending request and cleared when the next send/receive cycle begins. If SPI0_CR[STR] is asserted twice before it can be cleared, the request is lost.

Note: SPI0_CDM, SPI0_MODE, and SPI0_TxD registers should be written in order to configure the SPI before writing SPI0_CR.

Figure 27-4. SPI Control Register (SPI0_CR)			
0:6		Reserved	
7	STR	Serial data transmit/receive. 0 Serial data transmit/receive disabled 1 Serial data transmit/receive enabled	Reset to '0' after serial transmission has begun. Setting SPI0_CR[STR]=1 initiates transmit and receive of one character on the SPI.

27.2.4 SPI Clock Divisor Modulus Register (SPI0_CDM)

The SPI clock is generated from the OPB clock by a clock divider. The divide ratio is controlled by the value of the SPI0_CDM register, which is initially reset to 00000000. SPI0_CDM represents a programmable binary number (with bit 0 as the most significant bit (MSb)) that is used for the initial value of a count down register. When the count equals zero, the counter is again loaded with the value of SPI0_CDM. This counter is used to generate the output SPI clock (SCPClkOut). The output clock frequency is given by the ratio:

SCPClkOut = OPBCLK/(4(CDM+1)) where CDM = 0–255.

Therefore, the minimum divide ratio is 4 when CDM = 0, and the maximum is 1024 when CDM = 255.

Figure 27-5. SPI Clock Divisor Modulus Register (SPI0_CDM)			
0	CDM0	Clock divisor modulus 0	
1	CDM1	Clock divisor modulus 1	
2	CDM2	Clock divisor modulus 2	
3	CDM3	Clock divisor modulus 3	
4	CDM4	Clock divisor modulus 4	
5	CDM5	Clock divisor modulus 5	
6	CDM6	Clock divisor modulus 6	
7	CDM7	Clock divisor modulus 7	

27.2.5 SPI Status Register (SPI0_SR)

Figure 27-6 describes the SPI0_SR register bits. After a transmit/receive cycle is complete, the received data is loaded into the RxData buffer. The SPI0_SR[RBR] is asserted and this creates an interrupt. If RxData already contained previously received data, then that data is overwritten. SPI0_SR[RBR] also confirms that the transmit byte has been sent. SPI0_SR[BSY] is asserted one cycle after the start of send/receive and de-asserted one cycle after the send/receive is complete.

Figure 27-6. SPI Status Register (SPI0_SR)			
0:5		Reserved	
6	BSY	Busy 0 Serial port is idle. 1 Serial transmit/receive is active.	
7	RBR	RxByteReady 0 RxD buffer is empty. 1 Serial data receive is complete and RxD is available.	

27.2.6 SPI Mode Register (SPI0_MODE)

SPI is configured via SPI0_MODE register as described in *Figure 27-7* (SPI0_MODE[SPE]). If the other bits of SPI0_MODE and SPI0_CDM are changed while the serial port is active, the behavior will not be predictable. When the SPI0_MODE register is at its reset value (00000000), the output clock, SCPClkOut, is forced to 0.

Figure 27-7. SPI Mode Register (SPI0_MODE)			
0:2		Reserved	
3	SCP	Serial Clock Phase 0 Data is latched on trailing edge of clock 1 Data is latched on leading edge of clock	
4	SPE	Serial Port Enabled 0 Serial port disabled 1 Serial port enabled	SPI0_MODE[SPE] is used to enable the SPI port operation. When SPI0_MODE[SPE]=0 the serial port control circuits are put into a known reset state. When SPI0_MODE[SPE]=1, the port is enabled and ready to initiate a send/receive cycle. This bit must be asserted before transmission is started (writing to the SPI0_CR register). If SPI0_MODE[SPE]=0 while the port is transmitting, the current transmit cycle is completed before SPI0_MODE[SPE]=0 is recognized.
5	RD	Reverse Data 0 Data Bit 0 is transmitted first. 1 Data Bit 7 is transmitted first.	
6	CI	Clock Invert 0 Idle clock = '0', Active clock = '1'. 1 Idle clock = '1', Active clock = '0'.	
7	IL	Internal Loopback 0 Loopback is disabled. 1 Loopback is enabled.	When SPI0_MODE[IL]=1 the serial output data is inverted and connected to the receiver input.

28. IIC Bus Interface

The PPC440EP provides two inter-integrated circuit (IIC) bus interfaces, IIC0 and IIC1. These interfaces comply with the specifications contained in the Phillips ® Semiconductors document *The I^2C Bus and How to Use It* (including specifications—1995 update).

Each IIC bus has a two wire, bi-directional, open-drain, low-speed serial interface. The serial clock (IIC0SClk and IIC1SClk) and serial data (IIC0SDA and IIC1SDA) lines are bidirectional, to support multiple bus masters and to mix high- and low-speed devices on the same bus.

Each IIC interface (referred to as IIC to distinguish it from the Phillips I 2 C bus) supports the following standard and enhanced features:

- 100 kHz and 400 kHz operation
- 8-bit data transfers
- 7 bit and 10 bit addressing
- Slave transmitter and receiver
- Master transmitter and receiver
- Multiple bus masters

28.1 Addressing

The IIC interfaces support 7-bit and 10-bit addressing for master and slave transfers. Addressing is described in detail in *IICx Low Master Address Register (IICx_LMADR)* on page 663, *IICx High Master Address Register (IICx_HMADR)* on page 664, *IICx Low Slave Address Register (IICx_LSADR)* on page 669, and *IICx High Slave Address Register (IICx_HSADR)* on page 670.

Descriptions of addressing modes and address formats follow.

28.1.1 Addressing Modes

For master transfers, the address mode (AMD) field of the IIC Control register (IICx_CNTL) controls whether 7-bit or 10-bit addresses are used. If IICx_CNTL[AMD] = 0, addresses contain 7 bits; if IICx_CNTL[AMD] = 1, addresses contain 10 bits.

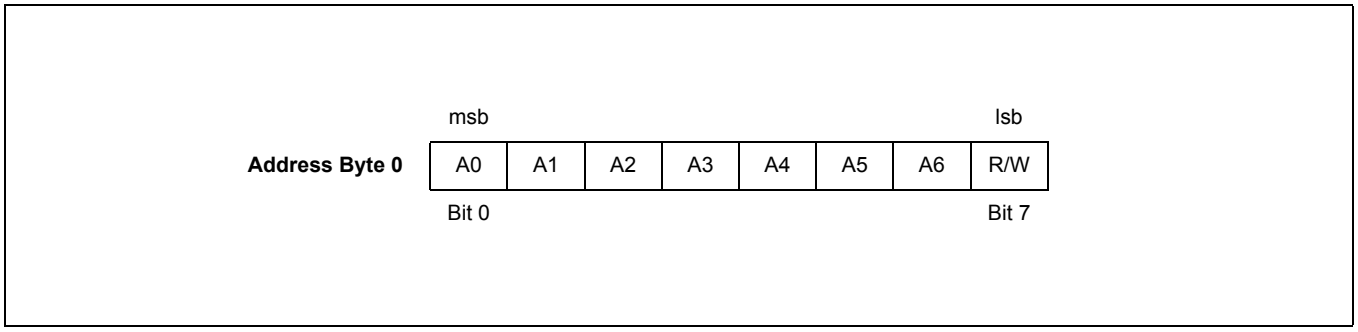
For slave transfers, the contents of the IICx High Slave Address register (IICx_HSADR) determines whether 7-bit or 10-bit addressing is used. If IICx_HSADR = 0b00000000, 7-bit addressing is used. If 10-bit addressing is to be used for slave transfers, IICx_HSADR = 0b11110*yy*x, where *yy* contains the high-order bits of the 10-bit address, and x is a don't care.

Programming Note: For slave transfers, IICx_CNTL[AMD] does not control addressing mode.

28.1.2 Seven-Bit Addresses

Figure 28-1 illustrates a 7-bit address. For master transfers, the address bits 0 through 6 (A0:A6) are read from IICx_LMADR. For slave transfers, A0:A6 are read from IICx_LSADR. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

Figure 28-1. 7-Bit Addressing



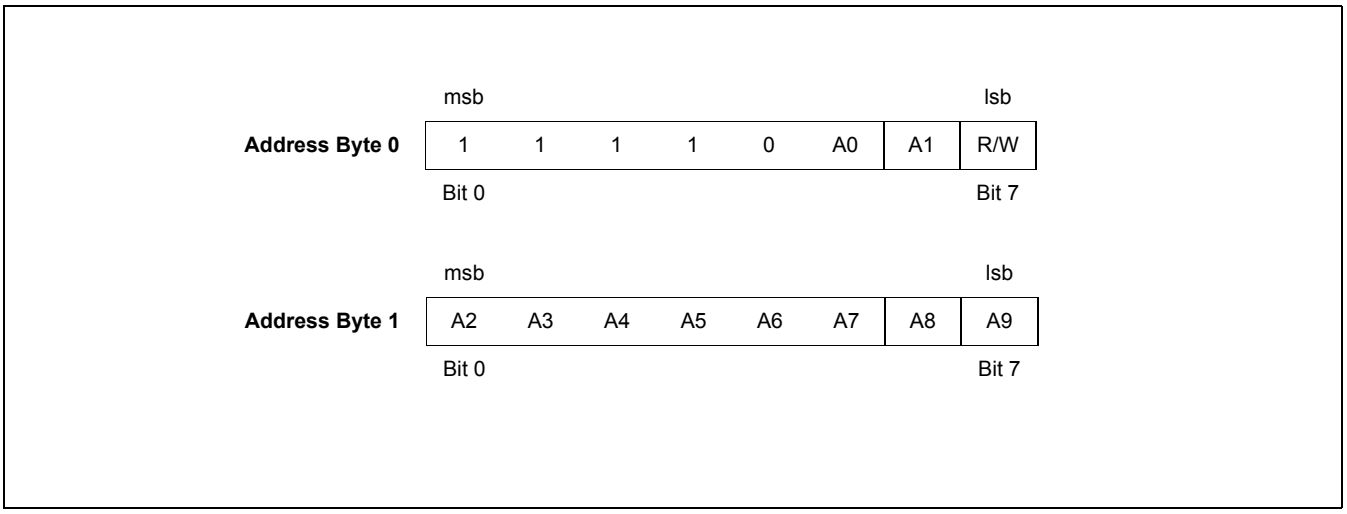
28.1.3 Ten-Bit Addresses

Figure 28-2 illustrates a 10-bit address. A0:A1 of address byte 0 are read from IICx_HMADR[A6:A7] (for master transfers) or IICx_HSADR[A6:A7] (for slave transfers). These are the two highest-order address bits transmitted on the IIC bus. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

For 10-bit addressing for master or slave transfers, respectively, IICx_HMADR[A0:A4] and IICx_HSADR [A0:A4] must contain 0b11110.

The low-order byte of the 10-bit address, contained in A0:A7 of address byte 1, are read from IICx_LMADR or IICx_LSADR for master or slave transfers, respectively.

Figure 28-2. 10-Bit Addressing



28.2 IIC Registers

Table 28-1 lists the IICx registers. Descriptions of the registers, in the listed order, follow in *IIC Register Descriptions* on page 662.

Table 28-1. IIC Registers

Register	Description	Address	Access	Effect of Reset	Bits	Page
IIC0_MDBUF	IIC0 Master Data Buffer	0x0 EF60 0700	R/W	Cleared	8,16	662
IIC0_SDBUF	IIC0x Slave Data Buffer	0x0 EF60 0702	R/W	Cleared	8,16	662
IIC0_LMADR	IIC0 Low Master Address	0x0 EF60 0704	R/W	No	8	663
IIC0_HMADR	IIC0 High Master Address	0x0 EF60 0705	R/W	No	8	664
IIC0_CNTL	IIC0 Control	0x0 EF60 0706	R/W	Cleared	8	664
IIC0_MDCNTL	IIC0 Mode Control	0x0 EF60 0707	R/W	Cleared	8	665
IIC0_STS	IIC0 Status	0x0 EF60 0708	R/W	Cleared	8	666
IIC0_EXTSTS	IIC0 Extended Status	0x0 EF60 0709	R/W	Cleared	8	668
IIC0_LSADR	IIC0 Low Slave Address	0x0 EF60 070A	R/W	No	8	669
IIC0_HSADR	IIC0 High Slave Address	0x0 EF60 070B	R/W	No	8	670
IIC0_CLKDIV	IIC0 Clock Divide	0x0 EF60 070C	R/W	Cleared	8	670
IIC0_INTRMSK	IIC0 Interrupt Mask	0x0 EF60 070D	R/W	Cleared	8	671
IIC0_XFRCNT	IIC0 Transfer Count	0x0 EF60 070E	R/W	Cleared	8	672
IIC0_XTCNTLSS	IIC0 Extended Control and Slave Status	0x0 EF60 070F	R/W	Cleared	8	673
IIC0_DIRECTCNTL	IIC0 Direct Control	0x0 EF60 0710	R/W	0x0F	4	674
IIC0_INTR	IIC0 Interrupt	0x0 EF60 0711	R	Cleared	2	675
IIC1_MDBUF	IIC1 Master Data Buffer	0x0 EF60 0800	R/W	Cleared	8,16	662
IIC1_SDBUF	IIC1 Slave Data Buffer	0x0 EF60 0802	R/W	Cleared	8,16	662
IIC1_LMADR	IIC1 Low Master Address	0x0 EF60 0804	R/W	No	8	663
IIC1_HMADR	IIC1 High Master Address	0x0 EF60 0805	R/W	No	8	664
IIC1_CNTL	IIC1 Control	0x0 EF60 0806	R/W	Cleared	8	664
IIC1_MDCNTL	IIC1 Mode Control	0x0 EF60 0807	R/W	Cleared	8	665
IIC1_STS	IIC1 Status	0x0 EF60 0808	R/W	Cleared	8	666
IIC1_EXTSTS	IIC1 Extended Status	0x0 EF60 0809	R/W	Cleared	8	668
IIC1_LSADR	IIC1 Low Slave Address	0x0 EF60 080A	R/W	No	8	669
IIC1_HSADR	IIC1 High Slave Address	0x0 EF60 080B	R/W	No	8	670
IIC1_CLKDIV	IIC1 Clock Divide	0x0 EF60 080C	R/W	Cleared	8	670
IIC1_INTRMSK	IIC1 Interrupt Mask	0x0 EF60 080D	R/W	Cleared	8	671
IIC1_XFRCNT	IIC1 Transfer Count	0x0 EF60 080E	R/W	Cleared	8	672
IIC1_XTCNTLSS	IIC1 Extended Control and Slave Status	0x0 EF60 080F	R/W	Cleared	8	673

Table 28-1. IIC Registers (continued)

Register	Description	Address	Access	Effect of Reset	Bits	Page
IIC1_DIRECTCNTL	IIC1 Direct Control	0x0 EF60 0810	R/W	0x0F	4	674
IIC1_INTR	IIC0 Interrupt	0x0 EF60 0811	R	Cleared	2	675

28.3 IIC Register Descriptions

The following sections contains the bit definitions for the various registers in the IIC interface.

28.3.1 IICx Master Data Buffer (IICx_MDBUF)

The IICx Master Data Buffer (IICx_MDBUF) is a 1-byte × 4-byte first-in/first-out (FIFO) buffer. Data placed in IICx_MDBUF is written onto the IIC bus when performing a master write operation. Data received from the IIC bus is read from IICx_MDBUF when performing a master read operation.

Figure 28-3. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit	
1		Data bit	
2		Data bit	
3		Data bit	
4		Data bit	
5		Data bit	
6		Data bit	
7		Data bit	

Half word reads from and writes to the IIC bus assume the MSB comes first. For half word writes, the first byte written to the IICx_MDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For half word reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_MDBUF while the FIFO is empty, obsolete data is read. Check the master buffer status, IICx_STS[MDBS], before reading IICx_MDBUF.

If a byte is written to IICx_MDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the master buffer full status, IICx_STS[MDBF], before writing IICx_MDBUF.

IICx_MDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FMDB] = 1.

28.3.2 IICx Slave Data Buffer (IICx_SDBUF)

The IICx Slave Data Buffer (IICx_SDBUF) is a 1-byte 4-byte first-in/first-out (FIFO) buffer. The data contained in IICx_SDBUF is either received from the IIC bus when the IIC interface is addressed as a slave during a write, or is written on the IIC bus when the IIC interface is addressed as a slave during a read operation.

Figure 28-4. IICx Slave Data Buffer (IICx_SDBUF)			
0		Data bit	
1		Data bit	
2		Data bit	
3		Data bit	
4		Data bit	
5		Data bit	
6		Data bit	
7		Data bit	

Half word reads from and writes to the IIC bus assume the MSB comes first. For half word writes, the first byte written to the IICx_SDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For half word reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_SDBUF while the FIFO is empty, obsolete data is read. Check the slave buffer status, IICx_XTCNTLSS[SDBD], before reading IICx_SDBUF.

If a byte is written to IICx_SDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the slave buffer full status, IICx_XTCNTLSS[SDBF], before writing IICx_SDBUF.

IICx_SDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FSDB] = 1.

28.3.3 IICx Low Master Address Register (IICx_LMADR)

The IICx Low Master Address (IICx_LMADR) and IICx High Master Address Register (IICx_HMADR) form addresses that the IIC interface transmits on the IIC bus.

When in 7-bit address mode, IICx_CNTL[AMD] = 0, IICx_LMADR[A0:A6] is the address transmitted on the IIC bus; IICx_LMADR[A7] is a don't care. When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_LMADR[A0:A7] is the second byte of the address transmitted on the IIC bus.

Figure 28-5. IICx Low Master Address Register (IICx_LMADR)			
0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

28.3.4 IICx High Master Address Register (IICx_HMADR)

IICx High Master Address Register (IICx_HMADR) provides the upper address bits in 10-bit addressing mode.

When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_HMADR must be programmed to 0b1111 0yyz, where yy are the high-order bits of a 10-bit address and z is a don't care.

IICx_HMADR[A5:A6] are the two most significant bits of the 10-bit address. IICx_HMADR[A7] is a don't care.

Figure 28-6. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

28.3.5 IICx Control Register (IICx_CNTL)

The IICx Control Register (IICx_CNTL) starts and stops IIC interface master transfers on the IIC bus. When a transfer begins, the IIC interface uses the values in IICx_CNTL to determine the type and size of the transfer.

> **Programming Note:** IICx_CNTL *must* be the last register programmed. Whenever IICx_CNTL[PT] = 1, the IIC interface attempts to perform the requested transfer using values set in other registers.

During and after transfers, the IICx_STS and IICx_EXTSTS registers can be read to determine the state of the IIC interface and the IIC bus.

Only IICx_CNTL[PT] is cleared when a requested master transfer is complete; the remaining bits are not affected.

Figure 28-7. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	

5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

Table 28-2 summarizes IIC interface operation for settings of IICx_CNTL[HMT, RPST, CHT, PT]. x represents a don't care in the RPST, CHT and PT columns.

Table 28-2. IIC Response to IICx_CNTL Field Settings

IICx_CNTL Fields				Resulting Action on IIC Bus and Inside IIC Interface
HMT	RPST	CHT	PT	
0	x	x	0	No action taken
0	0	1	1	Start, Transfer, ACK on last byte, Pause
0	0	0	1	Start, Transfer, NACK on last byte, Stop
1	x	x	x	NACK on current byte, Stop
0	1	x	1	Start, Transfer, NACK on last byte, Wait

Settings of IICx_CNTL[HMT, RPST, CHT, PT] result in the following actions.

Start IIC Start condition generated, if the IIC interface was stopped or waiting.

Stop IIC Stop condition generated; IIC interface enters the Stop condition.

ACK IIC Acknowledge condition generated.

NACK IIC Not Acknowledge condition generated, if performing a read.

Transfer Requested bytes are transferred.

Pause IIC interface enters the Pause state.

Wait IIC interface enters the Wait state.

IICx_CNTL[HMT] overrides IICx_CNTL[RPST, CHT].

IICx_CNTL[RPST, PT] overrides IICx_CNTL[CHT].

28.3.6 IICx Mode Control Register (IICx_MDCNTL)

The IICx Mode Control Register (IICx_MDCNTL) sets the major modes of operation on the IIC bus. In addition, IICx_MDCNTL can force the data buffers into the empty state.

In typical applications, IICx_MDCNTL is configured once, during software initialization. Applications providing complex error handling may reconfigure this register more often.

Programming Note: IICx_CLKDIV must be initialized before IICx_MDCNTL. IICx_LSADR and IICx_HSADR should also be configured before IICx_MDCNTL.

Note that the IIC hardware does not implement time-out functions on the IIC bus. Such functions must be implemented, in software, by setting IICx_CNTL[HMT] = 1, or setting IICx_XTCNTLSS[SRST] = 1.

Regarding IICx_MDCNTL[HSCL], a "slave not ready" condition occurs during a slave receive operation, if there is no space in the slave data buffer at the start of a write operation, or if the slave data buffer fills during the write. In a slave transmit operation, a slave not ready condition occurs if there is no data in the slave data buffer at the start of a read operation, or if the slave data buffer becomes empty during the read.

Using IICx_MDCNTL[HSCL] to handle slave not ready conditions can affect system performance. A slave holding the IICSCL signal low guarantees data delivery to or from the requesting master, but prevents other masters from performing transfers over the IIC bus. There is no general rule for handling slave not ready conditions; each system has its own requirements.

Figure 28-8. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICSCL signal low until slave is ready.	This field is used only when in slave mode.

28.3.7 IICx Status Register (IICx_STS)

The IICx Status register (IICx_STS) contains the state of the IIC interface and the status of any previously requested master transfers.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Programming Note: IICx_STS should be the first register read by an interrupt or error handler routine. IICx_STS can also be read in a polling loop if the IIC interrupts are not used.

All IICx_STS bit fields except for IICx_STS[SSS] must be cleared before requesting another master transfer. IICx_STS[SSS] is not affected by master transactions.

Figure 28-9. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRC, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (SDR0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the SDR0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

The Error and Pending Transfer, IICx_STS[ERR, PT], bit fields indicate the success or failure of the requested transfer. *Table 28-3* lists the transfer status for all combinations of the IICx_STS[ERR,PT] bit fields.

Table 28-3. IICx_STS[ERR, PT] Decoding

ERR	PT	Status
0	0	Requested transfer completed without errors
0	1	Requested transfer is in progress; no errors were detected
1	0	Requested transfer is complete, but not all data was transferred
1	1	Requested transfer is in progress; but an error was detected

Programming Note: No action regarding a master transfer should be taken unless all pending transfers have completed, IICx_STS[PT] = 0.

If an error requires the IIC interface to send a Stop, the Stop Complete bit field is set, IICx_STS[SCMP] = 1. Note that slave operations should be serviced regardless of the state of a requested master transfer.

IIC1_MDCNTL[EUBS] must be set after a reset before IIC interface can be placed in sleep mode. The IIC interface is placed in sleep mode by setting the CPM0_ER[IICx] via software. Awaking the IIC interface is possible directly through software by clearing the CPM0_ER[IICx] or indirectly by detecting a Start condition on the IIC bus. When a Start condition is detected, the IIC interface is awakened, clearing the CPM0_ER[IICx] and the IICx_STS[SLPR].

The IICx_STS[MDBS, MDBF] contain the current status of the Master Data Buffer, IICx_MDBUF. When the IICx_MDBUF contains data, IICx_STS[MDBS] is set. When the IICx_MDBUF is full, IICx_STS[MDBF] is set.

The state of the IICx_MDBUF is not instantly recorded by the IICx_STS[MDBS, MDBF]. The delay depends on the size of the buffer access. For half word accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

28.3.8 IICx Extended Status Register (IICx_EXTSTS)

The IICx Extended Status register (IICx_EXTSTS) reports additional IIC status.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Figure 28-10. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. • Writing 1 to IICx_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	Bus Control State 000 Unused; if this value is read an error occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read an error occurred.	Read-only.

4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.
7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

IICx_EXTSTS[IRQP, IRQD] and IICx_STS[IRQA] provide a FIFO for storing interrupts. A new interrupt is considered pending, and remains pending while an on-deck interrupt is present. Once the on-deck interrupt becomes active, the pending interrupt moves on-deck, and remains on-deck until there is no active interrupt. When the active interrupt is cleared, the on-deck (initially pending) interrupt becomes active.

Programming Note: An active interrupt remains active until software clears it.

IICx_EXTSTS[BCS] indicates the state of the IIC interface. The field is read-only.

IICx_EXTSTS[LA, ICT, XFRA] are cleared when IICx_EXTSTS[XFRA] = 1.

Writing 1 to IICx_EXTSTS[IRQP, IRQD, LA, ICT, XFRA] clears these fields.

28.3.9 IICx Low Slave Address Register (IICx_LSADR)

The IICx Low Slave Address Register (IICx_LSADR) and IICx_ High Slave Address Register (IICx_HSADR) program the slave address of the IIC interface. IICx_HSADR is used only for 10-bit addressing, and is not programmed in 7-bit addressing mode.

When 7-bit addressing is used, IICx_LSADR is written with the slave address; IICx_HSADR must be written with zeros. For 7-bit addressing, IICx_LSADR[A0:A6] contain the address transmitted on the IIC bus; IICx_LSADR[A7] is a don't care.

When 10-bit addressing is used, IICx_LSADR[A0:A7] contain the second address byte transmitted on the IIC bus.

Figure 28-11. IICx Low Slave Address Register (IICx_LSADR)			
0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

28.3.10 IICx High Slave Address Register (IICx_HSADR)

For 7-bit addressing, set IICx High Slave Address Register (IICx_HSADR) to 0.

To enable 10-bit slave addressing, IICx_HSADR must be programmed to 0b1111 0*yyx*, where *yy* are the high-order bits of a 10-bit address and *x* is a don't care.

Programming Note: IICx_HSADR is used only for 10-bit addressing, and should be set to 0 for 7-bit addressing mode.

Thus, in 10-bit address mode, IICx_HSADR[A6:A7] contain the two highest -order bits of the 10-bit address; IICx_HSADR[A7] is a don't care. IICx_LSADR contains the low-order byte of the 10-bit address.

Figure 28-12. IICx High Slave Address Register (IICx_HSADR)			
0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

Thus, in 10-bit address mode, bits 0:6 are used to decode the first address byte that was transmitted on the IIC bus, and bit 7 is in a don't care.

28.3.11 IICx Clock Divide Register (IICx_CLKDIV)

The IICx Clock Divide Register (IICx_CLKDIV) establishes a reference between the OPB clock and the IIC bus serial clock.

Programming Note: IICx_CLKDIV must be initialized before IICx_MDCTRL. Until IICx_CLKDIV is initialized, all IIC bus activity is ignored.

Figure 28-13. IICx Clock Divide Register (IICx_CLKDIV)			
0	DIV0	Divisor bit 0	
1	DIV1	Divisor bit 1	
2	DIV2	Divisor bit 2	
3	DIV3	Divisor bit 3	
4	DIV4	Divisor bit 4	
5	DIV5	Divisor bit 5	
6	DIV6	Divisor bit 6	
7	DIV7	Divisor bit 7	

IICx_CLKDIV divides PPC440EP's on-chip peripheral bus (OPB) clock to form the base clock for the IIC bus.

Table 22-4 lists the divisor values for several OPB frequency ranges. These divisor values apply for standard and fast mode. Select the divisor value by matching the OPB clock frequency to the corresponding frequency range in Table 22-4. For example, if the OPB clock frequency is 50MHz, select a divisor value of 0x4.

Table 28-4. IICx Clock Divide Programming

OPB Frequency Range (MHz)	Divisor Value
20	0x1
$20 < f \leq 30$	0x2
$30 < f \leq 40$	0x3
$40 < f \leq 50$	0x4
$50 < f \leq 60$	0x5
$60 < f \leq 70$	0x6
$70 < f \leq 80$	0x7
$80 < f \leq 90$	0x8
$90 < f \leq 100$	0x9
$100 < f \leq 110$	0xA
$110 < f \leq 120$	0xB
$120 < f \leq 130$	0xC
$130 < f \leq 140$	0xD
$140 < f \leq 150$	0xE

28.3.12 IICx Interrupt Mask Register (IICx_INTRMSK)

The IICx Interrupt Mask Register (IICx_INTRMSK) specifies which conditions can generate an IIC interrupt when the IIC interrupt is enabled, IICx_MDCNTL[EINT]=1.

Figure 28-14. IICx Interrupt Mask Register (IICx_INTRMSK)			
0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SRC] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. **Note:** IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

28.3.13 IICx Transfer Count Register (IICx_XFRCNT)

The IICx Transfer Count Register (IICx_XFRCNT) reports the number of bytes transferred on the IIC bus during a master or a slave operation.

Figure 28-15. IICx Transfer Count Register (IICx_XFRCNT)			
0		Reserved	
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved	
4		Reserved	

5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved	

IICx_XFRCNT[MTC] is cleared when there is a pending transfer, IICx_CNTL[PT] = 1.

IICx_XFRCNT[STC] is cleared when:

- A slave operation starts on the IIC bus
- Software indicates the slave does not need service by clearing IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SWS].

28.3.14 IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

The IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) provides additional control of IIC interface functions and reports the status of slave operations.

Figure 28-16. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IICx_SDBUF.
1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set. 2. If ICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data. 3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IIC-SCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data. 4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.
2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	

3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IICx_SDBUF is full during a slave write.	1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IICSCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.
4	SDBD	Slave Data Buffer Has Data 0 IICx_SDBUF is empty 1 IICx_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

Writing a 1 to IICx_XTCNTLSS[SRC, SRS, SWC, SWS] clears these fields.

The IICx_XTCNTLSS[SBSS, SDBF] contain the current status of the Slave Data Buffer, IICx_SDBUF. When the IICx_SDBUF contains data, IICx_XTCNTLSS[SDBD] is set. When the IICx_SDBF is full, IICx_XTCNTLSS[SDBF] is set.

The state of the IICx_SDBUF is not instantly recorded by the IICx_XTCNTL[SDBD, SDBF]. The delay depends on the size of the buffer access. For half-word accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

If any of the following fields: IICx_XTCNTLSS[SRC, SRS, SWC, SWS] = 1 and IICx_MDCNTL[HSCL] = 0; no new slave operations will be accepted over the IIC bus. A NACK is issued until IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], and IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], are cleared.

Soft reset, IICx_XTCNTLSS[SRST], provides a last means of recovery from IIC interface or IIC bus failure. Once enabled, soft reset completely resets the IIC interface. All IIC registers are affected. All transmissions from the IIC interface are terminated. Enabling soft reset during an IIC transmission may improperly terminate the transmission and hang the IIC bus.

28.3.15 IICx Direct Control Register (IICx_DIRECTCNTL)

The IICx Direct Control Register (IICx_DIRECTCNTL), which controls and monitors the IIC serial clock (IICSCL) and serial data (IICSDA) signal, is used for error recovery when a malfunction is detected on the IIC interface.

Figure 28-17. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved	
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1	

5	SCLC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSCL	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1	Read-only

IICx_DIRECTCNTL[SDAC, SCLC] can be written to control the IICSDA and IICSCL signals. When controlling the IICSDA and IICSCL signals directly, the IIC controller must be placed in the reset state, IICx_XTCNTL[SRST] = 1.

IICx_DIRECTCNTL[MSDA, MSCL] are used to verify that IICx_DIRECTCNTL[SDAC, SCLC] were written successfully, and that the IICSCL signals can be controlled. If IICx_DIRECTCNTL[MSDA, MSCL] do not correspond to IICx_DIRECTCNTL[SDAC, SCLC], respectively, toggle IICSCL repeatedly to regain control.

IICx_DIRECTCNTL[SDAC, SCLC, MDSA, MSCL] = 1 after a chip or system reset. A Soft Reset, IICx_XTCNTLSS[SRST] = 1, does not affect the state of IICx_DIRECRCNTL.

28.3.16 IICx Interrupt Register (IICx_INTR)

This register is used for diagnostic purposes, providing information on queued or pending interrupts. This register is not visible on the OPB if Enable_32byte_window is false.

Bit 1 is cleared when the on-deck interrupt is cleared and the pending IRQ has become the on-deck IRQ. Bit 0 is cleared when the active interrupt is cleared and the on-deck IRQ has become the active IRQ. Both bits are cleared by a hard or a soft reset.

The pending and on-deck interrupts, along with the active interrupt in the status register, form a miniature FIFO for storing the interrupts. A new interrupt is first set into the pending state. It will stay pending as long as an on-deck interrupt is present. Once the on-deck interrupt is cleared, or if none was present at the time the new interrupt occurred, the pending interrupt is moved into the on-deck state. An on-deck interrupt remains in the on-deck state as long as an active interrupt is present. Once the active interrupt is cleared, or if none was present at the time the pending interrupt went to the on-deck state, the on-deck interrupt is moved into the active state. Note that an active interrupt remains in the active state until it is cleared by the program.

Figure 28-18. IICx Interrupt Register (IICx_INTR)

0	IRQD	IRQ On-Deck Set when an IRQ is still active and another interrupting condition was generated.	This bit might also be momentarily set while a new interrupting condition moves from the on-deck to the active state. When IRQ active is set to 0, an on-deck IRQ will cause IRQ active to be set 1, and IRQ on-deck will be cleared. If there is a pending IRQ, then IRQ on-deck will be set to 1 on the next system clock. The on-deck interrupt can be cleared by writing 1 to this bit. If interrupts are disabled, then bit 2 in the Mode Control Register is 0, and this bit will not be set.

1	IRQP	IRQ Pending Set when an IRQ is still active, an IRQ is on-deck, and another interrupting condition was generated.	This bit might also be momentarily set while a new interrupting condition moves from the pending to the on deck state. When IRQ on-deck is set to 0, a pending IRQ will cause IRQ on-deck to be set to 1, and IRQ pending will be cleared. The pending interrupt can be cleared by writing 1 to this bit. If interrupts are disabled, then bit 2 in the Mode Control Register is 0, and this bit will not be set.
2:7		Reserved	

28.4 Interrupt Handling

Service request on the IIC interface can be monitored by polling the IICx_STS[SSS, PT] or by using the IIC interrupt to the UIC. When the IICx_MDCNTL[EINT] is enabled, the IIC interface generates an interrupt to the UIC if an unmasked IIC interrupt condition occurs. The interrupt is recorded by the UIC if UIC0_ER[IICx] is enabled. The conditions that generate an IIC interrupt to the UIC are determined by the interrupt mask settings in IICx_INTMSK.

The IIC interface can queue up to three interrupts since there is one interrupt for master operations and two for slave operations. The current interrupt is referred to as the *active* interrupt. The first interrupt in the queue is the on-deck interrupt; the second queued interrupt is called the pending interrupt. The queue holds multiple interrupts until the active interrupt is cleared by writing a 1 to IICx_STS[IRQA]. When an active interrupt is cleared, the on-deck interrupt becomes the active interrupt and the pending interrupt becomes the on-deck interrupt.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

Under certain conditions, the IIC interface merges slave read (write) needs service and slave read (write) complete interrupts into one interrupt. If a slave read (write) needs service interrupt is active, or queued, and a slave read (write) complete interrupt occurs, and IICx_XTCNTLSS has not yet been read, the two interrupts are merged into a single interrupt. This merge function is performed in the IIC interface logic, and is not under software control.

28.5 General Considerations

1. After a reset, the IIC interface enters the unknown IIC bus state. This state is exited when either activity is seen on the bus or when the exit unknown IIC bus state bit (in the mode control register), is set to a 1. If the IIC interface is being used in a single master system as the master, then the exit unknown IIC bus state bit must be used to force the logic out of the unknown state.
2. Once a byte is written into either the master or slave buffer, a total of four OPB clock periods must occur before the data can be read. Flushing the master or slave buffer also requires four OPB clock periods to complete.
3. IICx_DIRECTCNTL [MSDA,MSC]are used to verify that IICx_DIRECTCNTL [SDAC,SCC] were written successfully,and that the IICSCL signals can be controlled.If IICx_DIRECTCNTL [MSDA, MSC] do not correspond to IICx__DIRECTCNTL [SDAC,SCC],respectively, toggle IICSCL repeatedly to regain control.

4. The master and slave buffers are 4 × 1 byte-wide FIFOs. Exercise care when using master and slave buffers. As an example, consider the case where one byte of data is written on the IIC bus. The data is first written into the master or slave buffer. After four OPB clock cycles the data is placed on the IIC bus. There is no way to verify data in the buffer without disturbing the IIC transaction. The act of verification requires removing the data. If the data is removed, invalid data is placed on the IIC bus.
5. Use care when monitoring the IICx_XCNTLSS[SDBD] or to IICx_STS[MDBS] to determine when data is present. These bits are set to 1 when the buffer contains data in any stage. Consider the case where the master buffer is empty prior to being loaded with a byte received over the IIC bus. The byte enters the fourth stage of the buffer and the IICx_STS[MDBS] is set to 1. Stages 1, 2, and 3 do not contain data. Therefore, the data is not available for four OPB clock cycles. Any attempt to prematurely read the data yields invalid data.
6. When responding to a slave needs service request, manage the data first. Read data out of the slave buffer, IICx_SDB, for slave reads or write data into the IICx_SDB for slave writes. Next clear the slave needs service request. For reads, clear IICx_XTCNTLSS[SRS]. For writes, clear IICx_XTCNTLSS[SWS]. Last, clear the active interrupt, IICx_STS[IRQA] =0.
7. There is no timeout function implemented in the IIC interface. If this type of error recovery function is needed, it must be implemented in software.
8. Avoid the situations listed in Section 7.2 of the *Phillips Semiconductors I^2C Specification,* dated 1995. For your convenience, the section is summarized as follows:

   If multiple masters can be simultaneously involved in a transfer to the same address, or device, then the design of the system must be done in such a way that arbitration between:

   - A repeated Start condition and a data bit does not occur.
   - A Stop condition and a data bit does not occur.
   - A repeated Start condition and a Stop condition does not occur.

An example of this situation occurs if one master writes 1 byte while another master writes 2 bytes to the same device.

29. GPIO Operations

This chapter describes the General Purpose I/O (GPIO) controllers attached to the on-chip peripheral bus (OPB) of the PPC440EP. The GPIO controllers provide flexible control of multiplexed I/Os selectable under program control. Each of the I/Os is multiplexed with other signals to reduce the quantity of I/O pins needed on the PPC440EP package.

29.1 Overview

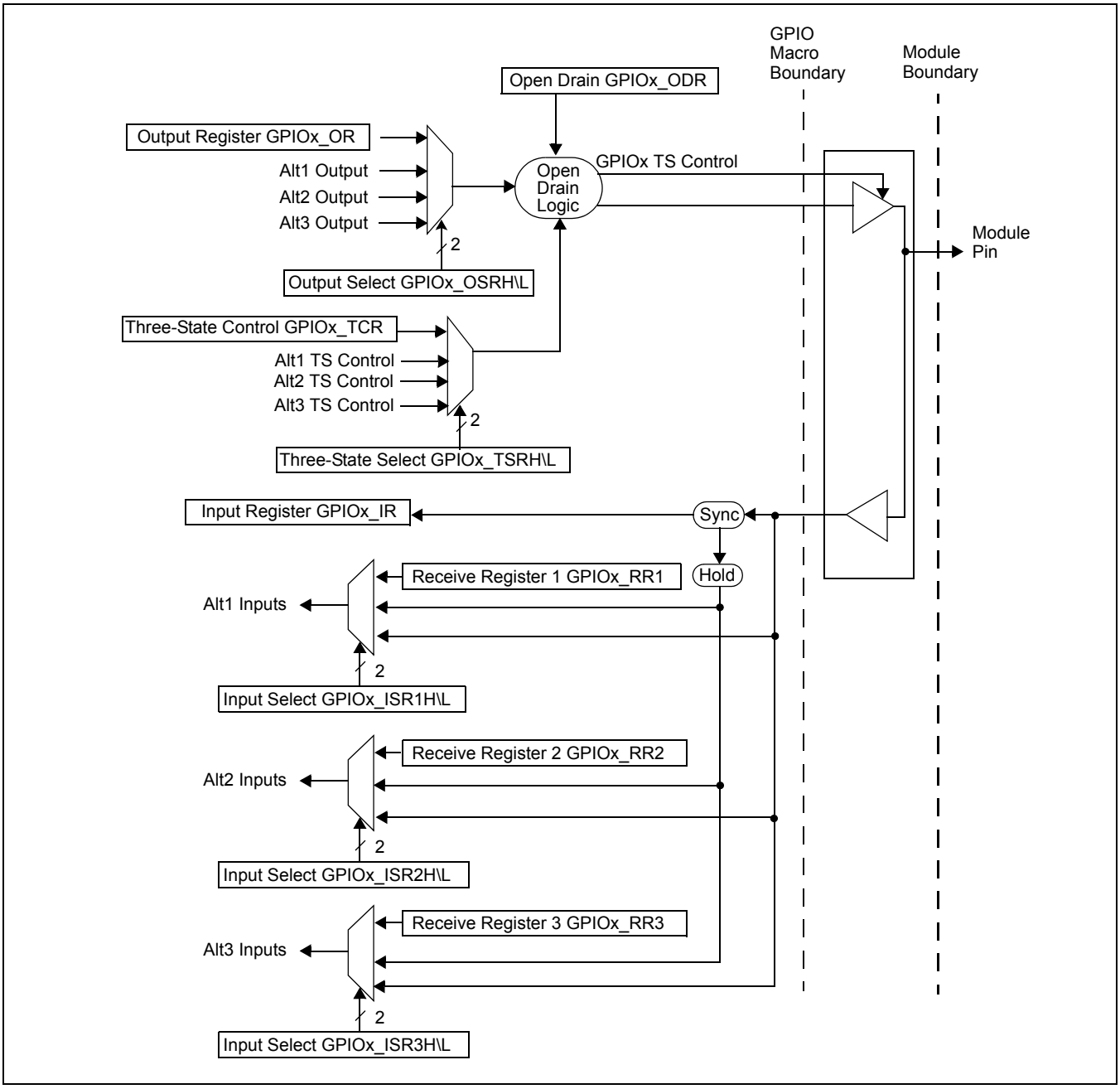
The PPC440EP has two 32-bit GPIO controllers: GPIO0 and GPIO1. GPIO0 provides 32 user-programmable external signals, multiplexed with system-related signal groups including trace outputs, external interrupt inputs, chip selects, DMA, EBC, ZMII and USB2D interface signals. GPIO1 provides another 32 programmable signals, multiplexed with USB2D, UART0, UART1, UART2, DMA and UIC interfaces. When information applies to both GPIO controllers in the PPC440EP, references are made generic to avoid repetition: GPIOx, not GPIO0 and GPIO1.

I/O signals maintain a bit-for-bit correspondence with both Input and Output Register bits. For example, a signal output to pin GPIO1_Out[6] can be sourced by GPIO1 Output Register bit 6 or by either of two alternative output signals, UART0_DTR and UART1_Tx.

Signals sourced to all GPIO outputs are selected by the Output Select Register (GPIOx_OSR), while output buffer operation (three-state output, open-drain output) is controlled by the Open Drain Register (GPIOx_ODR) and the Three-State Control Register (GPIOx_TCR).

Figure 29-1 shows data flow between one module pin and the alternative sources and destinations to which the module pin is connected. For example, a signal input on a GPIO pin is captured in the corresponding bit of the GPIO Input Register, as well as multiplexed to one or two other internal functional connections. The multiplexed signals are generically referred to as Alt1, Alt2 or Alt3 inputs and outputs.

Figure 29-1. GPIO Dataflow and Configuration Registers



29.2 Features

Each of the GPIO Controllers has the following features:

- Direct control of all GPIO controller functions from registers programmed via memory-mapped addresses
  - Each output can be selected from one of four sources.
  - Each output three-state control can be selected from one of four sources.
  - Selection of the input source for the Alternate 1 input
  - Selection of the input source for the Alternate 2 input
  - Selection of the input source for the Alternate 3 input
  - Settable register as one of the possible inputs for the Alternate 1 input
  - Settable register as one of the possible inputs for the Alternate 2 input
  - Settable register as one of the possible inputs for the Alternate 3 input
- Control of 32 bidirectional GPIO module pins
  - Each GPIO output can be set from the corresponding Output Register bit.
  - Each GPIO output has programmable three-state control.
  - Each GPIO output can also be programmed to emulate an open drain output.
  - Each GPIO input is observable from the corresponding Input Register bit.

29.3 Clock and Power Management

The GPIO controllers support clock and power management. Unconditional Sleep (Class 1) power management is implemented for each controller, and is enabled by setting the corresponding CPM0_ER[GPIOx] bit, as shown in *CPM Enable Register (CPM0_ER)* on page 301.

29.4 GPIO Registers

When an I/O is used as GPIO it is controlled by the corresponding bit in the Output Register GPIOx_OR, the Three-State Control Register GPIOx_TCR, the Open Drain Register GPIOx_ODR, and the Input Register GPIOx_IR. Whether an I/O is used as a GPIO or a functional output is selected using the Output Select Register pair GPIOx_OSRH and GPIOx_OSRL. The source of the three-state control is selected using the Three-State Select Register pair GPIOx_TSRH and GPIOx_TSRL. When an alternate input is used, the source of the alternate input is selected using the Input Select Register GPIOx_ISRnL and GPIOx_ISRnH. Note that there are two input select register pairs; one pair for Alternate 1 functions and one pair for Alternate 2 functions. The controller also contains Receive Registers GPIOx_RRx. A detailed description of the function of each of the registers is contained in *Detailed Register Descriptions* on page 683.

Table 29-1 contains a summary of the GPIO registers.

Table 29-1. GPIO Register Summary

Mnemonic	Address	Access	Description	Page
GPIO0_OR	0x0 EF60 0B00	R/W	GPIO0 Output	683
GPIO0_TCR	0x0 EF60 0B04	R/W	GPIO0 Three-State Control Register	683
GPIO0_OSRL	0x0 EF60 0B08	R/W	GPIO0 Output Select Register Low	683

Table 29-1. GPIO Register Summary (continued)

Mnemonic	Address	Access	Description	Page
GPIO0_OSRH	0x0 EF60 0B0C	R/W	GPIO0 Output Select Register High	683
GPIO0_TSRL	0x0 EF60 0B10	R/W	GPIO0 Three-State Select Register Low	684
GPIO0_TSRH	0x0 EF60 0B14	R/W	GPIO0 Three-State Select Register High	684
GPIO0_ODR	0x0 EF60 0B18	R/W	GPIO0 Open Drain Register	684
GPIO0_IR	0x0 EF60 0B1C	R	GPIO0 Input Register	685
GPIO0_RR1	0x0 EF60 0B20	R/W	GPIO0 Receive Register 1	686
GPIO0_RR2	0x0 EF60 0B24	R/W	GPIO0 Receive Register 2	
GPIO0_RR3	0x0 EF60 0B28	R/W	GPIO0 Receive Register 3	686
GPIO0_ISR1L	0x0 EF60 0B30	R/W	GPIO0 Input Select Register 1 Low	685
GPIO0_ISR1H	0x0 EF60 0B34	R/W	GPIO0 Input Select Register 1 High	685
GPIO0_ISR2L	0x0 EF60 0B38	R/W	GPIO0 Input Select Register 2 Low	685
GPIO0_ISR2H	0x0 EF60 0B3C	R/W	GPIO0 Input Select Register 2 High	685
GPIO0_ISR3L	0x0 EF60 0B40	R/W	GPIO0 Input Select Register 3 Low	685
GPIO0_ISR3H	0x0 EF60 0B44	R/W	GPIO0 Input Select Register 3 High	685
GPIO1_OR	0x0 EF60 0C00	R/W	GPIO1 Output Register	683
GPIO1_TCR	0x0 EF60 0C04	R/W	GPIO1 Three-State Control Register	683
GPIO1_OSRL	0x0 EF60 0C08	R/W	GPIO1 Output Select Register Low	683
GPIO1_OSRH	0x0 EF60 0C0C	R/W	GPIO1 Output Select Register High	683
GPIO1_TSRL	0x0 EF60 0C10	R/W	GPIO1 Three-State Select Register Low	684
GPIO1_TSRH	0x0 EF60 0C14	R/W	GPIO1 Three-State Select Register High	684
GPIO1_ODR	0x0 EF60 0C18	R/W	GPIO1 Open Drain Register	684
GPIO1_IR	0x0 EF60 0C1C	R	GPIO1 Input Register	685
GPIO1_RR1	0x0 EF60 0C20	R/W	GPIO1 Receive Register 1	686
GPIO1_RR2	0x0 EF60 0C24	R/W	GPIO1 Receive Register 2	
GPIO1_RR3	0x0 EF60 0C28	R/W	GPIO1 Receive Register 3	686
GPIO1_ISR1L	0x0 EF60 0C30	R/W	GPIO1 Input Select Register 1 Low	685
GPIO1_ISR1H	0x0 EF60 0C34	R/W	GPIO1 Input Select Register 1 High	685
GPIO1_ISR2L	0x0 EF60 0C38	R/W	GPIO1 Input Select Register 2 Low	685
GPIO1_ISR2H	0x0 EF60 0C3C	R/W	GPIO1 Input Select Register 2 High	685
GPIO1_ISR3L	0x0 EF60 0C40	R/W	GPIO1 Input Select Register 3 Low	685
GPIO1_ISR3H	0x0 EF60 0C44	R/W	GPIO1 Input Select Register 3 High	685

Notes: 1. All GPIO registers are memory-mapped and accessed using load/store instructions at the register address. All GPIO registers are aligned on word boundaries. The input and output select registers are also aligned on double word boundaries.
2. Low register set controls GPIOx I/O signals 0 to 15. High register set controls GPIOx I/O signals 16 to 31.

29.5 GPIO Register Reset Values

When a system reset occurs, all register bits in the GPIO controllers, except GPIOx_IR, are reset to 0. All outputs are placed in high impedance. GPIOx_IR is not reset because it is synchronized to the OPB clock and always tracks the state of the I/O pins.

29.6 Detailed Register Descriptions

The following sections provide detailed descriptions of the GPIO controller registers. The two GPIO controllers are attached to the OPB bus. The GPIOx_IR register is read-only; all other registers are both read- and write-accessible.

29.6.1 GPIO Output Register (GPIOx_OR)

When a bit in one of the two GPIO controllers is used as a GPIO and specifically as an output, the state of the output is controlled by the value in the GPIOx Output Register, GPIOx_OR. Whether the setting of the bit in the GPIOx_OR is visible on the I/O pin is a function of the settings in the GPIOx_TCR, GPIOx_ODR, GPIOx_OSRH, GPIOx_OSRL, GPIOx_TSRH, and GPIOx_TSRL registers.

Figure 29-2. GPIO Output Register (GPIOx_OR)

0:31		GPIO0_OR register bits	

29.6.2 GPIO Three-State Control Register (GPIOx_TCR)

The GPIOx_TCR register is one source for three-stating a corresponding output. For each bit in the GPIOx_TCR Register to control the corresponding output, the appropriate two bits in the GPIO Three-State Select Register High and Low (GPIOx_TSRH, GPIOx_TSRL) must be programmed to 0b00. If the GPIOx_TCR register is selected; setting a bit to 1 in GPIOx_TCR enables the associated output driver. Clearing the bit to 0 forces the corresponding output into high impedance state. When the same bit is also set in GPIOx_ODR, the output emulates an open-drain output, regardless of the bit setting in GPIOx_TCR.

Figure 29-3. GPIO Three-State Register (GPIOx_TCR)

0:31		GPIO0_TCR register bits	

29.6.3 GPIO Output Select Registers (GPIOx_OSRH, GPIOx_OSRL)

The GPIOx_OSR register pair (GPIOx_OSRH, GPIOx_OSRL) determines what signal source is sent to the output pin. For each output there can be up to four sources of output value. Two bits in the GPIOx_OSRH/L register pair are needed to select each output. This requires a total of 64 bits to control the 32 output bits in one GPIO controller. The 64 bits are made up of two 32 bit registers. *Table 29-2* shows how these two bits control the selection of the signal connected to that GPIO Out pin.

Note: The correspondence between GPIOx_OSRH/L and pins 0:63 is summarized in *Section 29.7* on page 687.

Table 29-2. GPIO Output Signal Selection

GPIOx_OSRH/L Bits	GPIO_Out Signal Source
00	GPIOx_OR
01	Alt1 output source
10	Alt2 output source
11	Alt3 output source

29.6.4 GPIO Three-State Select Registers (GPIOx_TSRH, GPIOx_TSRL)

The GPIOx_TSR register pair (GPIOx_TSRH, GPIOx_TSRL) determines what signal source is used for the three-state control input for the GPIO output signals.

For each output there can be two sources of three-state control. Two bits in the GPIOx_TSRH/L register pair are needed for each output. This requires a total of 64 bits to control the 32 output bits in one GPIO controller. The 64 bits are made up of two 32 bit registers.

The three-state selection for a pin requires the same combination as the output select register selection. That is, if the Alt2 is selected then the Alt2 three-state control will be selected. *Table 29-3* shows how these two bits control the selection of the signal connected to that GPIO_Out pin.

Note: The correspondence between GPIOx_TSRH/L and pins 0:63 is summarized in *Section 29.7* on page 687.

Table 29-3. GPIO Three-State Selection

GPIOx_TSRH/L Bits	GPIO Three-State Control Source
00	GPIOx_TCR
01	Alt1 three-state source
10	Alt2 three-state source
11	Alt3 three-state source

29.6.5 GPIO Open Drain Register (GPIOx_ODR)

The GPIOx_ODR configures the module I/O three-state driver to emulate open drain drivers on a bit-by-bit basis. *Table 29-4* shows the function of the GPIOx_ODR register and its interaction with the three-state control signal and output signal. While the GPIOx_ODR register can control Alternate 1, 2 and 3 outputs, as

well as alternate three-state control signals, this feature is not used in the PPC440EP. For this reason the registers listed in the table are used when a bit is used as a GPIO.

Table 29-4. GPIO0_ODR Control Settings

GPIOx_ODR bit	GPIOx_TCR bit	GPIOx_OR bit	State of Module pin
0	0	x	Forced to high impedance
0	1	0	Driving 0
0	1	1	Driving 1
1	x	0	Driving 0
1	x	1	Forced to high impedance

Figure 29-4. GPIO Open Drain Register (GPIOx_ODR)

0:31		GPIO0_ODR register bits	

29.6.6 GPIO Input Register (GPIOx_IR)

The state of each bit in the GPIOx_IR Register reflects the corresponding GPIO controller input signal. All input signals are synchronized to OPBClk before being stored in the GPIOx_IR Register. The GPIOx_IR register is read-only and does not change during a read access. To receive valid data as input from a module pin, the three-state output attached to that module pin is first placed in high impedance by setting the corresponding three-state control bit in the GPIOx_TCR register.

Figure 29-5. GPIO Input Register (GPIOx_IR)

0:31		GPIO register bits	

29.6.7 GPIO Input Select Registers (GPIOx_ISRnH, GPIOx_ISRnL)

The GPIOx_ISR1H/L, GPIOx_ISR2H/L and GPIOx_ISR3H/L registers determine what signal source is used as the input signals. For each Altn input there can be up to three sources of input value. Two bits in the GPIOx_OSR1H/L register pair are needed for each Alt1 input, two bits in the GPIOx_OSR2H/L register pair are needed for each Alt2 input, and two bits in the GPIOx_OSR3H/L register pair are needed for each Alt3 input. This requires a total of 64 bits to control the 32 Alt1 input bits, 64 bits to control the 32 Alt2 input bits in one GPIO controller, and 64 bits to control the 32 Alt3 input bits. The 64 bits are made up of two 32 bit registers. *Figure 29-5* shows how these two bits control the selection of the input. For normal operation the only setting used is 01 which selects the pin input.

Note: The correspondence between GPIOx_ISRnH/L and pins 0:63 is summarized in *Section 29.7* on page 687.

Table 29-5. GPIO Alternate Input Signal Selection

GPIOx_ISRnH/L Bits	Altn Input Signal
00	GPIO Receive Register (GPIOx_RRn)
01	Input from GPIO pin (not synchronized) - configuration used by Altn signals
10	Input from GPIO pin (synchronized and latched)
11	Reserved

29.6.8 GPIO Receive Registers (GPIOx_RR1, GPIOx_RR3)

For normal operation the receive registers are not used, as noted in later sections on selecting the Alt1, Alt2 and Alt3 signals. They may however be useful during software debug. There are times when a programmer would like to control input values without having to determine how to cause the system to produce the desired value at an input. By programming the desired input bit in GPIOx_RR1 for Alt1 inputs, GPIOx_RR2 for Alt2 inputs, GPIOx_RR3 for Alt3 inputs and then setting the appropriate two bits, in the associated GPIOx_ISRnH/L register pair to 00, a desired input value can be set. Note that these registers only affect the alternate inputs, not the GPIOx_IR register.

29.6.9 Control of GPIO Signals 49–63

GPIO49:63 signals are multiplexed with CPU trace signals. In order to use these signals in GPIO mode, the SDR0_PFC0 register must be properly initialized (*Section 8.4.2.5* on page 197). There are no alternates inputs or outputs for these pins. They are controlled by the GPIO1_xxxH registers.

29.7 GPIO Signal Multiplexing

The following tables describe the multiplexed signal assignments for the GPIO0 and GPIO1controllers. All of the GPIO signals are multiplexed with one or more additional signals. Each of the package pins on which these signals appear has one of the indicated signal (the default or primary signal) connected following a reset of the chip. The GPIO signals are not the primary signal for any pin. The primary signal is the signal in the Alternate 1 column.

Table 29-6. Alternate 1 Configuration for GPIO00–GPIO15

GPIO Signal	Alternate 1 Signal	I/O	GPIO0_TCR		GPIO0_OSRL		GPIO0_TSRL		GPIO0_ISR1L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO00	PerAddr07[1]	I/O	0	x	0:1	01	0:1	01	0:1	01
GPIO01	PerAddr06[1]	I/O	1	x	2:3	01	2:3	01	2:3	01
GPIO02	PerAddr05[1]	I/O	2	x	4:5	01	4:5	01	4:5	01
GPIO03	PerAddr04[1]	I/O	3	x	6:7	01	6:7	01	6:7	01
GPIO04	PerAddr03[1]	I/O	4	x	8:9	01	8:9	01	8:9	01
GPIO05	PerAddr02[1]	I/O	5	x	10:11	01	10:11	01	10:11	01
GPIO06	PerCS1, $\overline{\text{NFCE1}}$[2]	O	6	x	12:13	01	12:13	01	12:13	xx
GPIO07	PerCS2, $\overline{\text{NFCE2}}$[2]	O	7	x	14:15	01	14:15	01	14:15	xx
GPIO08	PerCS3, $\overline{\text{NFCE3}}$[2]	O	8	x	16:17	01	16:17	01	16:17	xx
GPIO09	PerCS4	O	9	x	18:19	01	18:19	01	18:19	xx
GPIO10	PerCS5	O	10	x	20:21	01	20:21	01	20:21	xx
GPIO11	PerErr	I	11	x	22:23	xx	22:23	01	22:23	01
GPIO12	EMCRxD0, EMC0RxD0, EMC0RxD	I	12	0	24:25	xx	24:25	01	24:25	01
GPIO13	EMCRxD1, EMC0RxD1, EMC1RxD	I	13	0	26:27	xx	26:27	00	26:27	01
GPIO14	EMCRxD2, EMC1RxD0	I	14	0	28:29	xx	28:29	00	28:29	01
GPIO15	EMCRxD3, EMC1RxD1	I	15	0	30:31	xx	30:31	00	30:31	01

Notes: 1. PerAddr02:07 can be configured as an output when not using the EBM.
2. SDR0_CUST0[NCG] controls the Alternate 1 signal multiplexing of the PerCS1:3/NFCE1:3 signals.

Table 29-7. Alternate 1 Configuration for GPIO16–GPIO31

GPIO Signal	Alternate 1 Signal	I/O	GPIO0_TCR		GPIO0_OSRH		GPIO0_TSRH		GPIO0_ISR1H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO16	EMCTxD0, EMC0TxD0, EMC0TxD	O	16	1	0:1	01	0:1	00	0:1	xx
GPIO17	EMCTxD1, EMC0TxD1, EMC1TxD	O	17	1	2:3	01	2:3	00	2:3	xx
GPIO20	EMCRxErr, EMC0RxErr	I	20	0	8:9	xx	8:9	00	8:9	01
GPIO22	EMCCrS EMC0CrsDV	I	22	0	12:13	xx	12:13	00	12:13	01
GPIO24	EMCTxEn, EMCSync	O	24	1	16:17	01	16:17	00	16:17	xx
GPIO26	See *Table 29-11*.									
GPIO27	ExtReq	I	27	0	22:23	xx	22:23	00	22:23	01
GPIO28	See *Table 29-11*.									
GPIO29	HoldAck	O	29	x	26:27	01	26:27	01	26:27	xx
GPIO30	ExtAck	O	30	x	28:29	01	28:29	01	28:29	xx
GPIO31	BusReq	O	31	x	30:31	01	30:31	01	30:31	xx

Table 29-8. Alternate 1 Configuration for GPIO32–GPIO47

GPIO Signal	Alternate 1 Signal	I/O	GPIO1_TCR		GPIO1_OSRL		GPIO1_TSRL		GPIO1_ISR1L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO32	USB2OM0	O	0	1	0:1	01	0:1	00	0:1	xx
GPIO33	USB2OM1	O	1	1	2:3	01	2:3	00	2:3	xx
GPIO34	$\overline{\text{UART0\_DCD}}$	I	2	0	4:5	xx	4:5	00	4:5	01
GPIO35	$\overline{\text{UART0\_DSR}}$	I	3	0	6:7	xx	6:7	00	6:7	01
GPIO36	$\overline{\text{UART0\_CTS}}$, $\overline{\text{UART0\_DSR}}$[1]	I	4	0	8:9	xx	8:9	00	8:9	01
GPIO37	$\overline{\text{UART0\_RTS}}$, $\overline{\text{UART0\_DTR}}$[1]	O	5	1	10:11	01	10:11	00	10:11	xx
GPIO38	$\overline{\text{UART0\_DTR}}$	O	6	1	12:13	01	12:13	00	12:13	xx
GPIO39	$\overline{\text{UART0\_RI}}$	I	7	0	14:15	xx	14:15	00	14:15	01
GPIO40	IRQ0	I	8	0	16:17	xx	16:17	00	16:17	01
GPIO41	IRQ1	I	9	0	18:19	xx	18:19	00	18:19	01
GPIO42	IRQ2	I	10	0	20:21	xx	20:21	00	20:21	01
GPIO43	IRQ3	I	11	0	22:23	xx	22:23	00	22:23	01
GPIO44	IRQ4	I	12	0	24:25	xx	24:25	00	24:25	01
GPIO45	IRQ6	I	13	0	26:27	xx	26:27	00	26:27	01
GPIO46	IRQ7	I	14	0	28:29	xx	28:29	00	28:29	01
GPIO47	IRQ8	I	15	0	30:31	xx	30:31	00	30:31	01

Notes: 1. SDR0_PFC1[U0ME] specifies for the UART0 in 4 wire mode (SDR0_PFC1[U0IM] = 1) which control signals are used: DSR/DTR or CTS/RTS.

Table 29-9. Alternate 1 Configuration for GPIO48

GPIO Signal	Alternate 1 Signal	I/O	GPIO1_TCR		GPIO1_OSRH		GPIO1_TSRH		GPIO1_ISR1H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO48	IRQ9	I	16	0	0:1	xx	0:1	00	0:1	01

Table 29-10. Alternate 2 Configuration for GPIO00–GPIO05

GPIO Signal	Alternate 2 Signal	I/O	GPIO0_TCR		GPIO0_OSRL		GPIO0_TSRL		GPIO0_ISR2L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO00	DMAReq2	I	0	0	0:1	xx	0:1	00	0:1	01
GPIO01	DMAAck2	O	1	1	2:3	10	2:3	00	2:3	xx
GPIO02	EOT2/TC2	I/O	2	x	4:5	10	4:5	10	4:5	01
GPIO03	DMAReq3	I	3	0	6:7	xx	6:7	00	6:7	01
GPIO04	DMAAck3	O	4	1	8:9	10	8:9	00	8:9	xx
GPIO05	EOT3/TC3	I/O	5	x	10:11	10	10:11	10	10:11	01

Table 29-11. Alternate 2 Configuration for GPIO26–GPIO31

GPIO Signal	Alternate 2 Signal	I/O	GPIO0_TCR		GPIO0_OSRH		GPIO0_TSRH		GPIO0_ISR2H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO26	USB2RxDV	I	26	0	20:21	xx	20:21	00	20:21	01
GPIO27	USB2RxErr	I	27	0	22:23	xx	22:23	00	22:23	01
GPIO28	USB2TxVal	O	28	1	24:25	10	24:25	00	24:25	xx
GPIO29	USB2Susp	O	29	1	26:27	10	26:27	00	26:27	xx
GPIO30	USB2XcvrSel	O	30	1	28:29	10	28:29	00	28:29	xx
GPIO31	USB2TermSel	O	31	1	30:31	10	30:31	00	30:31	xx

Table 29-12. Alternate 2 Configuration for GPIO34–GPIO47

GPIO Signal	Alternate 2 Signal	I/O	GPIO1_TCR		GPIO1_OSRL		GPIO1_TSRL		GPIO1_ISR2L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO34	UART1_CTS, UART1_DSR[1]	I	2	0	4:5	xx	4:5	00	4:5	01
GPIO35	UART1_RTS, UART1_DTR[1]	O	3	1	6:7	10	6:7	00	6:7	xx
GPIO38	UART1_Tx	O	6	1	10:11	10	10:11	00	10:11	xx
GPIO39	UART1_Rx	I	7	0	12:13	xx	12:13	00	12:13	01
GPIO40-GPIO43	Alternate 1 signal only. See *Table 29-9*.									
GPIO44	DMAAck1	O	12	1	24:25	10	24:25	00	24:25	xx
GPIO45	EOT1/TC1	I/O	13	x	26:27	10	26:27	10	26:27	01
GPIO46	DMAReq0	I	14	0	28:29	xx	28:29	00	28:29	01
GPIO47	DMAAck0	O	15	1	30:31	10	30:31	00	30:31	xx

Notes: 1. SDR0_PFC1[U1ME] specifies for UART1 in 4 wire mode (SDR0_PFC1[U0IM]=1) which control signals are used: DSR/DTR or CTS/RTS..

Table 29-13. Alternate 2 Configuration for GPIO48

GPIO Signal	Alternate 2 Signal	I/O	GPIO1_TCR		GPIO1_OSRH		GPIO1_TSRH		GPIO1_ISR2H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO48	EOT0/TC0	I/O	16	x	0:1	10	0:1	10	0:1	01

Table 29-14. Alternate 3 Configuration for GPIO34–GPIO37

GPIO Signal	Alternate 3 Signal	I/O	GPIO1_TCR		GPIO1_OSRL		GPIO1_TSRL		GPIO1_ISR3L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO34	UART2_Tx	O	2	1	4:5	11	4:5	00	4:5	xx
GPIO35	UART2_Rx	I	3	0	6:7	xx	6:7	00	6:7	01
GPIO36	UART3_Rx	I	4	0	8:9	xx	8:9	00	8:9	01
GPIO37	UART3_Tx	O	5	1	10:11	11	10:11	00	10:11	xx

30. Universal Serial Bus Interfaces

The following sections provide and overview of the Universal Serial Bus (USB) interfaces implemented in the PPC440EP chip. Detailed information on how to program the interfaces is provided in supplemental documents covering each of the available interfaces.

30.1 Attributions

The information related to USB 1.1 Host support describes the operation and programming of the IBM® USB 1.1 Host Interface core instantiated in the PPC440EP embedded processor chip.

This information is supplied by and derived from IBM sources, and is being used under rights granted to AMCC® by IBM. All other rights and permissions in connection with the use of this information are reserved by IBM.

The information related to USB 2.0 Device support describes the operation and programming of the Mentor Graphics Corporation® USB 2.0 Device Controller core instantiated in the PPC440EP embedded processor chip.

This PPC440EP Supplemental document contains trade secret and confidential information of Mentor Graphics Corporation or its licensors; your access and use thereof is subject to the obligation of confidentiality stated herein.

This information can only be used by those customers who have negotiated and signed a non-disclosure agreement (NDA) with AMCC®.

This information is being used under rights granted to AMCC by the Mentor Graphics Corporation. All other rights and permissions in connection with the use of this information are reserved by the Mentor Graphics Corporation.

Mentor Graphics and Inventra are trademarks of the Mentor Graphics Corporation.

30.2 USB Overview

The USB is a serial communications interface specification designed to provide a common solution for interconnecting various personal computer peripherals with a single host computing device in a star topology. The USB architecture consists of a half duplexed, packetized, single mastered, tiered tree structure, supporting up to 127 unique devices on the tree. The master device on the USB tree is known as the USB Host Controller.

The USB Host Controller interacts with the USB Host Controller Driver software to conduct transactions on the USB wire. These transactions occur through the Host Controller's root hub. A root hub is the portion of the Host Controller which interacts with the USB cable and may contain a number of ports. Each port may connect to a USB cable with which it communicates with down stream hubs and or devices. A USB hub is a hardware entity which allows one up stream port to communicate to a number of down stream ports. A USB device is an entity which resides at the far end of a down stream USB cable and provides some specific function (that is, keyboard, mouse, printer, modem, etc.).

The USB cable consists of four wires. Two of the wires are a twisted differential pair which are used for transmitting a non-return-to-zero-inverted (NRZI) encoded data stream. The other two wires are used for power and ground.

Version 1.1 of the USB specification defines two transfer rates. The Low Speed transfer rate of 1.5 Mbps and the Full Speed transfer rate of 12 Mbps. Version 2.0 of the USB specification, which is a full replacement of Version 1.1 of the specification, in addition to defining the Low Speed and Full Speed transfer rates also defines the new High Speed transfer rate of 480 Mbps.

30.2.1 PPC440EP USB Implementation

The EP USB hardware includes the following:

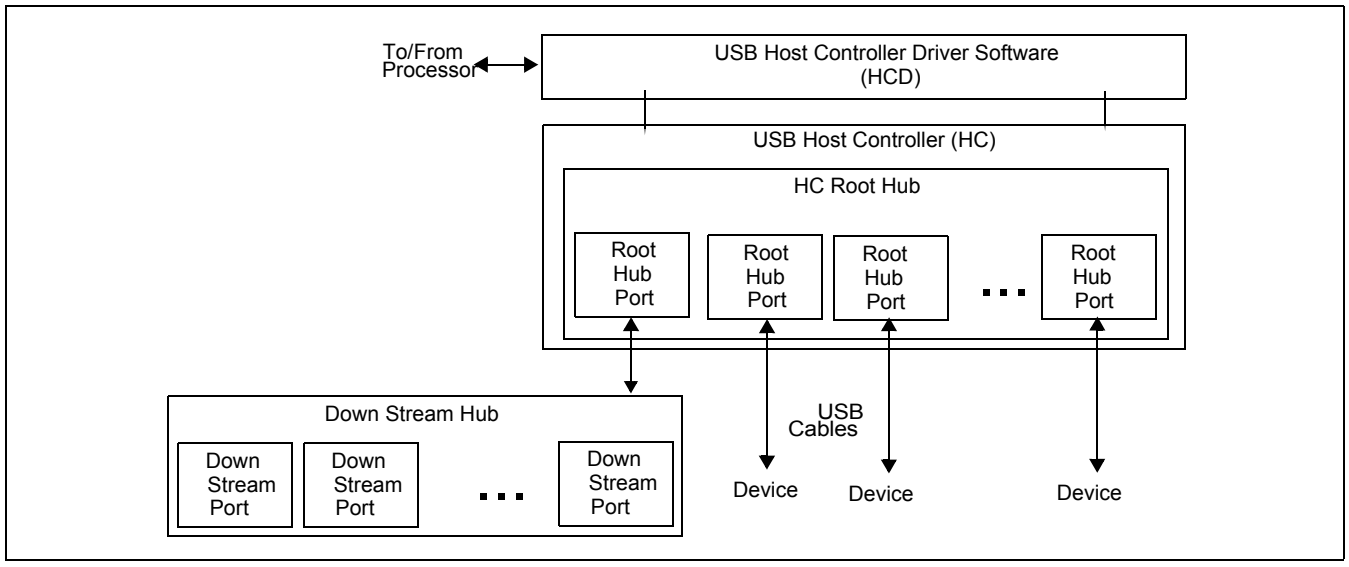
- USB1.1 Device function with transceiver
- USB1.1 Host function with a transceiver BUSB5
- USB2.0 Device function with UTMI interface or with a transceiver BUSB5.

Along with other low-data-rate peripherals, USB1.1 Host and device interfaces connect to the processor through the primary On-Chip Peripheral Bus (OPB). The PPC440EP secondary OPB (OPB1) is dedicated to USB2.0 and DMA.

The USB architecture consists of a half duplexed, packetized, single mastered, tiered tree structure, supporting up to 127 unique devices on the tree. The master device on the USB tree is known as the USB host controller.

The USB host controller (HC) interacts with the USB host controller driver (HCD) software to conduct transactions on the USB wire. These transactions occur through the host controller's root hub. A root hub is the portion of the host controller that interacts with the USB cable and can contain a number of ports. Each port can connect to a USB cable through which it communicates with down stream hubs and or devices. A USB hub is a hardware entity that allows one up-stream port to communicate with a number of down stream ports. A USB device is an entity that resides at the far end of a down stream USB cable and provides some specific function (such as, keyboard, mouse, printer, modem, etc.). The relationships among these USB entities are shown in the following figure.

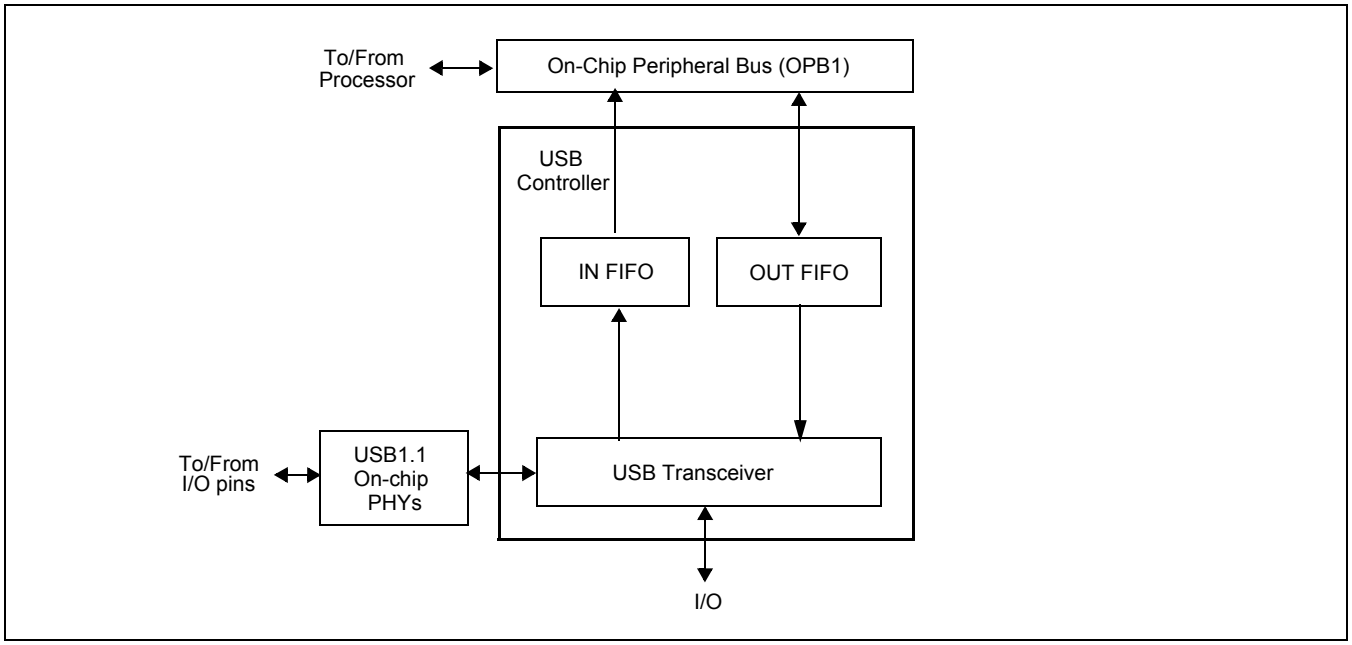
Figure 30-1. USB Interface Block Diagram



PPC440EP implements USB1.1 host and device, and USB2.0 device interfaces.

The following figure shows the USB controller data flow.

Figure 30-2. USB Controller Data Flow



30.2.2 USB Configuration Register (SDR0_USB0)

The SDR0_USB0 register described in the following figure enables the selection of either the USB2.0 or USB1.1 device interface, and the enabling or disabling of little endian mode.

Reset value = 0

Figure 30-3. USB Configuration Register (SDR0_USB0)

0:29		Reserved	
30	USB2D	USB2.0 Device Interface Selection 0 USB2.0 device interface enabled 1 USB1.1 device interface enabled	
31	LE	Little Endian 0 Little endian disabled 1 Little endian enabled	

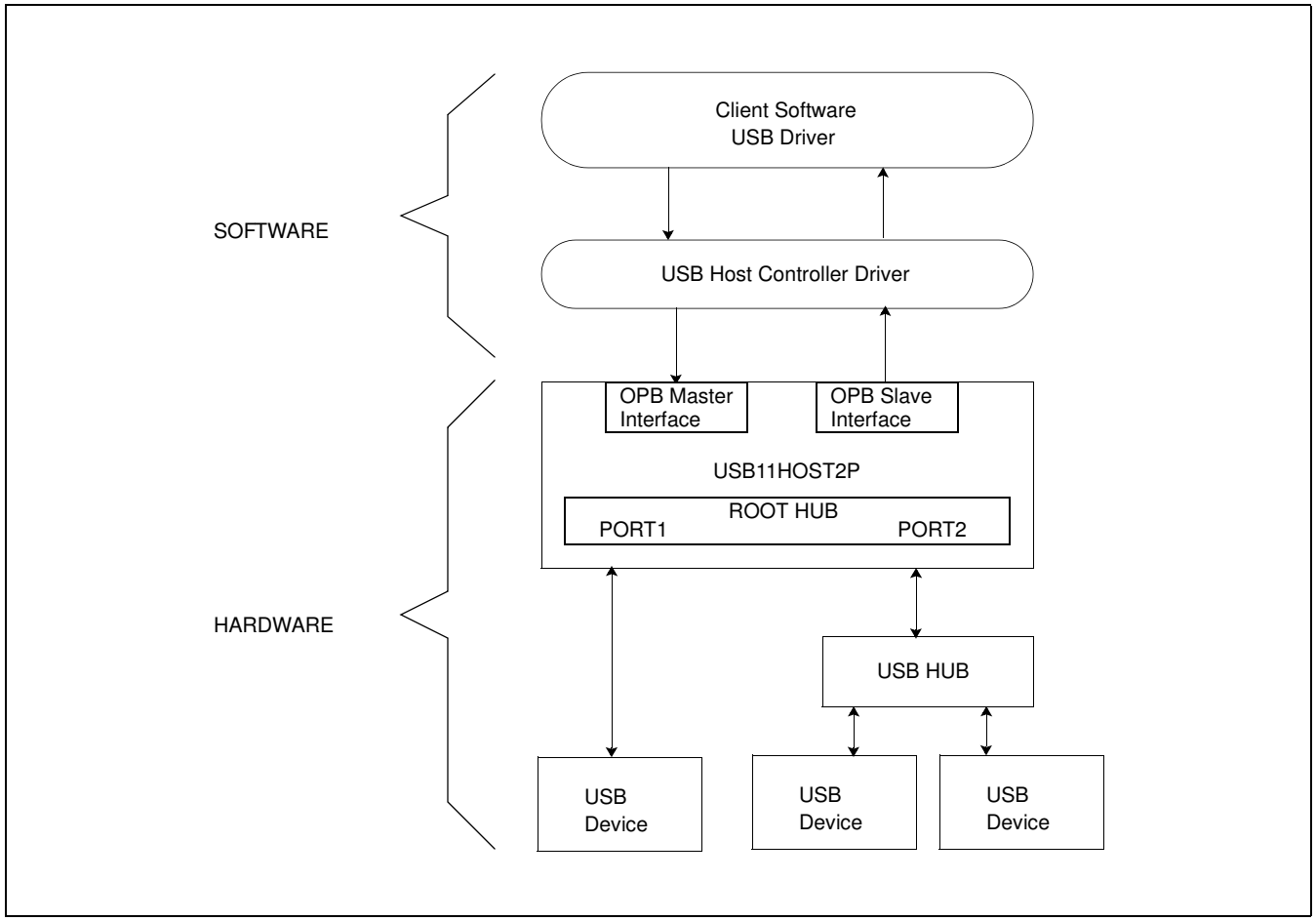
30.3 USB 1.1 Host Interface

The USB 1.1 Host interface implements the Universal Serial Bus (USB) Version 1.1 Open Host Controller Interface (OHCI) Version 1.0a function. The interface is a 2-port Host Controller that provides the interface between the USB wire and the USB 1.1 Host Controller Driver (HCD) software. The HCD issues commands to the USB 1.1 Host support which executes these commands and reports status back to the HCD. HCD commands include transferring data from the USB host to the USB device, as well as transferring data from the USB device to the USB host. When transferring data from the host to device, the Host Controller reads from an allocated area in the host system memory, packetizes, adds cyclic redundancy check information, serializes and encodes the data for transmission over the USB wire to the USB device. The Host Controller then reports back to the HCD the status of the transfer. When transferring data from the device to the host, the HCD allocates an area of system memory that

the data will be transferred into, and then signals the Host Controller hardware to attempt to receive data from a particular USB device. If the device has data to send, it transfers this data over the USB wire and the Host Controller decodes, deserializes, cyclic redundancy checks (CRC) the data, transfers this data to the appropriate system memory buffer, and reports the status of the transfer.

The USB 1.1 Host Controller interfaces to an OPB system bus architecture by means of separate 32-bit On Chip Peripheral Buses (OPB), one master bus and one slave bus. The core contains two FIFOs. One FIFO is used for transmit data and the other is used for receive data. Each FIFO is organized as 16 entries of 32-bit words. The function interfaces to the USB environment by means of an integrated root hub and two USB ports. The function supports both Full Speed (12 Mbps) and Low Speed (1.5 Mbps) transactions and up to 127 connected devices. This function does not contain support for legacy keyboard and mouse.

Figure 30-4. USB Host Controller Hardware/Software Interface



30.3.1 Features

- Compliant to USB 1.1 Specification and OHCI Version 1.0a Host Controller Specification.
- Supports Full Speed (12 Mbps) and Low Speed (1.5 Mbps) devices.
- Supports all transfer types (Isochronous, Interrupt, Control, and Bulk).
- USB11HOST2P contains an integrated root hub with two ports.

- Independent 32-bit OPB Master and 32-bit OPB slave interfaces. The Master OPB interface and Slave OPB interface can run asynchronously to each other.
- Programmable OPB slave base address.
- Does not support legacy I/O controlled keyboard and mouse devices.

30.3.2 References

This information is to be used as a supplement to the following documents. Refer to these documents when designing a system that provides USB 1.1 Host support:

- *Universal Serial Bus Specification Revision 1.1*, Compaq®, Intel®, Microsoft®, NEC, September 23, 1998
- *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a*, Compaq, Microsoft, National Semiconductor, March 17, 1999
- *On-Chip Peripheral Bus Architecture Specification Version 1.5*, IBM, July 1999

30.3.3 USB OHCI Registers

The host controller driver (HCD) controls the USB 1.1Host Controller through a set of non-cacheable memory-mapped registers which are defined by the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* and which are contained within the core. Their location in the memory map is controlled by the OPBSLBASE00..22 bus. Thus the memory decode space for these registers is 512B. Any write to a non-populated location within this 512B region will be acknowledged without error but will have no side effect. A read from a non-populated location within the 512B region will return a value of 0.

All of the registers are required to be read and written as full words. Failure to do so will cause the OPB slave interface to assert an error response to any request which is smaller than a full 32-bit full word. Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for further details on accessing these register.

Table 30-2 describes the OHCI registers and their reset values. HCD access refers to Host Controller Driver access which is done by means of the the OPB slave interface. HC access refers to Host Controller access and is done internally in the Host support function. These registers are categorized into six groups: Control and Status, Memory Pointer, Frame Counter, Root Hub, and OPB Configuration registers.

The mnemonics and names used in this document are the same as those given in the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a*. However, when referring to this specification, since it is architected in little endian format and the PPC440EP support is architected in big endian format, disregard the specifications register bit numbering and use what is shown in the following register description tables. In this document, all register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Table 30-1 provides a cross-reference between the register names used in *Table 30-2* and the actual register names as they should be specified for programming the PPC440EP. The offsets specfied in Table 30-2 are offrsets from a PPC440EP OPB base address of 0x 0 EF60 1000

Table 30-1. PPC440EP Register Name to OHCI Register Name Cross-Reference

PPC440EP Register Name	OHCI Register Name
USBH0_REVID	*HcRevision*
USBH0_CR	*HcControl*
USBH0_CS	*HcCommandStatus*
USBH0_IS	*HcInterruptStatus*
USBH0_IE	*HcInterruptEnable*
USBH0_ID	*HcInterruptDisable*
USBH0_HCCA	*HcHCCA*
USBH0_PCED	*HcPeriodCurrentED*
USBH0_CHED	*HcControlHeadED*
USBH0_CCED	*HcControlCurrentED*
USBH0_BHED	*HcBulkHeadED*
USBH0_BCED	*HcBulkCurrentED*
USBH0_DH	*HcDoneHead*
USBH0_FI	*HcFmInterval*
USBH0_FR	*HcFmRemaining*
USBH0_FN	*HcFmNumber*
USBH0_PS	*HcPeriodicStart*
USBH0_LST	*HcLSThreshold*
USBH0_RHDA	*HcRhDescriptorA*
USBH0_RHDB	*HcRhDescriptorB*
USBH0_RHSR	*HcRhStatus*
USBH0_RHPS0	*HcRhPortStatus1*
USBH0_RHPS1	*HcRhPortStatus2*
USBH0_OPBMC	*OPBMC*

Table 30-2. OHCI Host Controller Operation Register Summary

OPB Base Address Offset	Register Name	HCD Access	HC Access	System Reset Value	Reference
Control Register Group					
0x000	*HcRevision*	R	R	0x00000110	*Figure 30-5* on page 700
0x004	*HcControl*	R/W	R/W - R	0x00000000	*Figure 30-6* on page 701
Status Register Group					
0x008	*HcCommandStatus*	R/W - R	R/W	0x00000000	*Figure 30-7* on page 703
0x00C	*HcInterruptStatus*	R/W	R/W	0x00000000	*Figure 30-8* on page 704
0x010	*HcInterruptEnable*	R/W	R	0x00000000	*Figure 30-9* on page 705
0x014	*HcInterruptDisable*	R/W	R	0x00000000	*Figure 30-10* on page 706
Memory Pointer Register Group					
0x018	*HcHCCA*	R/W	R	0x00000000	*Figure 30-11* on page 707
0x01C	*HcPeriodCurrentED*	R	R/W	0x00000000	*Figure 30-12* on page 708
0x020	*HcControlHeadED*	R/W	R	0x00000000	*Figure 30-13* on page 709
0x024	*HcControlCurrentED*	R/W	R/W	0x00000000	*Figure 30-14* on page 710
0x028	*HcBulkHeadED*	R/W	R	0x00000000	*Figure 30-15* on page 711
0x02C	*HcBulkCurrentED*	R/W	R/W	0x00000000	*Figure 30-16* on page 712
0x030	*HcDoneHead*	R	R/W	0x00000000	*Figure 30-17* on page 713
Frame Counter Register Group					
0x034	*HcFmInterval*	R/W	R	0x27782EDF	*Figure 30-18* on page 714
0x038	*HcFmRemaining*	R	R/W	0x00000000	*Figure 30-19* on page 715
0x03C	*HcFmNumber*	R	R/W	0x00000000	*Figure 30-20* on page 716
0x040	*HcPeriodicStart*	R/W	R	0x00000000	*Figure 30-21* on page 717
0x044	*HcLSThreshold*	R/W	R	0x00000628	*Figure 30-22* on page 718
Root Hub Register Group					
0x048	*HcRhDescriptorA*	R/W -R	R	0x10000902	*Figure 30-23* on page 719
0x04C	*HcRhDescriptorB*	R/W	R	0x00000000	*Figure 30-24* on page 720
0x050	*HcRhStatus*	R/W - R	R/W - R	0x00000000	*Figure 30-25* on page 721
0x054	*HcRhPortStatus1*	R/W	R/W	0x00000000	*Figure 30-26* on page 723
0x058	*HcRhPortStatus2*	R/W	R/W	0x00000000	*Figure 30-26* on page 723
OPB Configuration Group					
0x070	*OPBMC*	R/W	R	0x0000_0000	*Figure 30-27* on page 725

30.3.4 Control and Status Registers

The following sections define the structure of the Control and Status registers.

30.3.4.1 HcRevision Register

Address Offset: 0x000

Width: 32

Reset Value: 0x0000_0010

HCD Access: R

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-5. HcRevision Register Format

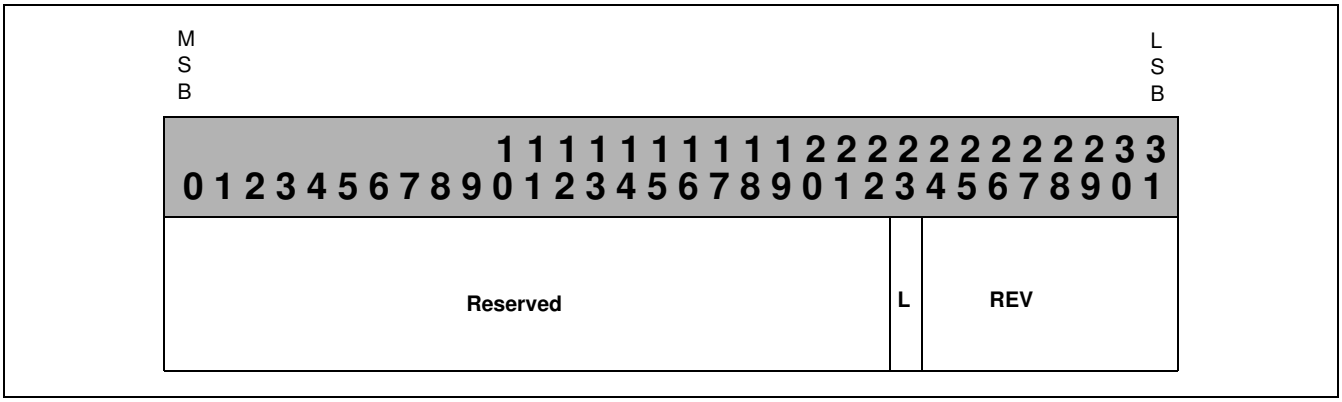


Table 30-3. HcRevision Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
L	0	R	R	N.A.	Legacy 0b: Specifies that the legacy registers are not present.
REV	0x10	R	R	N.A.	Revision 10h: Specifies the version of the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* from which the core was implemented.

30.3.4.2 HcControl Register

Address Offset: 0x004

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R/W-R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-6. HcControl Register Format

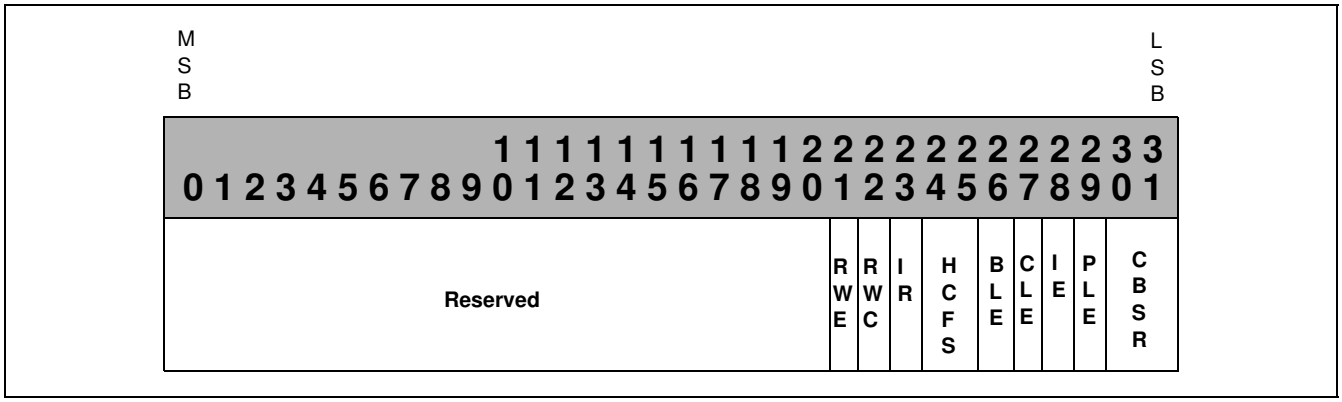


Table 30-4. HcControl Register Field Description (Sheet 1 of 2)

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
RWE	0b	R	R	HCD	RemoteWakeupEnable The USB11HOST2P core does not support remote wakeup signaling, so this bit is always read as 0.
RWC	0b	R	R	HCD	RemoteWakeupConnected The USB11HOST2P core does not support remote wakeup signaling, so this bit is always read as 0.
IR	0b	R/W	R	HCD	InterruptRouting If clear, all interrupts, except OC, are routed to the USB11HOST2P interrupt pin USBINT1. If set, interrupts are routed to the USB11HOST2P System Management Interrupt pin USBINT2.
HCFS	00b	R/W	R/W	HCD	HostControllerFunctionalState for USB
BLE	0b	R/W	R	HCD	BulkListEnable Setting this bit is guaranteed to take effect in the next frame (not the current frame).
CLE	0b	R/W	R	HCD	ControlListEnable Setting this bit is guaranteed to take effect in the next frame (not the current frame).

Table 30-4. HcControl Register Field Description (Sheet 2 of 2)

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
IE	0b	R/W	R	HCD	IsochronousEnable
PLE	0b	R/W	R	HCD	PeriodicListEnable Setting this bit is guaranteed to take effect in the next frame (not the current frame).
CBSR	00b	R/W	R	HCD	ControlBulkServiceRatio

30.3.4.3 HcCommandStatus RegisterControl

Address Offset: 0x008

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W-R

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-7. HcCommandStatus Register Format

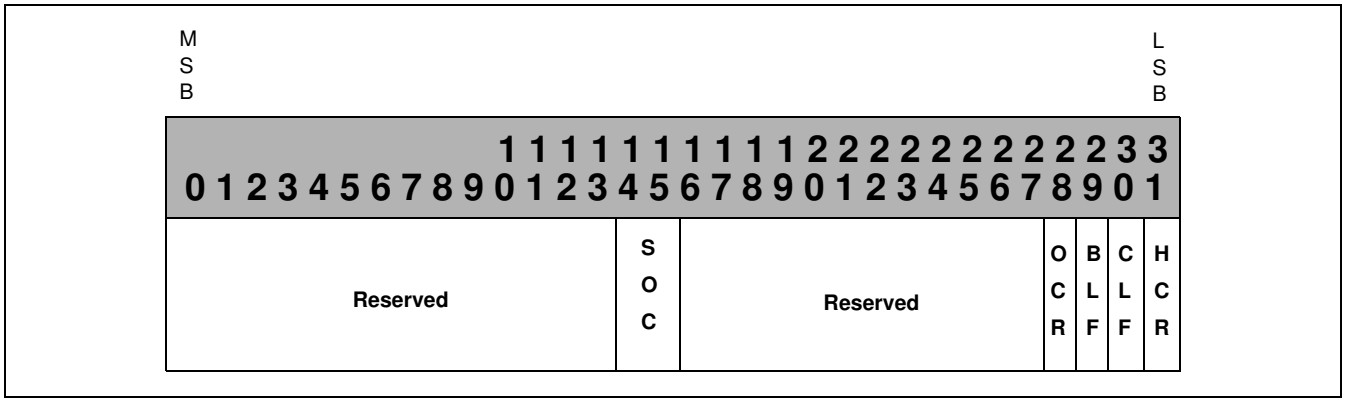


Table 30-5. HcCommandStatus Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
SOC	00b	R	R/W	HC	SchedulingOverrunCount
OCR	0b	R/W	R/W	HC	OwnershipChangeRequest This bit is cleared after the ownership changeover is complete. This occurs when the IR bit of HcControl changes from the state it was sampled at during the write of OCR to a 1.
BLF	0b	R/W	R/W	HC	BulkListFilled
CLF	0b	R/W	R/W	HC	ControlListFilled
HCR	0b	R/W	R/W	HC	HostControllerReset

30.3.4.4 HcInterruptStatus Register

Address Offset: 0x00C

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-8. HcInterruptStatus Register Format

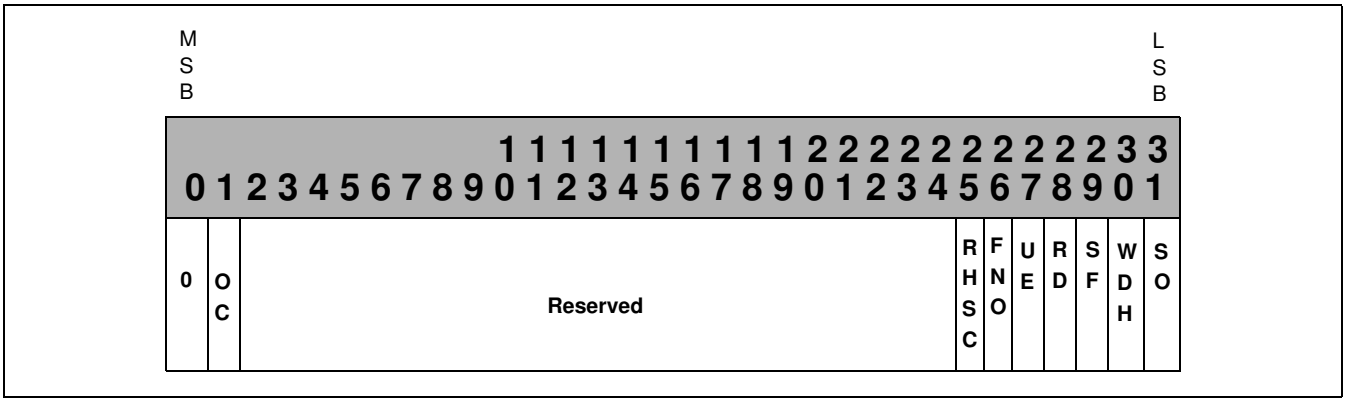


Table 30-6. HcInterruptStatus Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
OC	0b	R/W	R/W	HC	OwnershipChange Interrupt generated by this event is a System Management Interrupt (SMI) on USB11HOST2P pin USBINT2.
RHSC	0b	R/W	R/W	HC	RootHubStatusChange
FNO	0b	R/W	R/W	HC	FrameNumberOverflow
UE	0b	R/W	R/W	HC	UnrecoverableError
RD	0b	R/W	R/W	HC	ResumeDetected This bit is also set if DRWE is set and a connect/ disconnect event occurs.
SF	0b	R/W	R/W	HC	StartofFrame
WDH	0b	R/W	R/W	HC	WritebackDoneHead
SO	0b	R/W	R/W	HC	SchedulingOverrun

30.3.4.5 HcInterruptEnable Register

Address Offset: 0x010

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-9. HcInterruptEnable Register Format

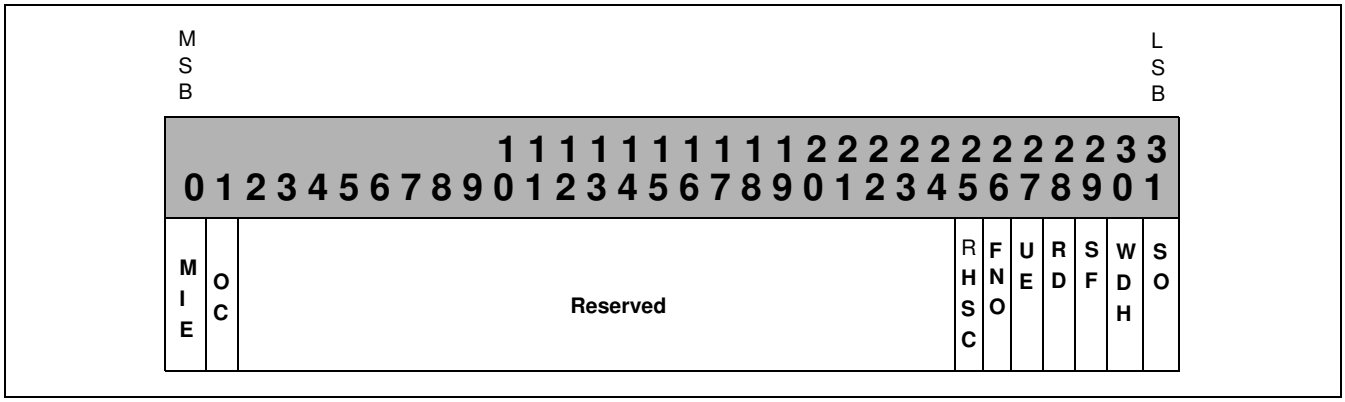


Table 30-7. HcInterruptEnable Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
MIE	0b	R/W	R	HCD	Master Interrupt Enable
OC	0b	R/W	R	HCD	OwnershipChange Interrupt Enable
RHSC	0b	R/W	R	HCD	RootHubStatusChange Interrupt Enable
FNO	0b	R/W	R	HCD	FrameNumberOverflow Interrupt Enable
UE	0b	R/W	R	HCD	UnrecoverableError Interrupt Enable
RD	0b	R/W	R	HCD	ResumeDetected Interrupt Enable
SF	0b	R/W	R	HCD	StartofFrame Interrupt Enable
WDH	0b	R/W	R	HCD	WritebackDoneHead Interrupt Enable
SO	0b	R/W	R	HCD	SchedulingOverrun Interrupt Enable

30.3.4.6 HcInterruptDisable Register

Address Offset: 0x014

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-10. HcInterruptDisable Register Format

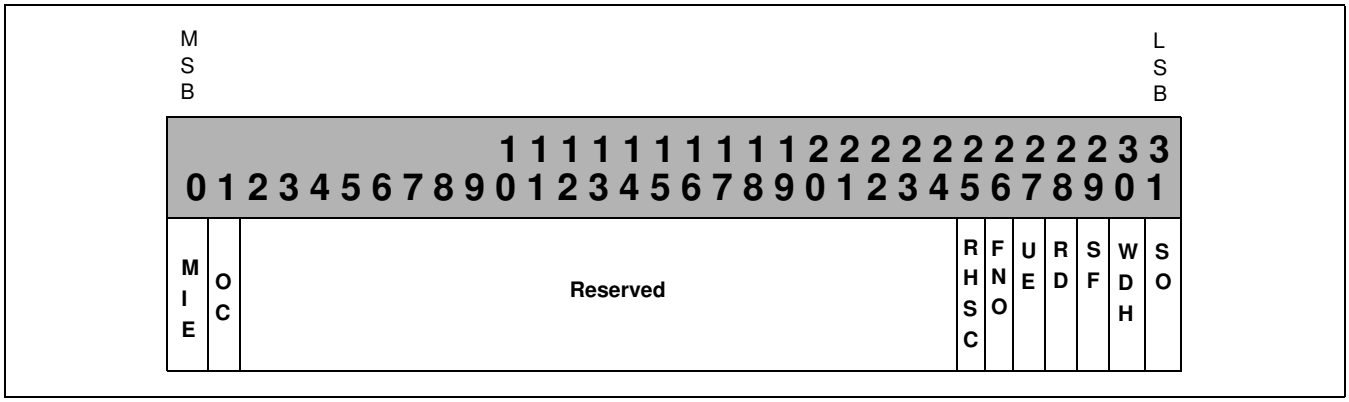


Table 30-8. HcInterruptDisable Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
MIE	0b	R/W	R	HCD	Master Interrupt Disable
OC	0b	R/W	R/W	HCD	OwnershipChange Interrupt Disable
RHSC	0b	R/W	R	HCD	RootHubStatusChange Interrupt Disable
FNO	0b	R/W	R	HCD	FrameNumberOverflow Interrupt Disable
UE	0b	R/W	R	HCD	UnrecoverableError Interrupt Disable
RD	0b	R/W	R	HCD	ResumeDetected Interrupt Disable
SF	0b	R/W	R	HCD	StartofFrame Interrupt Disable
WDH	0b	R/W	R	HCD	WritebackDoneHead Interrupt Disable
SO	0b	R/W	R	HCD	SchedulingOverrun Interrupt Disable

30.3.5 Memory Pointer Registers

The following sections define the structure of the Memory Pointer registers.

30.3.5.1 HcHCCA Register

Address Offset: 0x018

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-11. HcHCCA Register Format

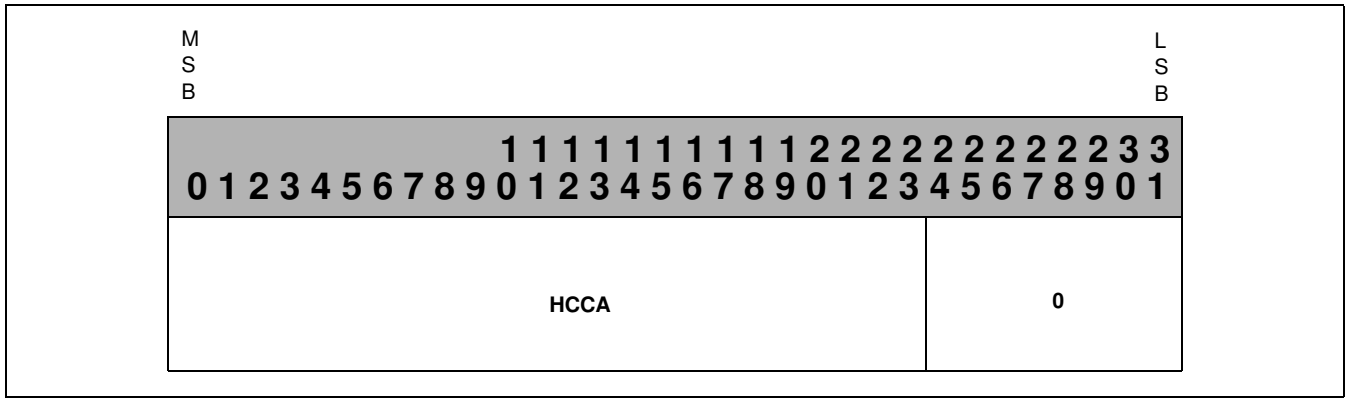


Table 30-9. HcHCCA Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
HCCA	0x0	R/W	R	HCD	Host Controller Communication Area Base address

30.3.5.2 HcPeriodCurrentED Register

Address Offset: 0x01C

Width: 32

Reset Value: 0x0000_0000

HCD Access: R

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-12. HcPeriodCurrentED Register Format

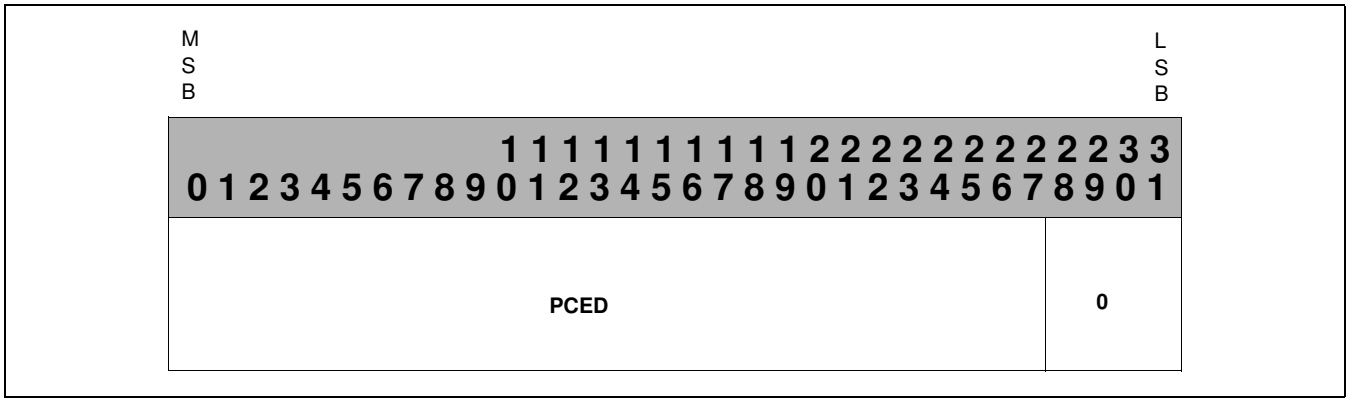


Table 30-10. HcPeriodCurrentED Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
PCED	0x0	R	R/W	HC	PeriodCurrentED

30.3.5.3 HcControlHeadED Register

Address Offset: 0x020

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-13. HcControlHeadED Register Format

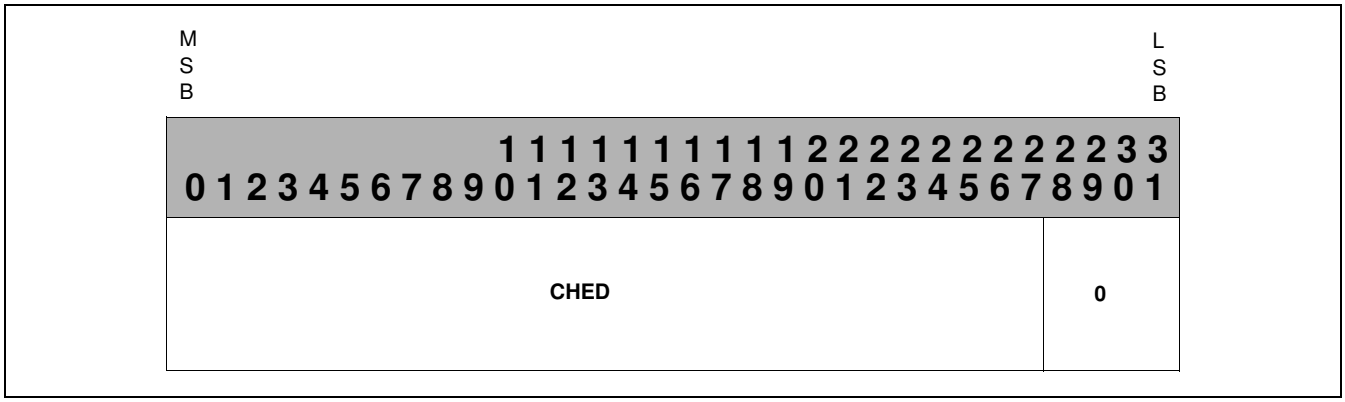


Table 30-11. HcControlHeadED Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
CHED	0x0	R/W	R	HCD	ControlHeadED

30.3.5.4 HcControlCurrentED Register

Address Offset: 0x024

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-14. HcControlCurrentED Register Format

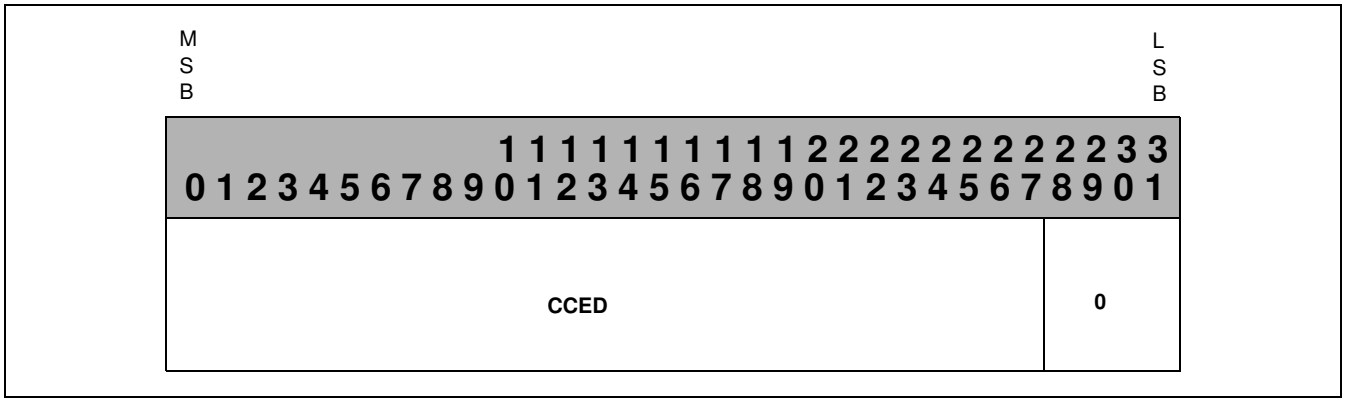


Table 30-12. HcControlCurrentED Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
CCED	0x0	R/W	R/W	HCD	ControlCurrentED

30.3.5.5 HcBulkHeadED Register

Address Offset: 0x028

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-15. HcBulkHeadED Register Format

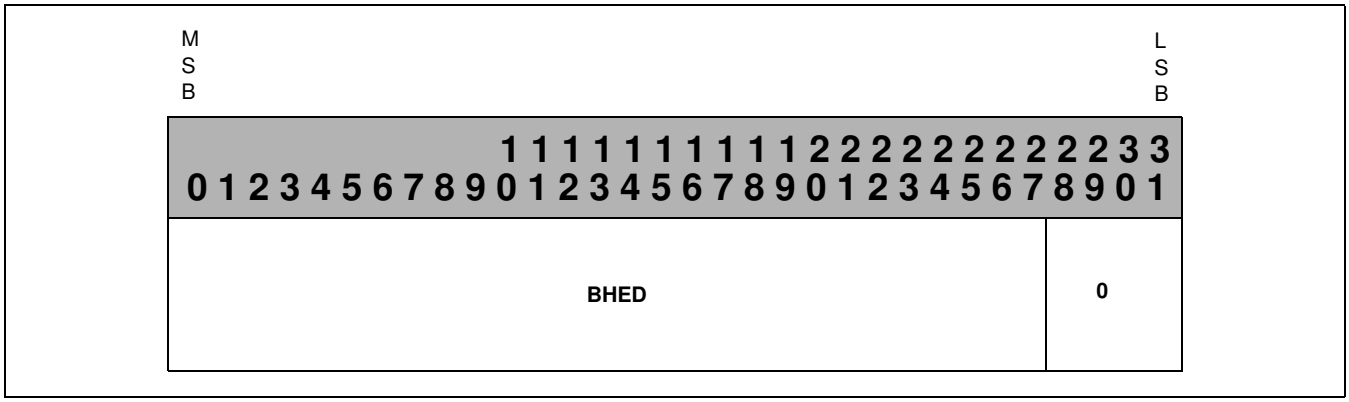


Table 30-13. HcBulkHeadED Register Format

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
BHED	0x0	R/W	R	HCD	BulkHeadED

30.3.5.6 HcBulkCurrentED Register

Address Offset: 0x02C

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-16. HcBulkCurrentED Register Format

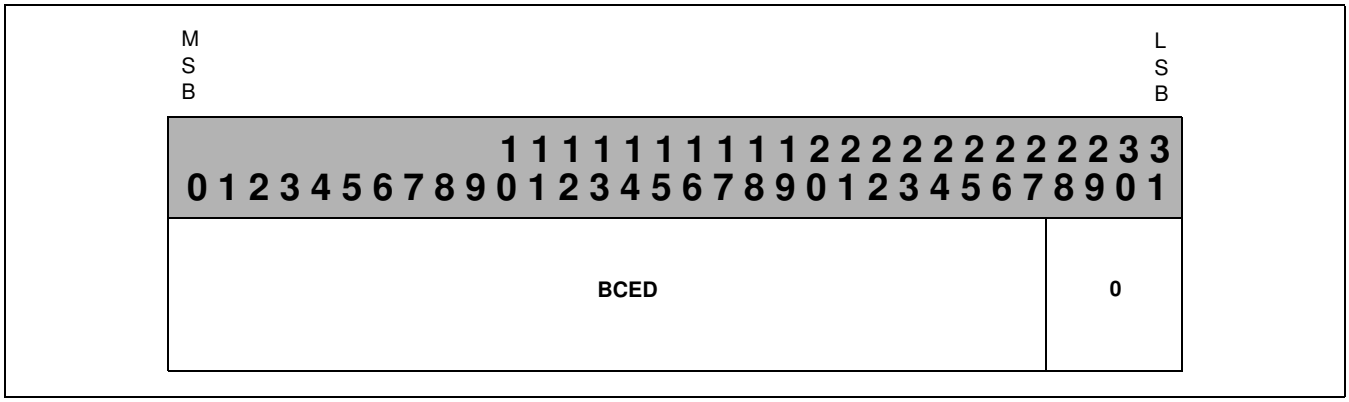


Table 30-14. HcBulkCurrentED Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
BCED	0x0	R/W	R/W	HCD	BulkCurrentED

30.3.5.7 HcDoneHead Register

Address Offset: 0x030

Width: 32

Reset Value: 0x0000_0000

HCD Access: R

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-17. HcDoneHead Register Format

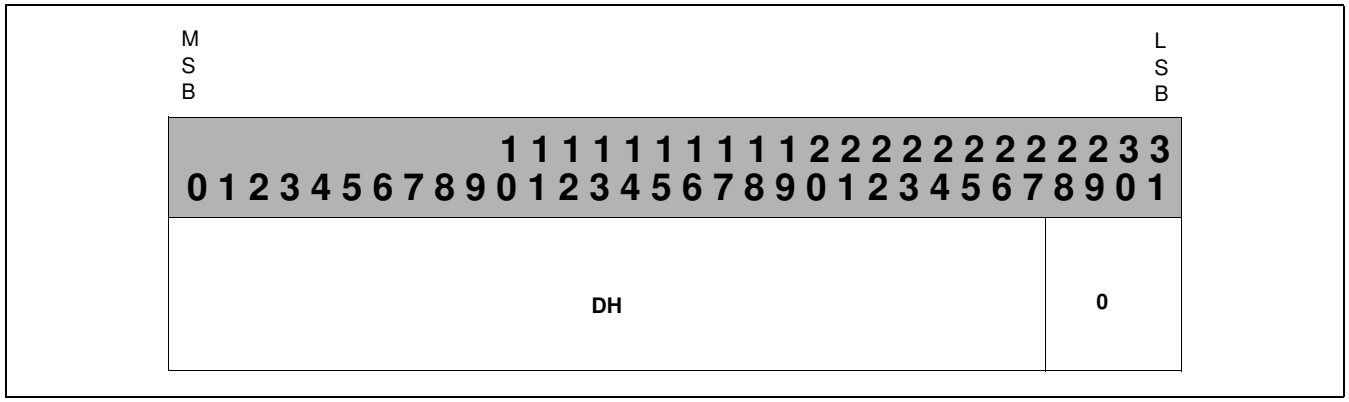


Table 30-15. HcDoneHead Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
DH	0x0	R	R/W	HC	DoneHead

30.3.6 Frame Counter Registers

The following sections define the structure of the Frame Counter registers.

30.3.6.1 HcFmInterval Register

Address Offset: 0x034

Width: 32

Reset Value: 0x2778_2EDF

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-18. HcFmInterval Register Format

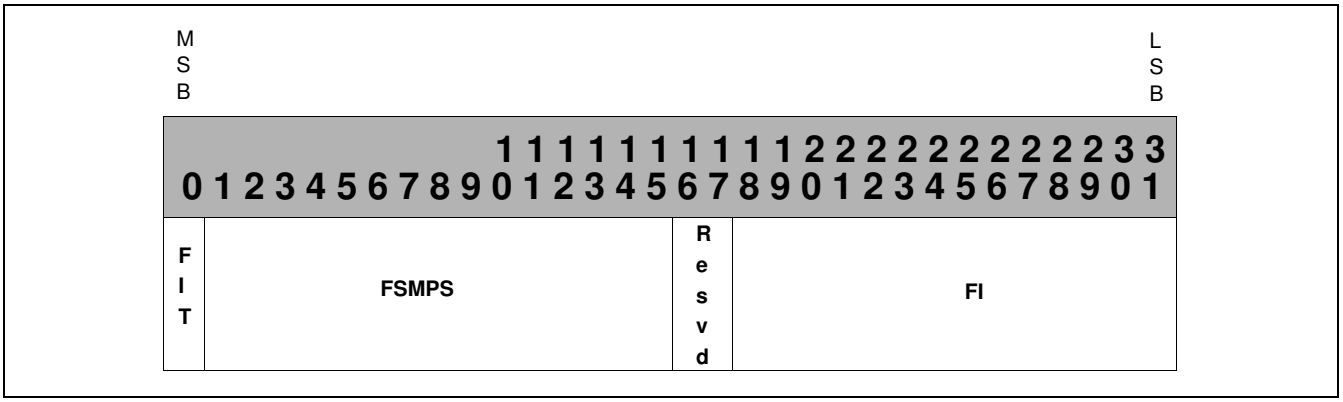


Table 30-16. HcFmInterval Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Description
FIT	0b	R/W	R	FrameIntervalToggle
FSMPS	0x2778	R/W	R	FSLargestDataPacket The reset value of 2778h reflects an overhead time of 210 full speed bits. FSMPS=(11999-210) x 6/7 = 10104 = 2778h
FI	0x2EDF	R/W	R	FrameInterval

30.3.6.2 HcFmRemaining Register

Address Offset: 0x038

Width: 32

Reset Value: 0x0000_0000

HCD Access: R

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-19. HcFmRemaining Register Format

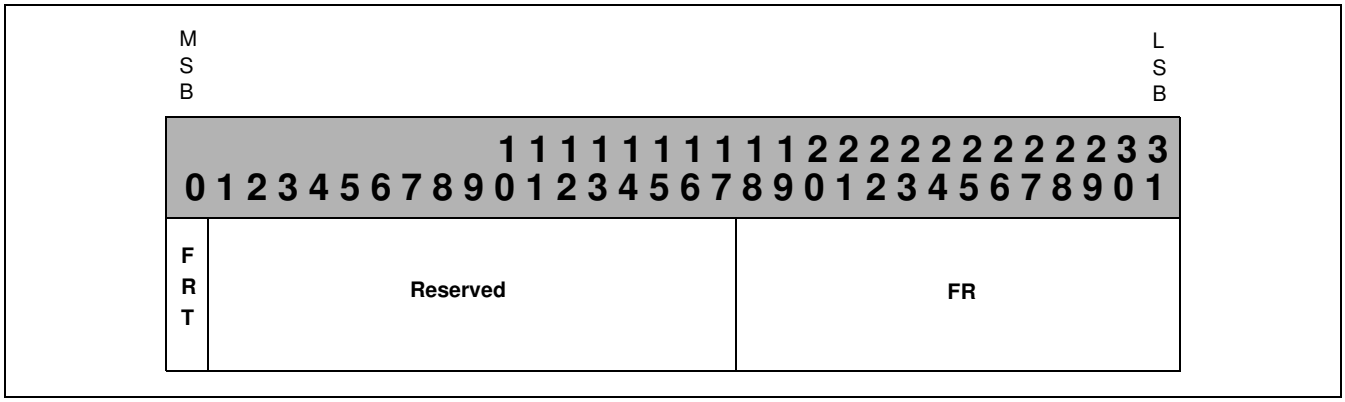


Table 30-17. HcFmRemaining Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
FRT	0x0b	R	R/W	HC	FrameRemainingToggle
FR	0x0	R	R/W	HC	FrameRemaining

30.3.6.3 HcFmNumber Register

Address Offset: 0x03C

Width: 32

Reset Value: 0x0000_0000

HCD Access: R

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-20. HcFmNumber Register Format

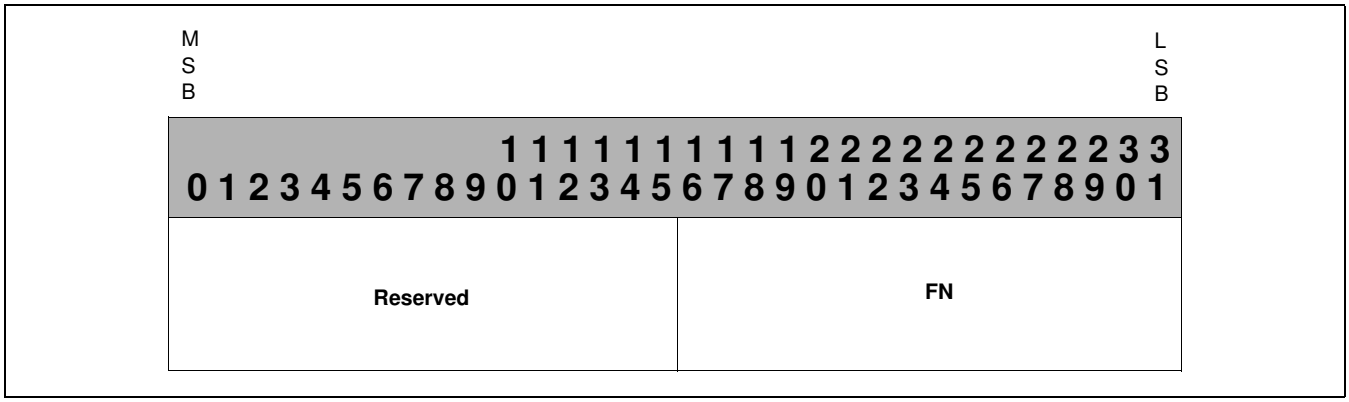


Table 30-18. HcFmNumber Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
FN	0x0	R	R/W	HC	FrameNumber This is incremented when *HcFmRemaining* is loaded with **FI** after a SOF event. It will be rolled over to 0h after ffffh.

30.3.6.4 HcPeriodicStart Register

Address Offset: 0x040

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-21. HcPeriodicStart Register Format

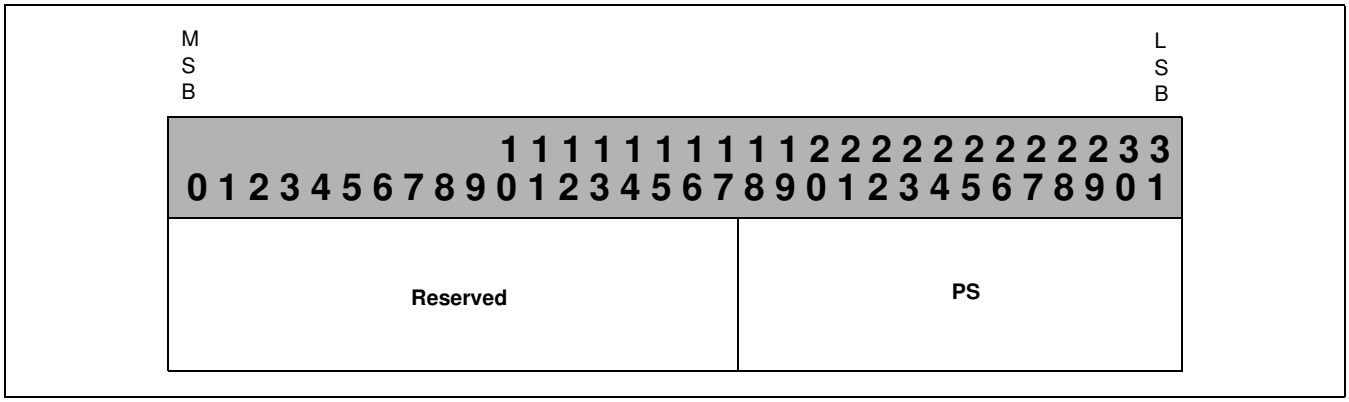


Table 30-19. HcPeriodicStart Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
PS	0x0	R/W	R	HCD	PeriodicStart 2A27h is a typical value to load into this field.

30.3.6.5 HcLSThreshold Register

Address Offset: 0x044

Width: 32

Reset Value: 0x0000_0628

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-22. HcLSThreshold Register Format

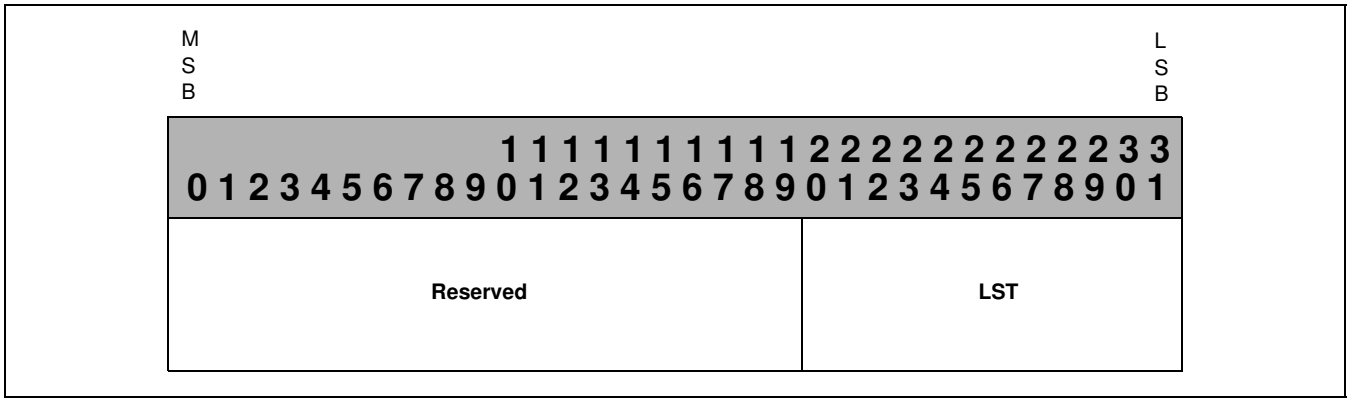


Table 30-20. HcLSThreshold Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
LST	0x628	R/W	R	HCD	LSThreshold

30.3.7 Root Hub Registers

The following sections define the structure of the Root Hub registers.

30.3.7.1 HcRhDescriptorA Register

Address Offset: 0x048

Width: 32

Reset Value: 0x10000_0902

HCD Access: R/W-R

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-23. HcRhDescriptorA Register Format

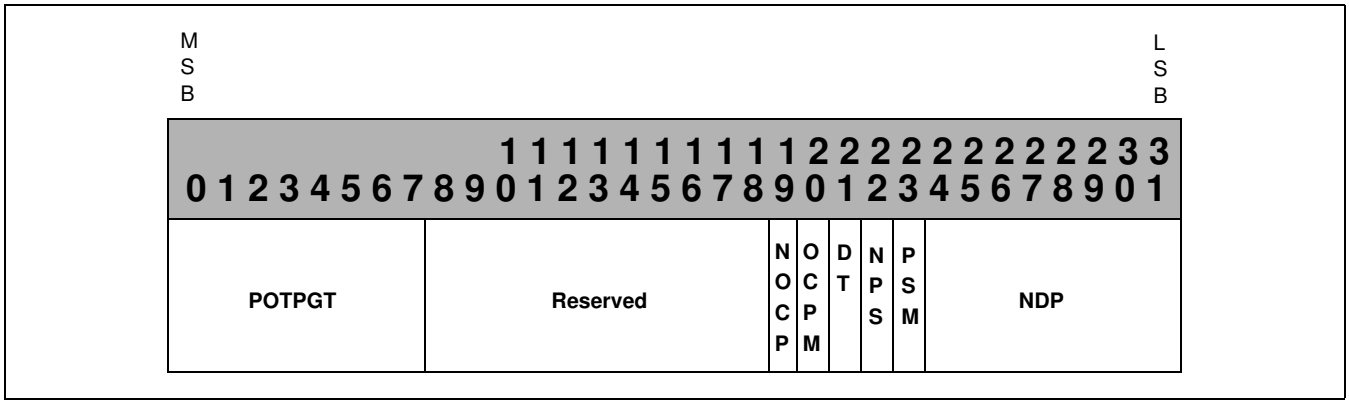


Table 30-21. HcRhDescriptorA Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
POTPGT	0x10	R/W	R	HCD	PowerOnToPowerGoodTime The reset value is 32 ms.
NOCP	0b	R/W	R	HCD	NoOverCurrentProtection
OCPM	1b	R/W	R	HCD	OverCurrentProtectionMode
DT	0b	R	R	NA	DeviceType
NPS	0b	R/W	R	HCD	NoPowerSwitching
PSM	1b	R/W	R	HCD	PowerSwitchingMode
NDP	0x02	R	R	NA	NumberDownstreamPorts USB11HOST2P supports two downstream ports.

30.3.7.2 HcRhDescriptorB Register

Address Offset: 0x04C

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-24. HcRhDescriptorB Register Format

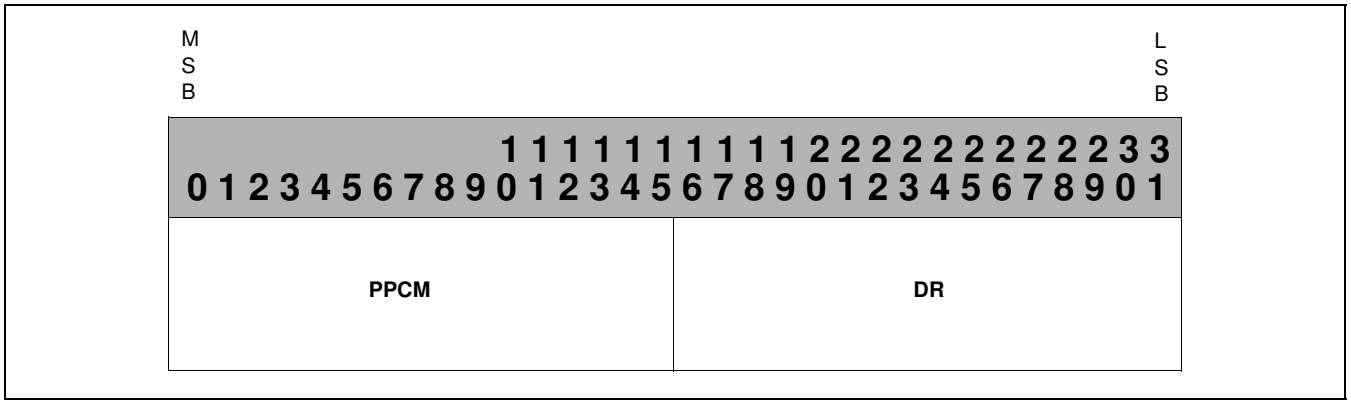


Table 30-22. HcRhDescriptorB Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Write Priority	Description
PPCM	0x0	R/W	R	HCD	PortPowerControlMask USB11HOST2P bit 15: reserved = 0 bit 14: Ganged-power mask on Port #1 bit 13: Ganged-power mask on Port #2 bit 0 - 12: reserved = 0
DR	0x0	R/W	R	HCD	DeviceRemovable USB11HOST2P bit 31: reserved = 0 bit 30: Device attached to Port #1 bit 29: Device attached to Port #2 bit 16 - 28: reserved = 0

30.3.7.3 HcRhStatus Register

Address Offset: 0x050

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W-R

HC Access: R/W-R

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-25. HcRhStatus Register Format

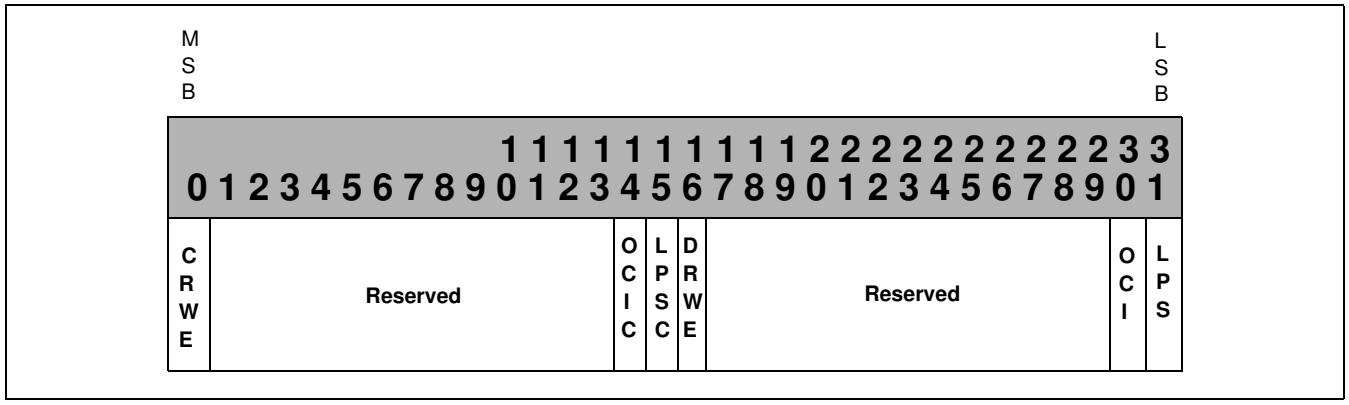


Table 30-23. HcRhStatus Register Field Description (Sheet 1 of 2)

Register Field	Root Hub Reset Value	HCD Access	HC Access	Write Priority	Description
CRWE	0b	W	R	HCD	ClearRemoteWakeupEnable Writing DRWE and CRWE to a value of '1' simultaneously is undefined and should not be attempted. Always read as 0 by HCD.
OCIC	0b	R/W	R/W	HC	OverCurrentIndicatorChange
LPSC	0b	R/W	R	HCD	LocalPowerStatusChange Writing LPSC and LPS to a value of '1' simultaneously is undefined and should not be attempted.
DRWE	0b	R/W	R	HCD	DeviceRemoteWakeupEnable Writing DRWE and CRWE to a value of '1' simultaneously is undefined and should not be attempted.
OCI	0b	R	R/W	HC	OverCurrentIndicator

Table 30-23. HcRhStatus Register Field Description (Sheet 2 of 2)

Register Field	Root Hub Reset Value	HCD Access	HC Access	Write Priority	Description
LPS	0b	R/W	R	HCD	LocalPowerStatus Writing LPS and LPSC to a value of '1' simultaneously is undefined and should not be attempted.

30.3.7.4 HcRhPortStatus[1:2] Register

These registers are used to control and report port events on each of the two ports. NumberDownstreamPorts, two for USB11HOST2P, represents the number of HcRhPortStatus registers that are implemented in the USB11HOST2P core.

Address Offset for PortStatus1: 0x054

Address Offset for PortStatus2: 0x058

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R/W

Refer to the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a* for details regarding the use of this register and for the description of each bit. Bit descriptions are only included in this document when additional information can be provided or for implementation-specific bits. The bit mnemonics and names used in this document are the same as those given in the OpenHCI specification. However, when referring to the OpenHCI specification, since it is architected in little endian format and the USB11HOST2P core is architected in big endian format, disregard the OpenHCI specification's register bit numbering and use what is documented here. All register contents are shown in big endian format, so for any register fields which contain a binary value, the most significant bit in the field is the bit with the lowest bit number.

Figure 30-26. HcRhPortStatus[1:2] Register Format

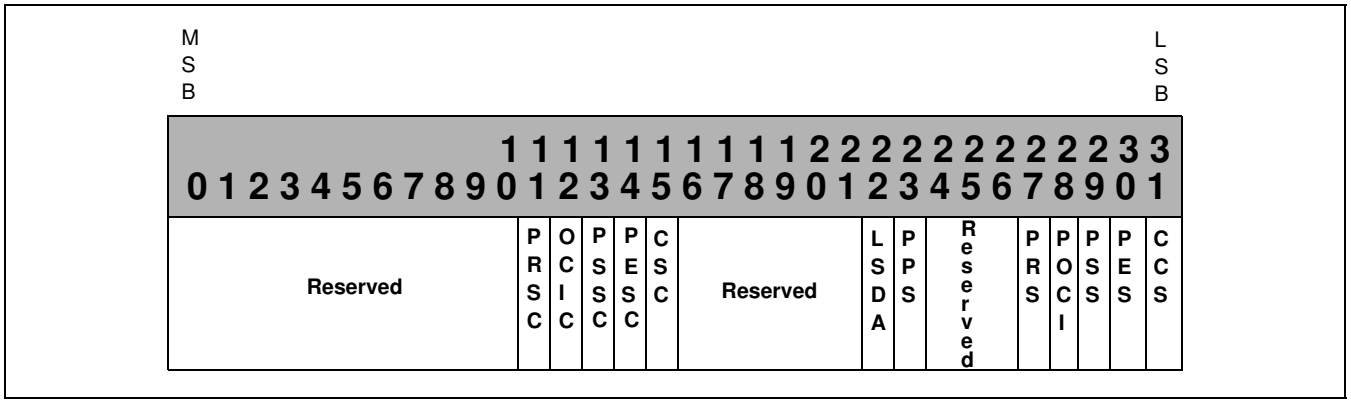


Table 30-24. HcRhPortStatus[1:2] Register Field Description (Sheet 1 of 2)

Register Field	Root Hub Reset Value	HCD Access	HC Access	Write Priority	Description
PRSC	0b	R/W	R/W	HCD	PortResetStatusChange This bit is set at the end of the 50-ms port reset signal.
OCIC	0b	R/W	R/W	HCD	PortOverCurrentIndicatorChange
PSSC	0b	R/W	R/W	HCD	PortSuspendStatusChange
PESC	0b	R/W	R/W	HCD	PortEnableStatusChange
CSC	0b	R/W	R/W	HCD	ConnectStatusChange

Table 30-24. HcRhPortStatus[1:2] Register Field Description (Sheet 2 of 2)

Register Field	Root Hub Reset Value	HCD Access	HC Access	Write Priority	Description
LSDA	0b	R/W	R/W	HCD	LowSpeedDeviceAttached Writing LSDA and PPS to a value of '1' simultaneously is undefined and should not be attempted.
PPS	0b	R/W	R/W	HCD	PortPowerStatus Writing LSDA and PPS to a value of '1' simultaneously is undefined and should not be attempted.
PRS	0b	R/W	R/W	HCD	PortResetStatus
POCI	0b	R/W	R/W	HCD	PortOverCurrentIndicator Writing PSS and POCI to value of '1' simultaneously is undefined and should not be attempted
PSS	0b	R/W	R/W	HCD	PortSuspendStatus Writing PSS and POCI to value of '1' simultaneously is undefined and should not be attempted
PES	0b	R/W	R/W	HCD	PortEnableStatus Writing CCS and PES to a value of '1' simultaneously is undefined and should not be attempted. **Note:** A SetPortEnable command will set PES but the root hub port will not be enabled by the SetPortEnable if PES was previously 0. Only a port reset will enable the root hub ports.
CCS	0b	R/W	R/W	HCD	CurrentConnectStatus Writing CCS and PES to a value of '1' simultaneously is undefined and should not be attempted.

30.3.8 OPB Configuration Registers

The following sections define the structure of the OPB Configuration registers.

30.3.8.1 OPBMC Register

The ***OPBMC*** register controls attributes of the OPB master interface.

The OPB Master Interface of the 440EP2P core may make locked next-sequential requests on the OPB in order to burst data to and from system memory. The maximum number of OPBMXFERACKs that can be done in an uninterrupted sequence can be controlled via the **DISABLE** and **MAXACK** fields.

The **DISABLE** field controls whether or not locked next-sequential requests are allowed to be generated by the OPB master interface. If **DISABLE** is '0', then locked next-sequential cycles may occur. If **DISABLE** is '1', then locked next-sequential cycles will not occur.

When the **DISABLE** field is '0' and the **MAXACK** field has a value of 0, then the OPB master interface can conduct as many continuously locked transfers as it deems necessary to transfer data in a burst fashion. The reset value of this register is 0. The **MAXACK** field controls the maximum number of transfer acknowledges that can occur while the OPB master bus is continuously locked. A non-zero value in **MAXACK** is interpreted as 4 times **MAXACK**. Thus a value of 0x01 for **MAXACK** means that the OPB master may conduct up to 4 transfers under continuous lock. On the second-to-last transfer acknowledge (that is, if **MAXACK** = 0x01, then the 3rd transfer acknowledge), the OPBMPENDREQ signal is sampled. If it is asserted (high), then the OPB master controller will relinquish the bus for at least one cycle. If OPBMPENDREQ is sampled inactive (low), then the OPB master may maintain the bus lock and will not sample OPBMPENDREQ for another 4* **MAXACK** transfers. The sampling of OPBMPENDREQ is only conducted at **MAXACK** transfer acknowledge intervals.

Address Offset: 0x070

Width: 32

Reset Value: 0x0000_0000

HCD Access: R/W

HC Access: R

Figure 30-27. OPBMC Register Format

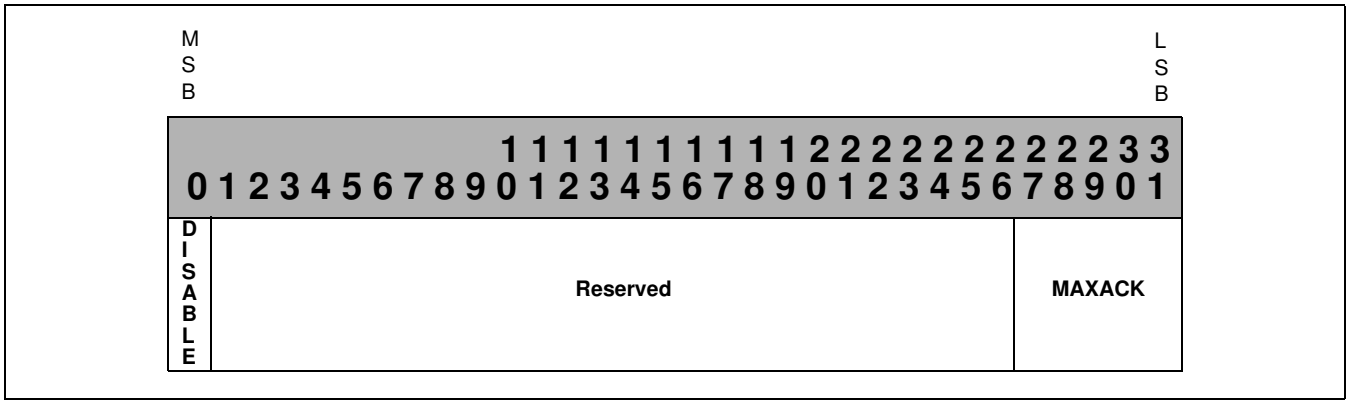


Table 30-25. OPBMC Register Field Description

Register Field	System Reset Value	HCD Access	HC Access	Description
DISABLE	0	R/W	R	DISABLE: When set to '1', the OPB master interface will not drive MBUSLOCK and MSEQADDR active, forcing the OPB master interface to assert MREQ and wait for OPBMGRANT on each request. When set to '0', the OPB master interface will drive MBUSLOCK and MSEQADDR active when more than four consecutive bytes are requested by the Host Controller.
MAXACK	0x0	R/W	R	OPB Master Maximum Continuously Locked Transfer Control 00000: unlimited continuously locked transfers 00001: up to 4 continuously locked transfers 00010: up to 8 continuously locked transfers ... 11110: up to 120 continuously locked transfers 11111: up to 124 continuously locked transfers

30.3.9 Programming Notes

This section describes the features of the PPC440EP USB 1.1Host support (USB11HOST2P) which are implementations of specific interpretations of either the *Universal Serial Bus Specification Revision 1.1* or the *OpenHCI, Open Host Controller Interface Specification for USB Release 1.0a*.

1. In section 8.7.2 of the *USB 1.1 spec*, it states that "USB devices timeout no sooner than 16 bit times and no latter (sic) than 18 bit times after the end of the previous EOP." The PPC440EP USB 1.1Host support times out at 16 bit times.
2. If an Isochronous ED has the ED(DR) = 00 or 11, then the ITD will use full word 0's bits 11 and 12 as the DP value, even though the DP field is undefined for an ITD. This should be considered a programming error. There is no hardware mechanism to detect this error condition.
3. If HCFS transitions from USBOPERATIONAL to any other state while the Control or Bulk lists are actively being processed, it is possible that the Endpoint Descriptor that is being processed during this transition will not be the first Endpoint Descriptor to be processed when reentering USBoperational.
4. For TDs which attempt to wrap data buffers from the end of a page to the beginning of the SAME page (that is, bufferstart > bufferend), the PPC440EP USB 1.1Host support will calculate a buffer size which is 8193 + bufferend - bufferstart.
5. ITD zero-length buffers are not possible during the ITD's last frame.
6. When USBOPERATIONAL is entered from USBSUSPEND, USBRESUME, or USBRESET, if the HcControlCurrentED or HcBulkCurrentED registers are non-zero, when the list is enabled, processing will continue with the ED whose address is in the list's CurrentED register. To force the USB11HOST2P to start from the list's HeadED, while the USB11HOST2P is NOT in USBOPERATIONAL, but after any processing of a TD that was active when USBOPERATIONAL was left, clear the list's CurrentED and be sure that the associated ListFilled bit is set.
7. When the Largest Data Packet Counter reaches a value of zero, no full speed transactions will be initiated by the host controller, including zero length data transfers.

30.4 USB 2.0 Device Interface

The PPC440EP Embedded Processor implements the Universal Serial Bus (USB) version 2.0 high speed (480 Mbps) and full speed (12 Mbps) device functions. The USB 2.0 Device controller has support for three IN Endpoints and three OUT Endpoints in addition to the required Endpoint 0. Endpoint 0 supports the required Control type transfers, while the additional Endpoints can be software controlled to support Bulk, Interrupt, or Isochronous transfer types. Each Endpoint can also be software controlled to select either DMA or interrupt requests for FIFO service.

The USB 2.0 Device support system access is provided through an OPB 32-bit slave interface. The USB 2.0 Device support USB cable connection is provided through either an 8-bit Universal Transceiver Macrocell Interface (UTMI) for high and full speed implementations or through a USB 1.1 Full Speed Transceiver Interface for full speed only implementations. The UTMI interface is used to connect a UTMI+ compliant off-chip or on-chip PHY. The USB 1.1 Full Speed Transceiver Interface is used to connect either an off-chip USB 1.1 Transceiver or an on-chip USB Full Speed Transceiver I/O cell such as BUSB5LP. This function incorporates the configurable, synthesizable, Mentor Graphics Inventra™ library USB 2.0 High/Full Speed Function controller and a single one-port SRAM array and its associated BIST controller for implementing all data storage FIFOs. *Figure 30-29* on page 729 shows a typical system implementation of this core.

30.4.1 Features

- USB 2.0 high and full speed device controller.
- 32-bit OPB slave interface.
- Provides both an 8-bit 60MHz UTMI+ interface for high and full speed operation and a USB 1.1 full speed transceiver interface for full speed only operation.
- DMA request support per Endpoint.
- Supports total of three Endpoints in each direction, including the required control Endpoint 0.
- Endpoints 1 to 3 can be IN only, OUT only, IN and OUT with shared FIFOs.
- Endpoints 1 to 3 can be configured to support either Interrupt/Bulk only transfer types, software programmable Interrupt/Bulk, or Isochronous transfer types.

Figure 30-28. Typical USB 2.0 Device System Application Block Diagram

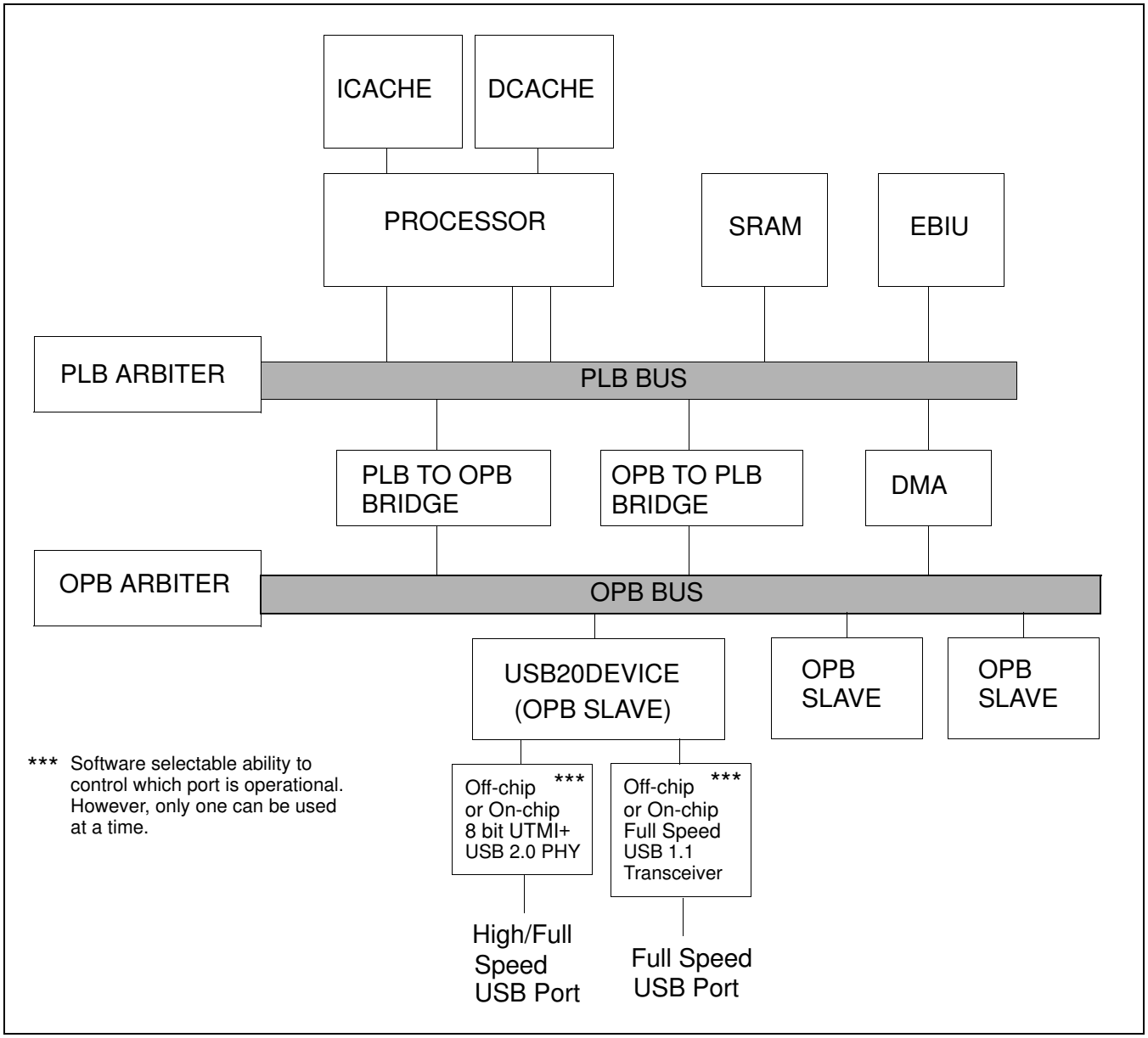
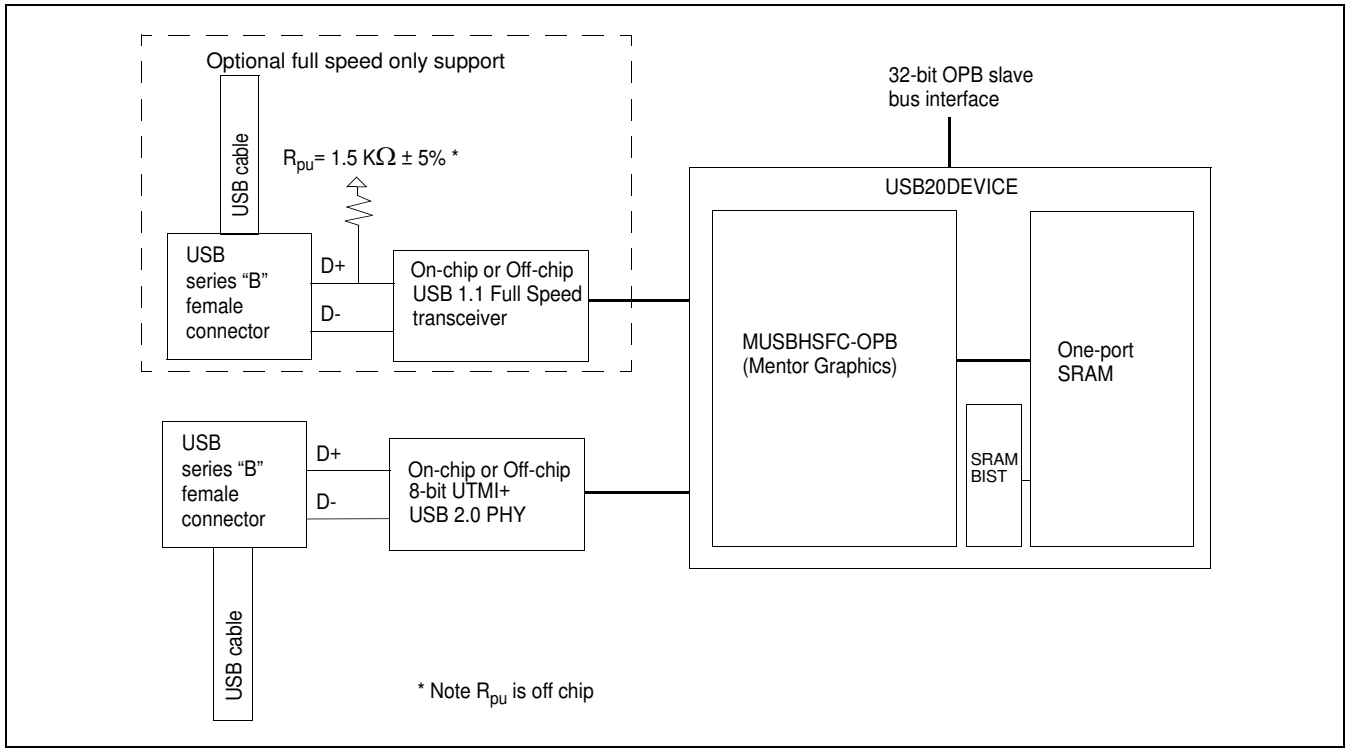


Figure 30-29. USB 2.0 Device System Block Diagram



30.4.2 References

- *USB Version 2.0 Specification*
- *USB Enhanced Host Controller Interface Specification*
- *USB Open Host Controller Interface Specification*
- *Inventra MUSBHSFC-OPB User Guide*, PD-40131.005-FC, 07/04 or later
- *Inventra MUSBHSFC-OPB Product Specification*, PS-40131.005-FC, 07/04 or later
- *Inventra MUSBHSFC-OPB Programmer's Guide*, PG-40131.005-FC, 07/04 or later
- *OPB Architecture Version 1.4 Specification*

30.4.3 Software Interface

All internal registers for the USB 2.0 Device support are contained within the Mentor Graphics Inventra Library USB 2.0 High/Full Speed Function Controller core. Because of this, the Mentor Graphic core's product specification and programmer's guide can be referenced for register and programming information. Refer to *Section 30.4.2 References* on page 729 for these documents.

The following sections provide the required detailed information concerning the USB 2.0 Device support register set as it is instantiated in the PPC440EP chip.

30.4.4 Memory Map Definitions

The USB 2.0 Device support memory map is divided into three sections:

1. Common Registers.
   These registers provide control and status for the entire function. See *Common Registers* on page 730.
2. Indexed Registers.
   These registers provide control and status for one Endpoint. There is an USB2D0_INMAXP and USB2D0_INCSR register for each IN Endpoint in the USB20DEVICE and a USB2D0_OUTMAXP, USB2D0_OUTCSR, and USB2D0_OUTCOUNT for each OUT Endpoint (except for Endpoint 0 which has a reduced registered set). Only the registers for one IN Endpoint and one OUT Endpoint appear in the register map at any one time. The Endpoints are selected by writing the Endpoint number to the USB2D0_INDEX register. Therefore, to access the registers for IN Endpoint 1 and OUT Endpoint 1, 1 must first be written to the USB2D0_INDEX register and then the control and status registers appear in the memory map. See *Indexed Registers* on page 740.
3. FIFOs.
   The FIFOs for each IN Endpoint appear as a 32-bit double words consecutively in the memory map. The FIFOs for each OUT Endpoint also appear consecutively at the same set of addresses. See *FIFOs* on page 749.

The following sections identify and classify the USB 2.0 Device support registers and FIFOs, and provide size and access information.

30.4.5 Common Registers

Table 30-26 contains the Common Register set for the USB 2.0 Device function. The following sections define the bit level detail for the registers.

.

Table 30-26. Common Registers

Register Name	Description	Address	Size (bits)	Access	Page
USB2D0_INTRIN	Interrupt register for Endpoint 0 plus IN Endpoints 1 to 3	0x0 5000 0100	16	R	731
USB2D0_POWER	Power management register	0x0 5000 0102	8	R/W	731
USB2D0_FADDR	Function address register	0x0 5000 0103	8	R/W	733
USB2D0_INTRINE	Interrupt enable register for USB2D0_INTRIN	0x0 5000 0104	16	R/W	734
USB2D0_INTROUT	Interrupt register for OUT Endpoints 1 to 3	0x0 5000 0106	16	R	734
USB2D0_INTRUSBE	Interrupt enable register for USB2D0_INTRUSB	0x0 5000 0108	8	R/W	735
USB2D0_INTRUSB	Interrupt register for common USB interrupts	0x0 5000 0109	8	R	736
USB2D0_INTROUTE	Interrupt enable register for IntrOut	0x0 5000 010A	16	R/W	737
USB2D0_TSTMODE	Enables the USB 2.0 test modes	0x0 5000 010C	8	R/W	738
USB2D0_INDEX	Index register for selecting the Endpoint status/control registers	0x0 5000 010D	8	R/W	739
USB2D0_FRAME	Frame number	0x0 5000 010E	16	R	739

30.4.5.1 USB2D0_INTRIN

USB2D0_INTRIN indicates which of the interrupts for IN Endpoints 1 – 3 are currently active. It also indicates whether the Endpoint 0 interrupt is currently active.

Note: Bits relating to Endpoints that have not been configured will always return 0. All active interrupts are cleared when this register is read.

Address: 0x50000100

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 30-30. Interrupts for Endpoints 0 and IN 1–3 Register (USB2D0_INTRIN)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	IN Endpoint 3 interrupt	
13	EP2	IN Endpoint 2 interrupt	
14	EP1	IN Endpoint 1 interrupt	
15	EP0	Endpoint 0 interrupt	

30.4.5.2 USB2D0_POWER

USB2D0_POWER controls Suspend and Resume signaling, and high-speed operation.

Address: 0x50000102

Size (bits): 8

Access: Read/Write

Reset: 0x20

Figure 30-31. USB2D0_POWER Register

Bit	Mnemonic	Description	Comments
0	ISOU	ISO Update 0 Do not wait for SOF token. 1 Wait for an SOF token from the time IPR is set before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.	The ISOU bit only affects Endpoints performing Isochronous transfers. It affects all IN Isochronous Endpoints and is normally used as a method of ensuring a "clean" start-up of an IN Isochronous pipe.
1	FSPHYE	Full-Speed PHY Enable 0 USB 1.1 PHY interface is not selected. 1 Select the USB 1.1 PHY interface.	The USB 1.1 PHY interface is limited to Full-speed operations. No attempt should be made to use High-speed mode when this interface has been selected. The FSPHYE bit should be set to 1 at power-up if the USB 1.1 PHY interface is required. Otherwise, it should be left at 0.

Figure 30-31. USB2D0_POWER Register (continued)

Bit	Mnemonic	Description	Comments
2	HSE	High-Speed Enable 0 Device only operates in Full-Speed mode. 1 Negotiate for High-Speed mode when the device is reset by the hub.	The HSE bit is used to disable High-Speed operation. Normally the interface will automatically negotiate for high-speed operation, when it is reset, by sending a "chirp" to the hub. However if this bit is cleared then the interface will not send any "chirps" to the hub so the function will remain in Full-Speed mode, even when connected to a High-Speed capable USB.
3	HSM	High-Speed Mode 0 Interface is not operating in High-Speed mode. 1 Interface has successfully negotiated for High-Speed mode.	The HSM bit is used to determine whether the interface is in High- or Full-Speed mode. It set to 1 when the function has successfully negotiated for high-speed operation during a USB reset.
4	RST	Reset 0 Reset signaling is not present on the bus. 1 Reset signaling is present on the bus.	The RST bit is used to determine when reset signaling is present on the USB. It is set to 1 when reset signaling is detected and remains high until the bus reverts to an idle state.
5	RSM	Resume 0 Do not generate Resume signaling. 1 Generate Resume signaling when the function is in Suspend mode.	The RSM bit is used to generate Resume signaling on the USB to perform remote wake-up from Suspend mode. Once set, it should be left high for approximately 10 ms (at least 1ms and no more than 15ms), then cleared.
6	SUSMDE	Suspend Mode 0 Exiting Suspend mode 1 Entering Suspend mode	The Suspend Mode bit is set to 1 when Suspend mode is entered. It is cleared when the USB2D0_INTRUSB register is read (as a result of receiving a Suspend interrupt). It will also be cleared if Suspend mode is exited by setting the RSM bit to initiate a remote wake-up.
7	ESUSPM	Enable Suspend Mode 0 Disable the SUSPENDM signal. 1 Enable the SUSPENDM signal.	The ESUSPM bit is set to enable the SUSPENDM signal to put the UTM (and any other hardware which uses the SUSPENDM signal) into Suspend mode. If this bit is not set, Suspend mode is detected as normal but the SUSPENDM signal remains high so that the UTM does not go into its low-power mode.

30.4.5.3 USB2D0_FADDR

USB2D0_FADDR is written with the function's 7-bit address (received through a SET_ADDRESS descriptor). It is then used for decoding the function address in subsequent token packets.

This register should be written with the address value contained in the SET_ADDRESS standard device request (see USB Specification Revision 2.0, Chapter 9), when it is received on Endpoint 0. The new address will not take effect immediately as the processor will still be using the old address for the Status stage of the device request. The function will continue to use the old address for decoding packets until the device request has completed. The status of the device request can be determined by reading bit 0 of this register. When a new address is written to this register, bit 0 is automatically set. It remains high until the device request has completed and is cleared when the new address takes effect.

Note: While the firmware may write the new address to the FADDR register immediately, it is recommended that this operation be performed in the Status phase of the operation in case the processor aborts the command.

Address: 0x50000103

Size (bits): 8

Access: Read/Write except bit 0 which is read-only

Reset: 0x00

Figure 30-32. USB2D0_FADDR Register

Bit	Mnemonic	Description	Comments
0	UPD	Update 0 Cleared when the new address takes effect (at the end of the current transfer). 1 Set when FADDR is written.	
1:7	FADDR	Function Address	

30.4.5.4 USB2D0_INTRINE

USB2D0_INTRINE provides interrupt enable bits for each of the interrupts in USB2D0_INTRIN. On reset, EP0–EP3 In are set to 1 where n is the number of IN Endpoints (in addition to Endpoint 0) that are included in the design, while the remaining bits are set to 0.

Note: Bits relating to Endpoints that have not been configured will always return 0.

Address: 0x50000104

Size (bits): 16

Access: Read/Write

Reset: 0xFF masked with the IN Endpoints implemented

Figure 30-33. USB2D0_INTRINE Register

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	Enable IN Endpoint 3 interrupt	
13	EP2	Enable IN Endpoint 2 interrupt	
14	EP1	Enable IN Endpoint 1 interrupt	
15	EP0	Enable Endpoint 0 interrupt	

30.4.5.5 USB2D0_INTROUT

USB2D0_INTROUT indicates which of the interrupts for OUT Endpoints 1 – 3 are currently active. (Endpoint 0 uses a single interrupt, included in the USB2D0_INTRIN register.) Note: Bits relating to Endpoints that have not been configured will always return 0.

Note: Bits relating to Endpoints that have not been configured will always return 0. All active interrupts are cleared when this register is read.

Address: 0x50000106

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 30-34. USB2D0_INTROUT Register

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	OUT Endpoint 3 interrupt	
13	EP2	OUT Endpoint 2 interrupt	
14	EP1	OUT Endpoint 1 interrupt	
15		Reserved	

30.4.5.6 USB2D0_INTRUSBE

USB2D0_INTRUSBE provides interrupt enable bits for each of the interrupts in USB2D0_INTRUSB.

Address: 0x50000108

Size (bits): 8

Access: Read/Write

Reset: 0x06

Figure 30-35. USB2D0_INTRUSBE Register

Bit	Mnemonic	Description	Comments
0:3		Reserved	
4	SOF	Start of Frame 0 Disable SOF interrupt in USB2D0_INTRUSB. 1 Enable SOF nterrupt in USB2D0_INTRUSB.	
5	RST	Reset 0 Reset signaling not detected on the bus. 1 Reset signaling detected on the bus.	
6	RSM	Resume 0 Resume signaling not detected on the bus while in Suspend mode. 1 Resume signaling detected on the bus while in Suspend mode.	
7	SUSP	Suspend 0 Suspend signaling not detected on the bus. 1 Suspend signaling detected on the bus.	

30.4.5.7 USB2D0_INTRUSB

USB2D0_INTRUSB indicates which USB interrupts are currently active.

Note: All active interrupts are cleared when this register is read.

Address: 0x50000109

Size (bits): 8

Access: Read/Write

Reset: 0x00

Figure 30-36. USB2D0_INTRUSB Register

Bit	Mnemonic	Description	Comments
0:3		Reserved	
4	SOF	Start of Frame 0 1 Set at the start of each frame.	
5	RST	Reset 0 Reset signaling not detected on the bus. 1 Reset signaling detected on the bus.	
6	RSM	Resume 0 Resume signaling not detected on the bus while in Suspend mode. 1 Resume signaling detected on the bus while in Suspend mode.	
7	SUSP	Suspend 0 Suspend signaling not detected on the bus. 1 Suspend signaling detected on the bus.	

30.4.5.8 USB2D0_INTROUTE

USB2D0_INTROUTE provides interrupt enable bits for each of the interrupts in USB2D0_INTROUT. On reset, EP1 Out – EP*m* Out are set to 1 where *m* is the number of OUT Endpoints (in addition to Endpoint 0) that are included in the design, while the remaining bits are set to 0.

Note: Bits relating to Endpoints that have not been configured will always return 0.

Address: 0x5000010A

Size (bits): 16

Access: Read/Write

Reset: 0xFF masked with the OUT Endpoints implemented

Figure 30-37. USB2D0_INTROUTE Register

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	Enable OUT Endpoint 3 interrupt	
13	EP2	Enable OUT Endpoint 2 interrupt	
14	EP1	Enable OUT Endpoint 1 interrupt	
15		Reserved	

30.4.5.9 USB2D0_TESTMODE

USB2D0_TESTMODE puts the function into one of the four test modes for High-Speed operation described in the USB 2.0 specification. It is not used in normal operation.

Note: Only one of bits 2:7 should be set at any time.

Address: 0x5000010C

Size (bits): 8

Access: Read/Write

Reset: 0x00

Figure 30-38. USB2D0_TESTMODE Register

Bit	Mnemonic	Description	Comments
0:1		Reserved	
2	FFS	Force Full-Speed Force the interface into Full-Speed mode	Set this bit to force the interface into Full-Speed mode when a USB reset is received.
3	FHS	Force High-Speed Force the interface into High-Speed mode	Set this bit to force the interface into High-Speed mode when a USB reset is received.
4	TESTP	Test Packet Enter Test Packet test mode.	In HS mode, set this bit to enter the Test Packet test mode. In this mode, the interface repetitively transmits a 53-byte test packet, whose form is defined in the USB Specification Revision 2.0, Section 7.1.20. **Note:** The test packet has a fixed format must be loaded into the Endpoint 0 FIFO before the test mode is entered.
5	TESTK	Test K Enter the Test K test mode	In HS mode, set this bit to enter the Test K test mode. In this mode, the interface transmits a continuous K on the bus.
6	TESTJ	Test J Enter the Test J test mode	In HS mode, set this bit to enter the Test J test mode. In this mode, the interface transmits a continuous J on the bus.
7	TSTSE0	Test SE0 NAK Enter the Test_SE0_NAK test mode	In HS mode, set this bit to enter the Test SE0 NAK test mode. In this mode, the interface remains in High-Speed mode but responds to any valid IN token with a NAK.

30.4.5.10 USB2D0_INDEX

USB2D0_INDEX determines which Endpoint control/status registers (indexed registers) are accessed at addresses 0x50000110 to 0x50000119.

Each IN Endpoint and each OUT Endpoint have their own set of control/status registers. Only one set of IN control/status and one set of OUT control/status registers appear in the memory map at any one time. Before accessing an Endpoint's control/status registers, the Endpoint number should be written to the USB2D0_INDEX register to ensure that the correct control/status registers appear in the memory map.

Address: 0x5000010D

Size (bits): 8

Access: Read/Write

Reset: 0x00

Figure 30-39. USB2D0_INDEX Register

Bit	Mnemonic	Description	Comments
0:3		Reserved	
4:7		Selected Endpoint	Bit 4 is the msb.

30.4.5.11 USB2D0_FRAME

USB2D0_FRAME holds the last received frame number.

Address: 0x5000010E

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 30-40. USB2D0_FRAME Register

Bit	Mnemonic	Description	Comments
0:3		Always 0.	
4:15		Frame number	Bit 4 is the msb.

30.4.6 Indexed Registers

There are six Endpoints in addition to Endpoint 0, three IN and three OUT. Their registers are accessed indirectly using the USB2D0_INDEX register. The number of the Endpoint must first be written to USB2D0_INDEX, then subsequent reads or writes to function specific registers (USB2D0_INCSR, USB2D0_INMAXP, USB2D0_OUTCSR, USB2D0_OUTMAXP, USB2D0_OUTCOUNT) return (or update) the information only for the Endpoint specified by the USB2D0_INDEX register.

Each of the three IN Endpoints implements two registers, USB2D0_INCSR and USB2D0_INMAXP. Note that the CSR function is defined differently depending on the Endpoint. Endpoint 0 implements only eight bits (and is called USB2D0_INCSR0), whereas Endpoints 1 to 3 implement 16 bits (referred to as USB2D0_INCSR). This explains why address 0x50000110 appears twice in the table.

Similarly, each of the three OUT Endpoints implement three registers, USB2D0_OUTCSR, USB2D0_OUTMAXP and USB2D0_OUTCOUNT. Note that the COUNT function is defined differently depending on the Endpoint. Endpoint 0 implements only eight bits (and is called USB2D0_OUTCOUNT0), whereas Endpoints 1 to 3 implement 16 bits (referred to as USB2D0_OUTCOUNT). This explains why address 0x5000011A appears twice in the table.

Table 30-27. Indexed Registers

Register Name	Description	Address	Size (bits)	Access	Page
USB2D0_INCSR0	Control Status register for Endpoint 0. (Index register set to select Endpoint 0)	0x0 5000 0110	8	R/W	741
USB2D0_INCSR	Control Status register for IN Endpoint. (Index register set to select Endpoints 1–3)	0x0 5000 0110	16	R/W	742
USB2D0_INMAXP	Maximum packet size for IN Endpoint. (Index register set to select Endpoints 1–3)	0x0 5000 0112	16	R/W	744
USB2D0_OUTCSR	Control Status register for OUT Endpoint. (Index register set to select Endpoints 1–3)	0x0 5000 0114	16	R/W	745
USB2D0_OUTMAXP	Maximum packet size for OUT Endpoint. (Index register set to select Endpoints 1–3)	0x0 5000 0116	16	R/W	747
USB2D0_OUTCOUNT0	Number of received bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)	0x0 5000 011A	8	R	748
USB2D0_OUTCOUNT	Number of bytes in OUT Endpoint FIFO. (Index register set to select Endpoints 1–3)	0x0 5000 011A	16	R	748

30.4.6.1 USB2D0_INCSR0

USB2D0_INCSR0 provides control and status bits for Endpoint 0.

Address: 0x50000110 (0x50000111 when the Index register is set to 0)

Size (bits): 8

Access: Read/Write

Reset: 0x00

Figure 30-41. USB2D0_INCSR0 Register

Bit	Mnemonic	Description	Comments
0	SSE	Serviced Setup End 0 SE bit is clear 1 Clear the SE bit	The processor writes 1 to this bit to clear the SE bit. SSE is cleared automatically after the operation is complete.
1	SOPR	Serviced Out Packet Ready 0 OPR bit is clear 1 Clear the OPR bit.	The processor writes 1 to this bit to clear the OPR bit. SOPR is cleared automatically after the operation is complete.
2	SS1	Send Stall 0 Terminate the STALL condition 1 Transmit a STALL handshake	The processor writes 1 to this bit to issue a STALL handshake. This bit is automatically cleared.
3	SE	Setup End 0 SSE bit set 1 Control tansaction ended before DE bit set	This bit is set when a control transaction ends before the DE bit has been set. An interrupt is generated and the FIFO flushed at this time. The bit is cleared by the processor writing 1 to the SSE bit.
4	DE	Data End 0 Last data packet processed 1 Last data packet	The processor sets this bit: 1. When setting IPR for the last data packet. 2. When clearing OPR after unloading the last data packet. 3. When setting IPR for a zero length data packet. It is cleared automatically.
5	SS0	Sent Stall 0 STALL terminated 1 STALL handshake has been transmitted	This bit is set when a STALL handshake has been transmitted. The processor should clear this bit.
6	IPR	In Packet Ready 0 Data packet in FIFO transmitted 1 Data packet loaded in FIFO	The processor sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when this bit is cleared.
7	OPR	Out Packet Ready 0 SOPR bit set 1 Data packet received	This bit is set when a data packet has been received. An interrupt is generated (if enabled) when this bit is set. The processor clears this bit by setting the SOPR bit.

30.4.6.2 USB2D0_INCSR

USB2D0_INCSR provides control and status bits for IN transactions through the currently selected Endpoint. There is an USB2D0_INCSR register for each IN Endpoint (not including Endpoint 0).

Address: 0x50000110

Size (bits): 16

Access: Read/Write

Reset: 0x00

Figure 30-42. USB2D0_INCSR Register

Bit	Mnemonic	Description	Comments
0	AUTOS	Automatically Set 0 Do not set IPR 1 Set IPR automatically	If the processor sets this bit, IPR is automatically set when the maximum packet size (value in USB2D0_INMAXP) is loaded into the IN FIFO. If a packet of less than the maximum packet size is loaded, IPR must be set manually. **Note:** This bit should not be set for High-Bandwidth Isochronous Endpoints.
1	ISO	ISO Mode 0 Bulk/Interrupt transfer mode 1 Isochronous transfer mode	The processor sets this bit to enable the IN Endpoint for Isochronous transfers (ISO mode), and clears it to enable the IN Endpoint for Bulk/ Interrupt transfers.
2	MODE	Mode 0 Endpoint direction is OUT 1 Endpoint direction is IN	The processor sets this bit to enable the Endpoint direction as IN, and clears it to enable the Endpoint direction as OUT. **Note:** This bit valid only when the Endpoint FIFO is used for both IN and OUT transactions, otherwise it is ignored.
3	DMAE	DMA Enable 0 Disable the DMA request for the IN Endpoint 1 Enable the DMA request for the IN Endpoint	The processor sets this bit to enable the DMA request for the IN Endpoint.
4	FDT	Force Data Toggle 0 IN data toggle switches after receiving an ACK 1 IN data toggle switches after each data packet is sent	The processor sets this bit to force the Endpoint's IN data toggle to switch after each data packet is sent regardless of whether an ACK was received. This can be used by Interrupt IN Endpoints which are used to communicate rate feedback for Isochronous Endpoints.
5:7		Reserved	
8	ITX	Incomplete Transmission 0 Not High-Bandwidth Isochronous transfer 1 Insufficient IN tokens received for split packet transmission during High-Bandwidth Isochronous transfer	When the Endpoint is being used for High-Bandwidth Isochronous transfers, this bit is set to indicate where a large packet has been split into two or three packets for transmission, but insufficient IN tokens have been received to send all the parts. **Note:** In anything other than a High-Bandwidth Isochronous transfer, this bit will always return 0.
9	CDT	Clear Data Toggle 0 Operation to reset the IN Endpoint data toggle complete 1 Reset the IN Endpoint data toggle	The processor writes 1 to this bit to reset the IN Endpoint data toggle.

Figure 30-42. USB2D0_INCSR Register (continued)

Bit	Mnemonic	Description	Comments
10	SS0	Sent Stall 0 STALL terminated 1 STALL handshake has been transmitted	This bit is set when a STALL handshake has been transmitted. The FIFO is flushed and the IPR bit is cleared. The processor should clear this bit.
11	SS1	Send Stall 0 Terminate the STALL condition 1 Transmit a STALL handshake to an IN token	The processor writes 1 to this bit to issue a STALL handshake to an IN token. The processor clears this bit to terminate the stall condition. **Note:** This bit has no effect where the Endpoint is being used for Isochronous transfers.
12	FFIFO	Flush FIFO 0 Flush complete 1 Flush the next packet to be transmitted from the Endpoint IN FIFO	The processor writes 1 to this bit to flush the next packet to be transmitted from the Endpoint IN FIFO. The FIFO pointer is reset and the IPR bit is cleared. **Note 1.** FFIFO has no effect when written before IPR has been set but can be set together with IPR to flush an unwanted packet before it is sent. **Note 2.** If the FIFO contains two packets, FFIFO will need to be set twice to completely clear the FIFO.
13	UR	Underrun 0 IN underrun processed 1 IN data underrun	In ISO mode, this bit is set when a zero-length data packet is sent after receiving an IN token with the IPR bit not set. In Bulk/Interrupt mode, this bit is set when a NAK is returned in response to an IN token. The processor should clear this bit.
14	FIFONE	FIFO Not Empty 0 No packet in the IN FIFO 1 At least one packet in the IN FIFO	This bit is set when there is at least one packet in the IN FIFO.
15	IPR	In Packet Ready 0 Space available in the FIFO 1 Data packet loaded into the FIFO	The processor sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. If the FIFO is double-buffered, it is also automatically cleared when there is space for a second packet in the FIFO. An interrupt is generated (if enabled) when the bit is cleared.

30.4.6.3 USB2D0_INMAXP

USB2D0_INMAXP defines the maximum amount of data that can be transferred through the selected IN Endpoint in a single frame/microframe (High-Speed transfers). There is an USB2D0_INMAXP register for each IN Endpoint (except Endpoint 0).

Address: 0x50000112

Size (bits): 16

Access: Read/Write

Reset: 0x00

Figure 30-43. USB2D0_INMAXP Register

Bit	Mnemonic	Description	Comments
0:2		Reserved	
3	3TRS	3 Multiplier	For Isochronous Endpoints operating in High-Speed mode and with the High-Bandwidth option enabled, the multiplier may only be either two or three (corresponding to 2TRS set or 3TRS set, respectively) and it specifies the maximum number of transactions that can take place in a single microframe. If either 3TRS or 2TRS is non-zero, the function will automatically *split any data packet written to the FIFO into up to two or three USB packets, each containing the specified payload (or less)*. The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be *transmitted* in each microframe. (For Isochronous transfers in Full-Speed mode or if High-Bandwidth is not enabled, bits 3TRS and 2TRS are ignored.)
4	2TRS	2 Multiplier	
5:15	MAXTRS	Maximum Payload/Transaction	Specifies the maximum payload in bytes that can be transferred in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full-Speed and High-Speed operations.

The maximum payload set in this register (multiplied by *m* where *m* = 2 or 3 in the case of High-Bandwidth Isochronous transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated Endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch can cause unexpected results.

The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the IN Endpoint, and should not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the Endpoint, the IN Endpoint FIFO should be completely flushed using the Flush FIFO bit in USB2D0_INCSR after writing the new value to this register.

30.4.6.4 USB2D0_OUTCSR

USB2D0_OUTCSR provides control and status bits for OUT transactions through the currently selected Endpoint. There is an USB2D0_OUTCSR register for each OUT Endpoint (except Endpoint 0).

Address: 0x50000114

Size (bits): 16

Access: Read/Write

Reset: 0x00

Figure 30-44. USB2D0_OUTCSR Register

Bit	Mnemonic	Description	Comments
0	AUTOC	Automatically Clear 0 Do not automatically clear the OPR 1 Automatically clear the OPR	If the processor sets this bit, OPR is automatically cleared when the maximum packet size (value in USB2D0_OUTMAXP) is unloaded from the OUT FIFO. If a packet of less than the maximum packet size is unloaded, OPR must be cleared manually. **Note:** This bit should not be set for High-Bandwidth Isochronous Endpoints.
1	ISO	ISO Mode 0 Disable the OUT Endpoint for Bulk/Interrupt transfers 1 Enable the OUT Endpoint for Isochronous transfers	The processor sets this bit to enable the OUT Endpoint for Isochronous transfers, and clears it to enable the OUT Endpoint for Bulk/Interrupt transfers.
2	DMAE	DMA Enable 0 Disable the DMA request for the OUT Endpoint 1 Enable the DMA request for the OUT Endpoint	The processor sets this bit to enable the DMA request for the OUT Endpoint.
3	DN	Disable NYET 0 Enable sending of NYET handshakes 1 Disable sending of NYET handshakes	The processor sets this bit to disable sending of NYET handshakes. When set, all successfully received OUT packets are ACK'd including at the point at which the FIFO becomes full. **Note:** This bit only has any effect in High-speed mode, in which mode it should be set for all Interrupt Endpoints.
4	DMAM	DMA Mode 0 Select DMA Mode 0 1 Select DMA Mode 1	Two modes of operation are supported: DMA Mode 0 in which a DMA request is generated for all received packets, together with an interrupt (if enabled); and DMA Mode 1 in which a DMA request (but no interrupt) is generated for OUT packets of the size specified in USB2D0_OUTMAXP and an interrupt (but no DMA request) is generated for OUT packets of any other size. The MCU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
5:6		Reserved	
7	IRX	Incomplete Receive Operation 0 OPR is cleared 1 Packet in the OUT FIFO is incomplete	This bit is set in a High-Bandwidth Isochronous transfer if the packet in the OUT FIFO is incomplete because parts of the data were not received. It is cleared when OPR is cleared. **Note:** In anything other than a High-Bandwidth Isochronous transfer, this bit will always return 0.

Figure 30-44. USB2D0_OUTCSR Register (continued)

Bit	Mnemonic	Description	Comments
8	CDT	Clear Data Toggle 0 Do not reset the OUT Endpoint data toggle 1 Reset the OUT Endpoint data toggle	The processor writes 1 to this bit to reset the OUT Endpoint data toggle.
9	SS0	Sent Stall 0 STALL terminated 1 STALL handshake has been transmitted	This bit is set when a STALL handshake has been transmitted. The processor should clear this bit.
10	SS1	Send Stall 0 STALL handshake transmitted 1 Transmit a STALL handshake to a DATA packet	The processor writes 1 to this bit to issue a STALL handshake to a DATA packet. The processor clears this bit to terminate the stall condition. **Note:** This bit has no effect where the Endpoint is being used for Isochronous transfers.
11	FFIFO	Flush FIFO 0 Flush complete 1 Flush the next packet to be read from the OUT Endpoint FIFO	The processor writes 1 to this bit to flush the next packet to be read from the OUT Endpoint FIFO. **Note:** If the FIFO contains two packets, FFIFO will need to be set twice to completely clear the FIFO.
12	DATAE	Data Error 0 OPR is reset 1 Data packet has a CRC error	This bit is set at the same time that OPR is set if the data packet has a CRC error. It is cleared when OPR is cleared. **Note:** This bit is only valid when the Endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
13	OVR	Overrun 0 OUT overrun processed 1 OUT data overrun	This bit is set if an OUT packet arrives while FIFOF is set; that is, the OUT packet cannot be loaded into the OUT FIFO. The processor should clear this bit. **Note:** This bit is only valid when the Endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
14	FIFOF	FIFO Full 0 Space available in the OUT FIFO 1 No more space available in the OUT FIFO	This bit is set when no more packets can be loaded into the OUT FIFO.
15	OPR	Out Packet Ready 0 Data packet unloaded from the OUT FIFO 1 Data packet received in the OUT FIFO	This bit is set when a data packet has been received. The processor should clear this bit when the packet has been unloaded from the OUT FIFO. An interrupt is generated (if enabled) when the bit is set.

30.4.6.5 USB2D0_OUTMAXP

USB2D0_OUTMAXP defines the maximum amount of data that can be transferred through the selected OUT Endpoint in a single frame/microframe (High-Speed transfers). There is an USB2D0_OUTMAXP register for each OUT Endpoint (except Endpoint 0).

Address: 0x50000116

Size (bits): 16

Access: Read/Write

Reset: 0x00

Figure 30-45. USB2D0_OUTMAXP Register

Bit	Mnemonic	Description	Comments
0:2		Reserved	
3	3TRS	3 Multiplier	For Isochronous Endpoints operating in High-Speed mode and with the High-Bandwidth option enabled, the multiplier may only be either two or three (corresponding to 2TRS set or 3TRS set, respectively) and it specifies the maximum number of transactions that can take place in a single microframe. If either 3TRS or 2TRS is non-zero, the function will automatically *combine the separate USB packets received in any microframe into a single packet within the OUT FIFO*. The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be *received* in each microframe. (For Isochronous transfers in Full-Speed mode or if High-Bandwidth is not enabled, bits 3TRS and 2TRS are ignored.)
4	2TRS	2 Multiplier	
5:15	MAXTRS	Maximum Payload/Transaction	The maximum payload in bytes that can be transferred a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full-Speed and High-Speed operations.

The maximum payload set in this register (multiplied by *m* where *m* = 2 or 3 in the case of High-Bandwidth Isochronous transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated Endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch can cause unexpected results.

The total amount of data represented by the value written to this register (specified payload $\times$ *m*) must not exceed the FIFO size for the OUT Endpoint, and should not exceed half the FIFO size if double-buffering is required.

30.4.6.6 USB2D0_OUTCOUNT0

USB2D0_OUTCOUNT0 indicates the number of received data bytes in the Endpoint 0 FIFO.

Note: The value returned changes as the contents of the FIFO change and is only valid while OPR in USB2D0_OUTCSR is set..

Address: 0x5000011A (0x5000001B when the Index register is set to 0)

Size (bits): 8

Access: Read/Write

Reset: 0x00

Figure 30-46. USB2D0_OUTCOUNT0 Register

Bit	Mnemonic	Description	Comments
0		Reserved	
1:7	CNT	Endpoint 0 OUT Count	Bit 1 is the msb.

30.4.6.7 USB2D0_OUTCOUNT

USB2D0_OUTCOUNT holds the number of received data bytes in the packet in the OUT FIFO.

Note: The value returned changes as the FIFO is unloaded and is only valid while OPR in USB2D0_OUTCSR is set. Also pipeline delays in the CPU/OPB interface may result in the count not being synchronized with the corresponding OPB transaction.

Address: 0x5000011A

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 30-47. USB2D0_OUTCOUNT Register

Bit	Mnemonic	Description	Comments
0:2		Reserved	Always 0.
3:15	CNT	Endpoint OUT count	Bit 3 is the msb.

30.4.7 FIFOs

The function provides four addresses for the processor to access the FIFOs for each Endpoint. Writing to these addresses loads data into the IN FIFO for the corresponding Endpoint. Reading from these addresses unloads data from the OUT FIFO for the corresponding Endpoint. The FIFOs are located on 32-bit full-word boundaries. The address assignment is defined in *Table 30-28*.

Note: Transfers to and from FIFOs may be 8-, 16-, or 32-bit as required, and any combination of access is allowed provided the data accessed is contiguous. However, all the transfers associated with one packet must be of the same width so that the data is consistently byte-, half-word- or full-word-aligned. The last transfer may however contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-half-word transfer.

Table 30-28. FIFO Address Assignment

Endpoint	FIFO Address
0	0x50000120
1	0x50000124
2	0x50000128
3	0x5000012C

Part V. Reference

31. Instruction Set

With the exception of the FPU, all instruction processing in the PPC440EP is handled by the PPC440 processor. Refer to the *PPC440 Processor User's Manual* for details.

32. Floating Point Instruction Set

Descriptions of the PPC440EP floating point instructions follow. Each description contains the following elements:

- Instruction names (mnemonic and full)
- Instruction syntax
- Instruction format diagram
- Pseudocode description
- Prose description
- Registers altered

Where appropriate, instruction descriptions list exceptions and invalid instruction forms, and provide programming notes.

Table 32-1 summarizes the PPC440EP instruction set by category.

Table 32-1. Instruction Categories

Category	Subcategory	Instruction Types
Floating Point	Computational	Elementary arithmetic, multiply-add, rounding and converting, square root and reciprocal estimate
	Noncomputational	Load/store, move, compare, Floating-Point Status and Control Register

32.1 Instruction Set Portability

To support embedded real-time applications, the PPC440EP implements the defined floating point instruction set of the Book-E Enhanced PowerPC Architecture, with the exception the **fctid**, **fsqrt**, and **fsqrts** instruction.

The Book-E Enhanced PowerPC Architecture defines some instructions that have record forms, often called "dot forms,' that update the CR1 field of the Condition Register (CR). The record forms of these instructions are not implemented.

32.2 Instruction Formats

For more detailed information about instruction formats, including a summary of instruction field usage and instruction format diagrams for the PPC440EP, see *Floating Point Instruction Summary* on page 831.

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions also have an extended opcode field. The remaining instruction bits are contained in additional fields. All instruction fields belong to one of the following categories:

- Defined

  These instruction fields contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

These fields contain operands, such as general purpose register specifiers and immediate values, each of which may contain any one of a number of values. The instruction format diagrams specify the field names of variable fields.

- Reserved

  Bits in a reserved field should be set to 0. In the instruction format diagrams, reserved fields are shaded.

If any bit in a defined field does not contain the specified value, the instruction is illegal and an Illegal Instruction exception type Program interrupt occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid and its result is architecturally undefined. Unless otherwise noted, the PPC440EP executes all invalid instruction forms without causing an Illegal Instruction exception.

32.3 Pseudocode

The pseudocode that appears in the instruction descriptions provides a semi-formal language for describing instruction operations.

The pseudocode uses the following notation:

+	Twos complement addition
%	Remainder of an integer division; (33 % 32) = 1.
$<^u$, $>^u$	Unsigned comparison relations
(FPR(r))	The contents of FPR r, where $0 \le r \le 31$.
(FRx)	The contents of an FPR, where *x* is A, B, C, S, or T
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
0bn	A binary number
0xn	A hexadecimal number
<, >	Signed comparison relations
=	Assignment
=, ≠	Equal, not equal relations
CEIL(x)	Least integer ≥ x.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies an location in main storage.
EXTS(x)	The result of extending *x* on the left with sign bits.
FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
FRx	An FPR, where *x* is A, B, C, S, or T
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.

GPRs	RA, RB, . . .
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
PC	Program counter.
Rx	A GPR, where *x* is A, B, S, or T
REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
REG[FLD]	A field in a named register
REG_b	A bit in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
$REG_{b:b}$	A range of bits in a named register
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
SPR(SPRN)	A Special Purpose Register (SPR) specified by the SPRF field in an **mfspr** or **mtspr** instruction
$c_{0:3}$	A four-bit object used to store condition results in compare instructions.
do	Do loop. "to" and "by" clauses specify incrementing an iteration variable; "while" and "until" clauses specify terminating conditions. Indenting indicates the scope of a loop.
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.
leave	Leave innermost do loop or do loop specified in a leave statement.
n	A decimal number
^nb	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
||	Concatenation
×	Multiplication
÷	Division yielding a quotient
⊕	Exclusive-OR (XOR) logical operator
–	Twos complement subtraction, unary minus
¬	NOT logical operator
∧	AND logical operator
∨	OR logical operator

32.3.1 Operator Precedence

Table 32-2 lists the pseudocode operators and their associativity in descending order of precedence:

Table 32-2. Operator Precedence

Operators	Associativity
REG_b, REG[FLD], function evaluation	Left to right
n_b	Right to left
¬, – (unary minus)	Right to left
×, ÷	Left to right
+, –	Left to right
||	Left to right
=, ≠, <, >, $\overset{u}{<}$, $\overset{u}{>}$	Left to right
∧, ⊕	Left to right
∨	Left to right
←	None

32.4 Register Usage

Each instruction description lists the registers altered by the instruction. Some register changes are explicitly detailed in the instruction description (for example, the target register of a load instruction). Some instructions also change other registers, but the details of the changes are not included in the instruction descriptions. Common examples of these kinds of register changes include the Floating-Point Registers (FPRs) and the Floating-Point Status and Control Register (FPSCR). For discussion of the FPRs, see *Floating-Point Registers (FPR0:FPR31)* on page 162. For discussion of the FPSCR, see *Floating-Point Status and Control Register (FPSCR)* on page 162.

32.5 Floating-Point Instructions

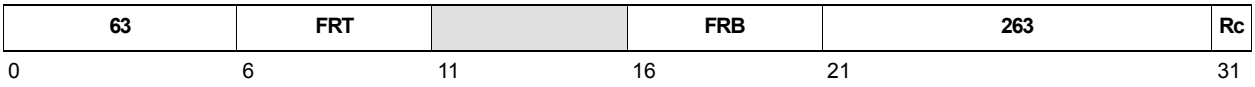
Primary opcode 63 is used for the double-precision arithmetic instructions and miscellaneous instructions (for example, the *Floating-Point Status and Control Register Manipulation* instructions). Primary opcode 59 is used for the single-precision arithmetic instructions.

The single-precision instructions for which there is a corresponding double-precision instruction have the same format and extended opcode as that double-precision instruction.

32.6 Alphabetical Instruction Listing

The following pages list the defined floating point instructions implemented in the PPC440EP.

fabs FRT, FRB Rc = 0



(FRT) ← 0 || $(FRB)_{1:63}$

The contents of FRB, with bit 0 set to zero, are placed into FRT.

Registers Altered

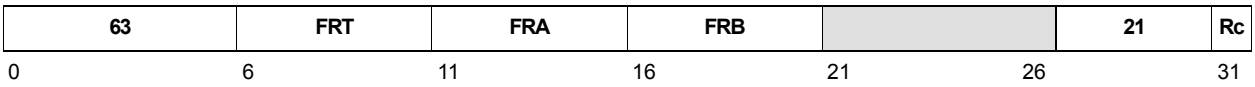
- FRT

Exceptions

An attempt to execute **fabs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fadd FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) + (FRB)

The floating-point operand in FRA is added to the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN] and placed into FRT.

Floating-point addition is based on exponent comparison and addition of the two significands. The exponents of the two operands are compared, and the significand accompanying the smaller exponent is shifted right, with its exponent increased by one for each bit shifted, until the two exponents are equal. The two significands are then added or subtracted as appropriate, depending on the signs of the operands, to form an intermediate sum. All 53 bits of the significand and the three guard bits (G, R, X) enter into the computation.

If a carry occurs, the significand of the sum is shifted right one bit position and the exponent is increased by one.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

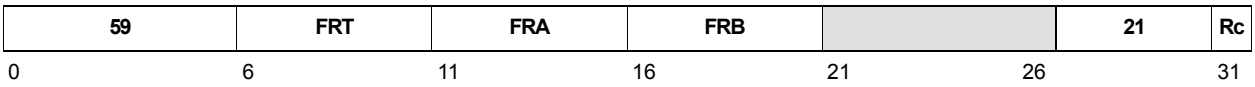
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]

Exceptions

An attempt to execute **fadd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fadds FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) $+_{sp}$ (FRB)

The floating-point operand in FRA is added to the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN] and placed into FRT.

Floating-point addition is based on exponent comparison and addition of the two significands. The exponents of the two operands are compared, and the significand accompanying the smaller exponent is shifted right, with its exponent increased by one for each bit shifted, until the two exponents are equal. The two significands are then added or subtracted as appropriate, depending on the signs of the operands, to form an intermediate sum. All 53 bits of the significand and the three guard bits (G, R, X) enter into the computation.

If a carry occurs, the significand of the sum is shifted right one bit position and the exponent is increased by one.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

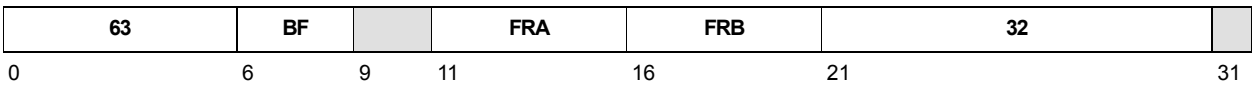
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]

Exceptions

An attempt to execute **fadds** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fcmpo BF, FRA, FRB



if (FRA) is a NaN or
(FRB) is a NaN then c ← 0b0001
else if (FRA) < (FRB) then c ← 0b1000
else if (FRA) > (FRB) then c ← 0b0100
else c← 0b0010
FPSCR[FPCC] ← c
$CR_{4 \times BF:4 \times BF + 3}$ ← c
if (FRA) is a SNaN or (FRB) is a SNaN then do
    if FPSCR[VE] = 0 then FPSCR[VXVC] ← 1
else if (FRA) is a QNaN or (FRB) is a QNaN then FPSCR[VXVC] ← 1

The floating-point operand in FRA is compared to the floating-point operand in FRB. The result of the compare is placed into the CR field BF and FPSCR[FPCC]. The comparison ignores the sign of zero (that is, +0 is considered equal to –0).

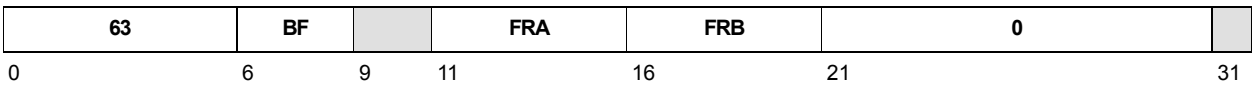
If (FRA) or (FRB) is a NaN, either quiet or signaling, the CR field BF and FPSCR[FPCC] are set to reflect unordered.

If (FRA) or (FRB) is a Signaling NaN and Invalid Operation is disabled (FPSCR[VE] = 0), FPSCR[VXVC] = 1. If neither (FRA) nor (FRB) is a Signaling NaN, but at least one operand is a Quiet NaN, FPSCR[VXVC] = 1.

Registers Altered

- CR[BF]
- FPSCR[FPCC, FX, VXSNAN, VXVC]

fcmpu BF, FRA, FRB



if (FRA) is a NaN or
(FRB) is a NaN then c ← 0b0001
else if (FRA) < (FRB) then c ← 0b1000
else if (FRA) > (FRB) then c ← 0b0100
else c← 0b0010
FPSCR[FPCC] ← c
$CR_{4 \times BF:4 \times BF+3}$ ← c
if ((FRA) is a SNaN or (FRB) is a SNaN) then FPSCR[VXSNAN] ← 1

The floating-point operand in FRA is compared to the floating-point operand in FRB. The result of the compare is placed into the CR field BF and FPSCR[FPCC]. The comparison ignores the sign of zero (that is, +0 is considered equal to –0).

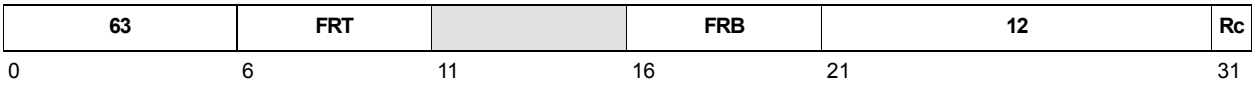
If (FRA) or (FRB) is a NaN, either quiet or signaling, the CR field BF and FPSCR[FPCC] are set to reflect unordered.

If either (FRA) or (FRB) is a Signaling NaN, FPSCR[VXSNAN] = 1.

Registers Altered

- CR
- FPSCR[FPCC, FX, VXSNAN]

fctiw FRT, FRB Rc = 0



The floating-point operand in FRB is converted to a 32-bit signed integer, using the rounding mode specified by FPSCR[RN], and placed into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.

If (FRB) is greater than 2^{31}–1, $FRT_{32:63}$ are set to 0x7FFFFFFF. If (FRB) is less than -2^{31}, $FRT_{32:63}$ are set to 0x80000000.

Except for enabled Invalid Operation Exceptions, FPSCR[FPRF] is undefined. FPSCR[FR] is set if the result is incremented when rounded. FPSCR[FI] is set if the result is inexact.

Registers Altered

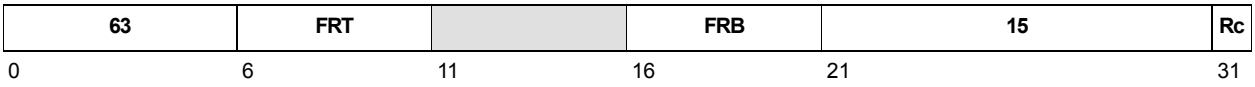
- FRT
- FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI]
- FPSCR[FPRF] undefined

Exceptions

An attempt to execute **fctiw** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fctiwz FRT, FRB Rc = 0



The floating-point operand in FRB is converted to a 32-bit signed integer, using the *Round toward Zero* rounding mode, and placed into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.

If (FRB) is greater than 2^{31}–1, $FRT_{32:63}$ are set to 0x7FFFFFFF. If (FRB) is less than -2^{31}, $FRT_{32:63}$ are set to 0x80000000.

Except for enabled Invalid Operation Exceptions, FPSCR[FPRF] is undefined. FPSCR[FR] is set if the result is incremented when rounded. FPSCR[FI] is set if the result is inexact.

Registers Altered

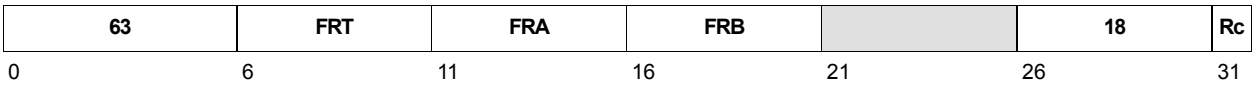
- FRT
- FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI]
- FPSCR[FPRF] undefined

Exceptions

An attempt to execute **fctiwz** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fdiv FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) ÷ (FRB)

The floating-point operand in FRA is divided by the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

The remainder is not supplied as a result.

Floating-point division is based on exponent subtraction and division of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

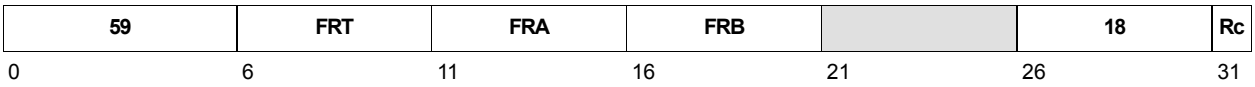
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]

Exceptions

An attempt to execute **fdiv** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fdivs FRT, FRA, FRB Rc = 0



FRT ← (FRA) ÷ (FRB)

The floating-point operand in FRA is divided by the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into (FRT).

The remainder is not supplied as a result.

Floating-point division is based on exponent subtraction and division of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

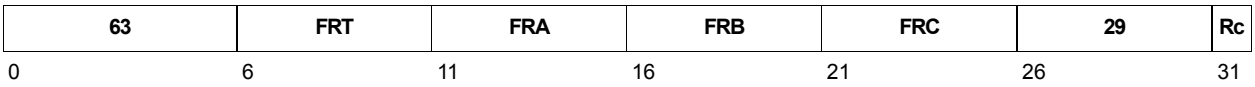
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]

Exceptions

An attempt to execute **fdivs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmadd FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] + (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

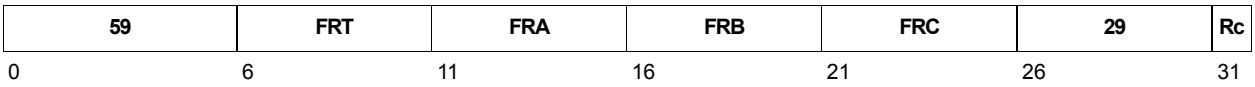
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmadd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmadds FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] + (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

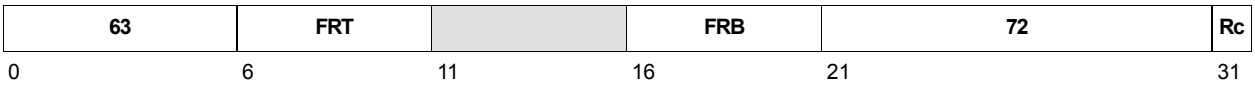
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmadds** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmr FRT, FRB Rc = 0



(FRT) ← FR(FRB)

The contents of FRB are placed into FRT.

Registers Altered

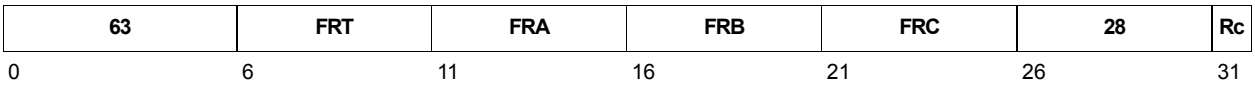
- FRT

Exceptions

An attempt to execute **fmr** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmsub FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] – (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

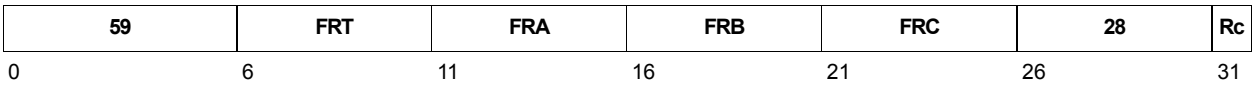
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmsub** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmsubs FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] – (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation exceptions when FPSCR[VE] = 1.

Registers Altered

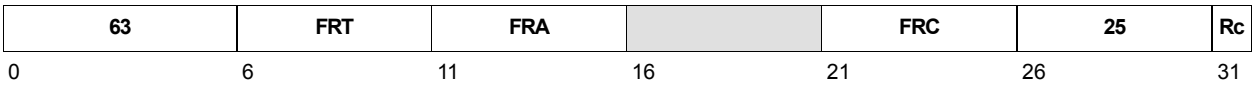
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmsubs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmul FRT, FRA, FRC Rc = 0



(FRT) ← (FRA) × (FRC)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

Floating-point multiplication is based on exponent addition and multiplication of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation exceptions when FPSCR[VE] = 1.

Registers Altered

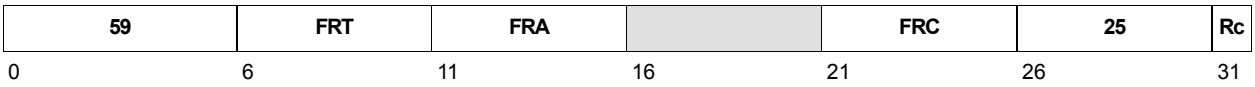
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]

Exceptions

An attempt to execute **fmul** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmuls FRT, FRA, FRC Rc = 0



(FRT) ← (FRA) × (FRC)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into FRT.

Floating-point multiplication is based on exponent addition and multiplication of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation exceptions when FPSCR[VE] = 1.

Registers Altered

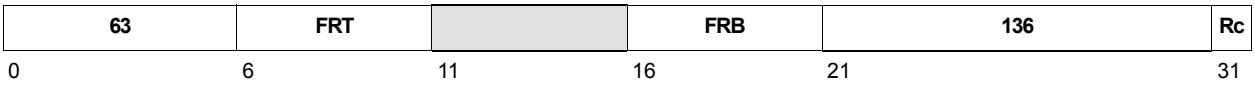
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]

Exceptions

An attempt to execute **fmuls** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnabs FRT, FRB Rc = 0



$(FRT) \leftarrow 1 \,||\, (FRB)_{1:63}$

The contents of FRB, with bit 0 set to one, are placed into FRT.

Registers Altered

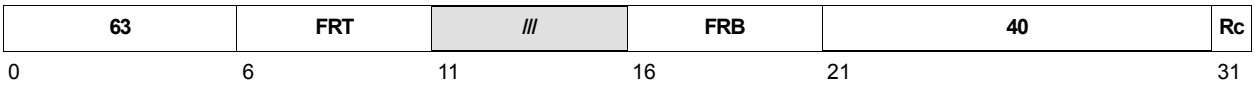
- FRT

Exceptions

An attempt to execute **fnabs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fneg FRT, FRB Rc = 0



$(FRT) \leftarrow \neg(FRB)_0 \,||\, (FRB)_{1:63}$

The contents of FRB, with bit 0 inverted, are placed into FRT.

Registers Altered

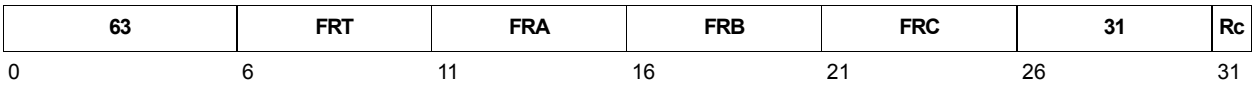
- FRT

Exceptions

An attempt to execute **fneg** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmadd FRT, FRA, FRC, FRB Rc = 0



FRT ← –([(FRA) × (FRC)] + (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN], then negated and placed into (FRT).

This instruction produces the same result as would be obtained by using the **fmadd** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their "sign" (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a "sign" bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the "sign" bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

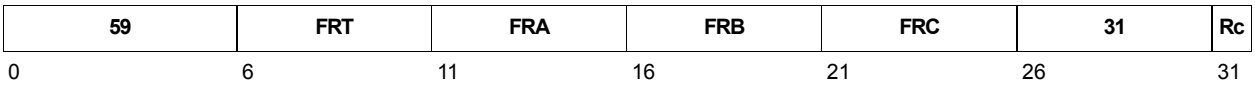
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmadd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmadds FRT, FRA, FRC, FRB Rc = 0



(FRT) ← –([(FRA) × (FRC)] + (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN], then negated and placed into FRT.

This instruction produces the same result as would be obtained by using the **fmadds** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their "sign" (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a "sign" bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the "sign" bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

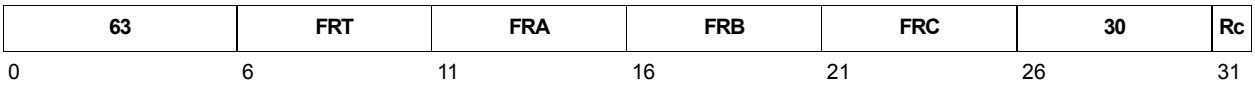
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmadds** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmsub FRT, FRA, FRC, FRB Rc = 0



(FRT) ← –([(FRA) × (FRC)] – (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN], then negated and placed into FRT.

This instruction produces the same result as would be obtained by using the **fmsub** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their ”sign” (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a “sign” bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the “sign” bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

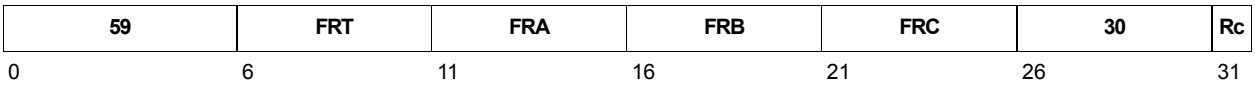
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmsub** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmsubs FRT, FRA, FRC, FRB Rc = 0



(FRT) ← –([(FRA) × (FRC)] – (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN], then negated and placed into FRT.

This instruction produces the same result as would be obtained by using the **fmsubs** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their ”sign” (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a “sign” bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the “sign” bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

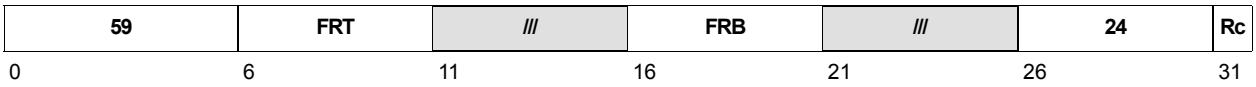
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmsubs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fres FRT, FRB (Rc=0)



(FRT) ← FPReciprocalEstimate(FRB)

A single-precision estimate of the reciprocal of the floating-point operand in FRB is placed into FRT. The estimate is correct to a precision of one part in 2^{13} of the reciprocal of (FRB), that is,

$$\left| \frac{\text{estimate} - \frac{1}{x}}{\frac{1}{x}} \right| \leq \frac{1}{2}13$$

where *x* is the initial value of (FRB).

Table 32-3 summarizes operation with operands having various special values.

*Table 32-3. **fres** Operation with Special Operand Values*

Operand	Result	Exception
–∞	–0	—
–0	–∞[1]	ZX
+0	+∞[1]	ZX
+∞	+0	—
SNaN	QNaN[2]	VXSNAN
QNaN	QNaN	—

Note: 1. No result if FPSCR[ZE] = 1
2. No result if FPSCR[VE] = 1.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

- FRT
- FPSCR[FX, OX, UX, ZX, VXSNAN]
- FPSCR[FPRF, FR, FI] undefined

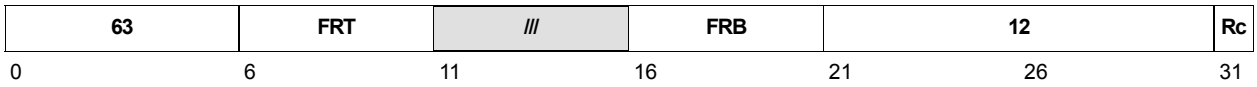
Exceptions

An attempt to execute **fres** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Architecture Notes

The value placed into FRT may vary between implementations, and between different executions on the same implementation.

frsp FRT, FRB (Rc=0)



The floating-point operand in FRB is rounded to single-precision, using the rounding mode specified by FPSCR[RN], and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

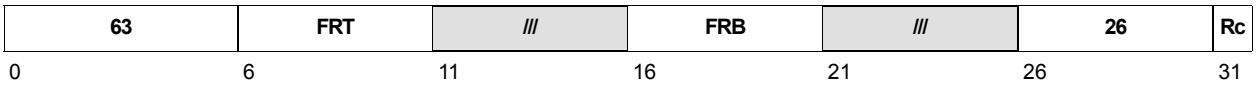
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN]

Exceptions

An attempt to execute **frsp** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

frsqrte FRT, FRB (Rc=0)



(FRT) ← FPReciprocalSquareRootEstimate(FRB)

A double-precision estimate of the reciprocal of the square root of the floating-point operand in FRB is placed into FRT. The estimate placed is correct to a precision of one part in 2^{13} of the reciprocal of the square root of (FRB), that is,

$$\left| \frac{\left(\text{estimate} - \frac{1}{\sqrt{x}}\right)}{\frac{1}{\sqrt{x}}} \right| \leq \frac{1}{2}13$$

where *x* is the initial value in FPR(FRB). Note that the value placed into FPR(FRT) may vary between implementations, and between different executions on the same implementation.

Table 32-4 summarizes operation with various special values of the operand.

*Table 32-4. **frsqrte** Operation with Special Operand Values*

Operand	Result	Exception
–∞	QNaN[2]	VXSQRT
< 0	QNaN[2]	VXSQRT
–0	–∞[1]	ZX
+0	+∞[1]	ZX
+∞	+0	—
SNaN	QNaN[2]	VXSNAN
QNaN	QNaN	—

Note: 1. No result if FPSCR[ZE] = 1
2. No result if FPSCR[VE] = 1.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

- FRT
- FPSCR[FX, ZX, VXSNAN, VXSQRT]
- FPSCR[FPRF, FR, FI] undefined

Exceptions

An attempt to execute **frsqrte** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

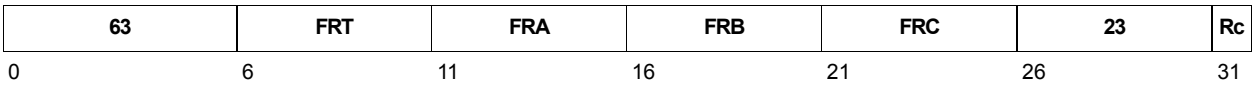
Architecture Notes

No single-precision version of this instruction is provided. If the floating-point operand in FRB is representable in single format, so is the resulting value of FRT.

Implementation Note

The precision of this estimate in this implementation is higher than that specified in Book-E (2^{-5}). Therefore, code relying on this higher precision may not work on other PowerPC implementations.

fsel FRT, FRA, FRC, FRB (Rc=0)



if (FRA) ≥ 0.0 then (FRT) ← (FRC)

else (FRT) ← (FRB)

The floating-point operand in FRA is compared to zero. If the operand is greater than or equal to 0, FRT is set to the contents of FRC. If the operand is less than 0 or is a NaN, FRT is set to the contents of FRB. The comparison ignores the sign of zero (that is, +0 is considered equal to –0).

Registers Altered

- FRT

Exceptions

An attempt to execute fsel[.] while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Programming Notes

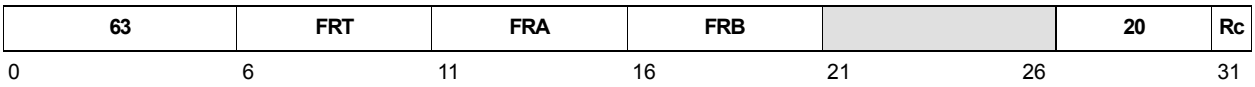
Care must be taken in using fsel if compatibility with IEEE 754 is required, or if the values being tested can be NaNs or infinities. See XREF TBD.

Examples of uses of this instruction can be found in XREF TBD.

Architecture Notes

The *Select* instruction is similar to a *Move* instruction, and therefore does not alter FPSCR.

fsub FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) – (FRB)

The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

The operation of **fsub** is identical to that of **fadd**, except that the contents of FRB participate in the operation with the sign bit (bit 0) inverted.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

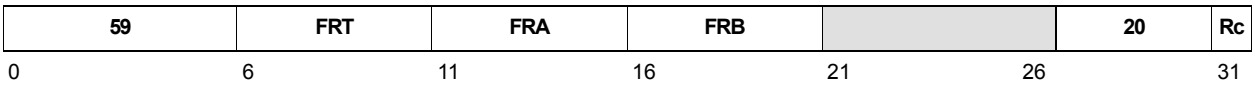
- FRT
- FPSRP[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN]

Exceptions

An attempt to execute **fsub** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fsubs FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) – (FRB)

The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into FRT.

The operation of **fsubs** is identical to that of **fadds**, except that the contents of FRB participate in the operation with the sign bit (bit 0) inverted.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

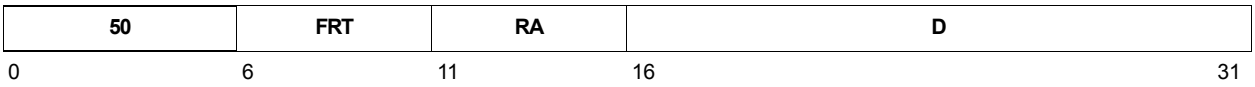
- FRT
- FPSCR[FPRF, FR, FI FX, OX, UX, XX, VXSNAN]

Exceptions

An attempt to execute **fsubs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

lfd FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← MEM(EA,8)

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The doubleword in storage addressed by EA is placed into FRT.

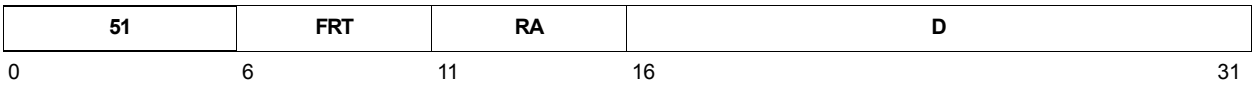
Registers Altered

- FRT

Exceptions

An attempt to execute **lfd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

lfdu FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← MEM(EA,8)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The doubleword in storage addressed by EA is placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

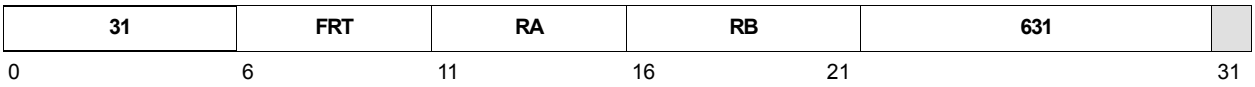
Exceptions

An attempt to execute **lfdu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Invalid Forms

RA = 0

lfdux FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← MEM(EA,8)
RA ← EA

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The doubleword in storage addressed by EA is placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

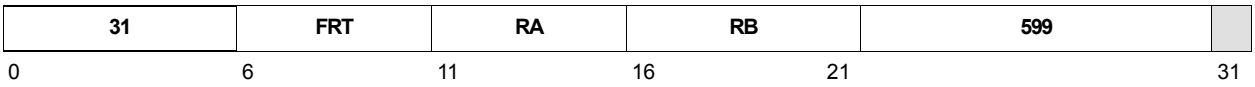
Exceptions

An attempt to execute **lfdux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Invalid Forms

RA = 0

lfdx FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← MEM(EA,8)

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The doubleword in storage addressed by EA is placed into FRT.

Registers Altered

- FRT

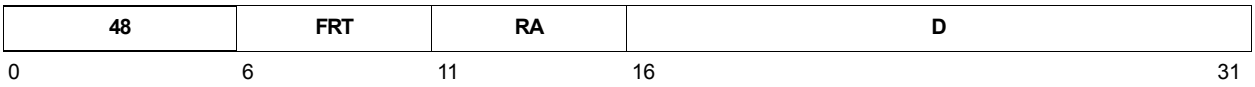
Exceptions

An attempt to execute **lfdx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfs FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← DOUBLE(MEM(EA,4))

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 169) and placed into FRT.

Registers Altered

- FRT

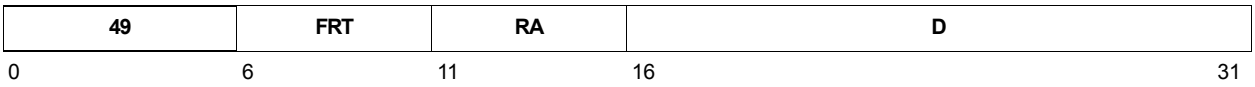
Exceptions

An attempt to execute **lfs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfsu FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← DOUBLE(MEM(EA,4))
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 169) and placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

Exceptions

An attempt to execute **lfsu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

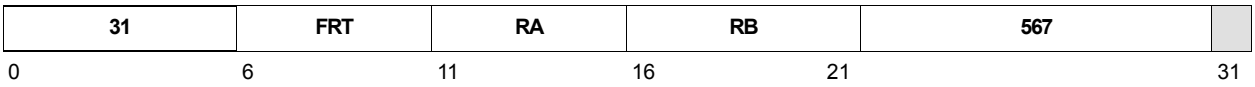
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfsux FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← DOUBLE(MEM(EA,4))
RA ← EA

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 169) and placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

Exceptions

An attempt to execute **lfsux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

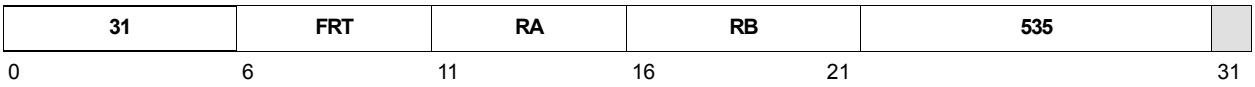
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfsx FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← DOUBLE(MEM(EA,4))

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 169) and placed into FRT.

Registers Altered

- FRT

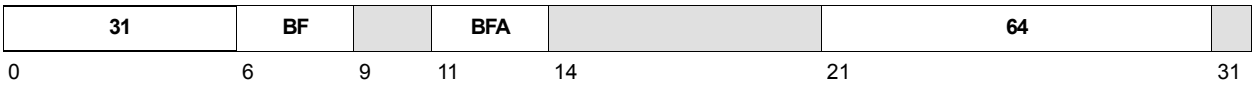
Exceptions

An attempt to execute **lfsx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

mcrfs BF, BFA



$CR_{BF \times 4:BF \times 4+3} \leftarrow FPSCR_{BFA \times 4:BFA \times 4 + 3}$

$FPSCR_{BFA \times 4:BFA \times 4 + 3} \leftarrow 0b0000$

The contents of the FPSCR specified by BFA are copied to the CR field specified by BF. All exception bits that are copied are set to 0 in the FPSCR. If FPSCR[FX] is copied, it is set to 0 in the FPSCR.

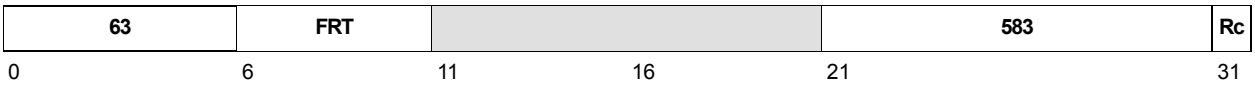
Registers Altered

- CR field BF
- FPSCR[FX, OX] if BFA=0
- FPSCR[UX, ZX, XX, VXSNAN] if BFA=1
- FPSCR[VXISI, VXIDI, VXZDZ, VXIMZ] if BFA=2
- FPSCR[VXVC] if BFA=3
- FPSCR[VXSOFT, VXSQRT, VXCVI] if BFA=5

Exceptions

An attempt to execute **mcrfs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

mffs FRT Rc = 0



(FRT) ← FPSCR

The contents of the FPSCR are placed into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.

Registers Altered

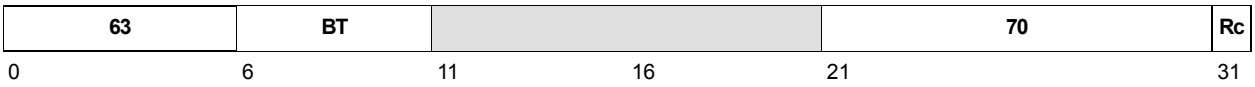
- FRT

Exceptions

An attempt to execute **mffs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

mtfsb0 BT Rc = 0



$FPSCR_{BT} \leftarrow 0$

Bit BT of the FPSCR is set to 0.

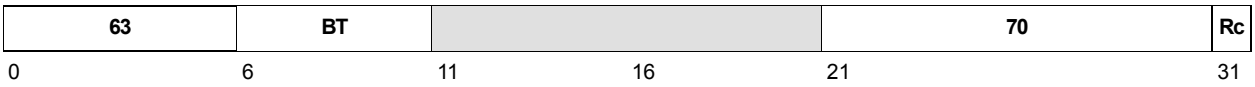
Registers Altered

- FPSCR[BT]

An attempt to execute **mtfsb0** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

mtfsb1 BT Rc = 0



$FPSCR_{BT} \leftarrow 1$

Bit BT of the FPSCR is set to 1.

Registers Altered

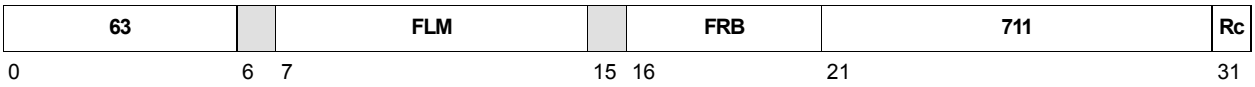
- FPSCR[BT, FX]

AIf Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

Programming Notes

FPSCR[FEX, VX] cannot be explicitly set.

mtfsf FLM, FRB Rc = 0



i ← 0
do while i<8
  if FLM_i=1 then $FPSCR_{4 \times i:4 \times i + 3} \leftarrow (FRB)_{4 \times i:4 \times i + 3}$
  i ← i+1

The contents of bits 32:63 of FRB are placed into the FPSCR under control of the field mask specified by FLM. The field mask identifies the affected 4-bit fields. Let *i* be an integer in the range 0–7. If FLM_i = 1, FPSCR field *i* (FPSCR bits $4 \times i - 4 \times i + 3$) is set to the contents of the corresponding field of the low-order 32 bits of FRB.

FPSCR[FX] is altered only if FLM_0 = 1.

Registers Altered

- FPSCR fields selected by mask

Exceptions

An attempt to execute **mtfsf** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

Programming Notes

Updating fewer than all eight fields of the FPSCR may have substantially poorer performance on some implementations than updating all the fields.

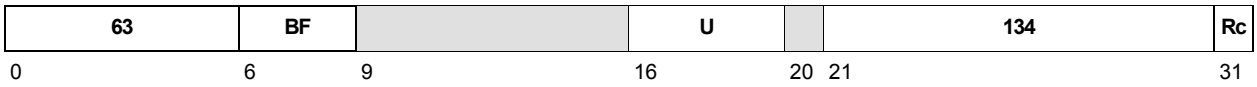
When $FPSCR_{0:3}$ is specified, bits 0 and 3 (FPSCR[FX, OX]) are set to the values of FRB_{32} and FRB_{35}. Even if **mtfsf** causes FPSCR[OX] to change from 0 to 1, FPSCR[FX] is set from FRB_{32}, not by the usual rule that FPSCR[FX] is set to 1 when an exception bit changes from 0 to 1. Bits 1 and 2 (FPSCR[FEX, VX]) are set according to the usual rule, given on XREF TBD, and not from $FRB_{33:34}$.

Implementation Note

In general, the source FPR should contain data produced by a double-precision load operation. The value moved to FPSCR is unpredictable if the source FPR contains denormalized single format data (that is, data that cannot be represented as a normalized single-precision number that was produced from a single precision operation.

This is similar to the achitectural rule for **stfiwx**.

mtfsfi BF, U Rc = 0



$FPSCR_{BF \times 4:BF \times 4 + 3} \leftarrow U$

The value of the U field is placed into FPSCR field BF.

$FPSCR_{FX}$ is altered only if BF = 0.

Registers Altered

- $FPSCR_{BF}$

Exceptions

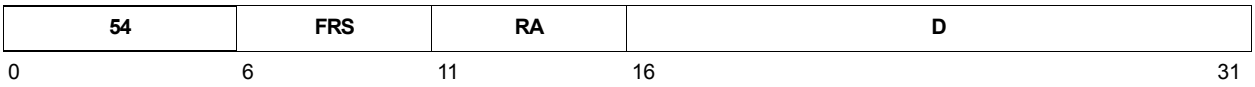
An attempt to execute **mtfsfi** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

Programming Notes

When $FPSCR_{0:3}$ is specified, bits 0 and 3 FPSCR[FX, OX] are set to the values of U_0 and U_3. Even if **mtfsfi** causes FPSCR[OX] to change from 0 to 1, FPSCR[FX] is set from U_0 and not by the usual rule that FPSCR[FX] is set to 1 when an exception bit changes from 0 to 1). Bits 1 and 2 (FPSCR[FEX, VX]) are set according to the usual rule, given on XREF TBD, and not from $U_{1:2}$.

stfd FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,8) ← (FRS)

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are stored into the doubleword in storage addressed by EA.

Registers Altered

- None

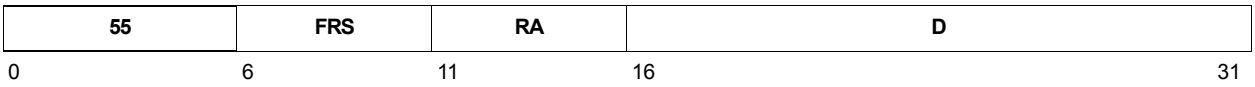
Exceptions

An attempt to execute **stfd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfdu FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,8) ← (FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are stored into the doubleword in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

An attempt to execute **stfdu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

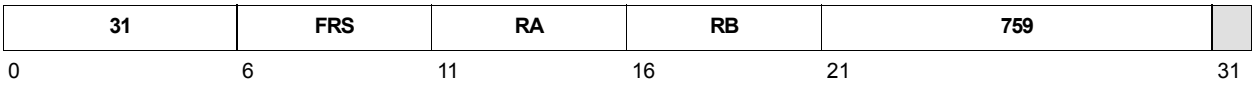
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfdux FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,8) ← (FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are stored into the doubleword in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

An attempt to execute **stfdux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

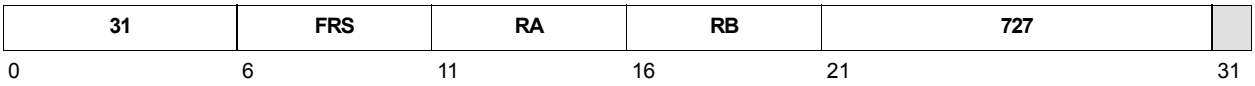
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfdx FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,8) ← (FRS)

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are stored into the doubleword in storage addressed by EA.

Registers Altered

- None

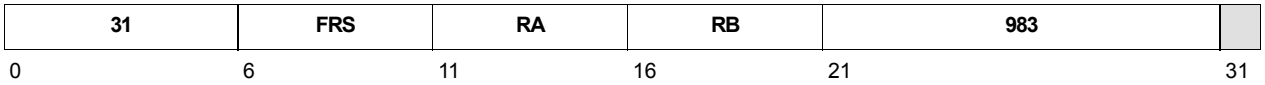
Exceptions

An attempt to execute **stfdx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfiwx FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,4) ← $(FRS)_{32:63}$

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of bits 32:63 of FRS are stored, without conversion, into the word in storage addressed by EA.

If the contents of FRS were produced, either directly or indirectly, by a *Load Floating-Point Single* instruction, a single-precision *Arithmetic* instruction, or **frsp**, the value stored is undefined. (The contents of FRS are produced directly by such an instruction if FRS is the target register for the instruction. The contents of FRS are produced indirectly by such an instruction if FRS is the final target register of a sequence of one or more *Floating-Point Move* instructions, with the input to the sequence having been produced directly by such an instruction.)

Registers Altered

- None

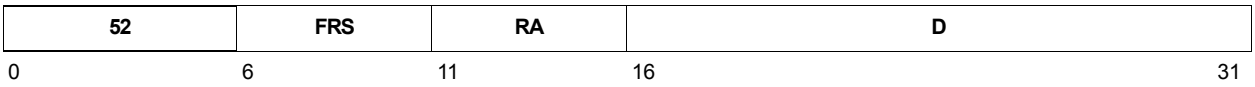
Exceptions

An attempt to execute **stfiwx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfs FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,4) ← SINGLE(FRS)

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are converted to single-precision format (see *Data Handling and Precision* on page 169) and stored into the word in storage addressed by EA.

Registers Altered

- None

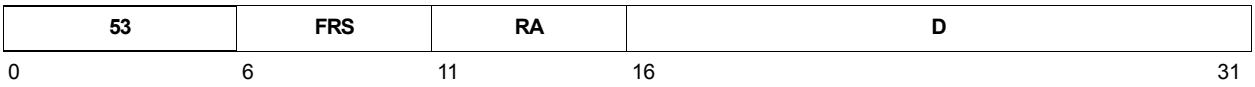
Exceptions

An attempt to execute **stfs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfsu FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,4) ← SINGLE(FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are converted to single-precision format (see *Data Handling and Precision* on page 169) and stored into the word in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

An attempt to execute **stfsu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

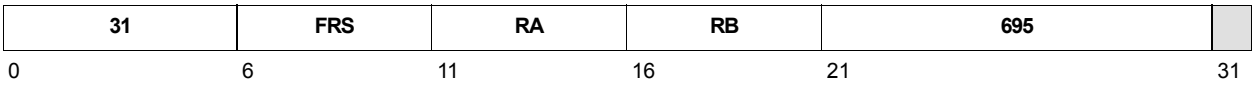
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfdux FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,4) ← SINGLE(FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are converted to single-precision format (see *Data Handling and Precision* on page 169) and stored into the word in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

An attempt to execute **stfsux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

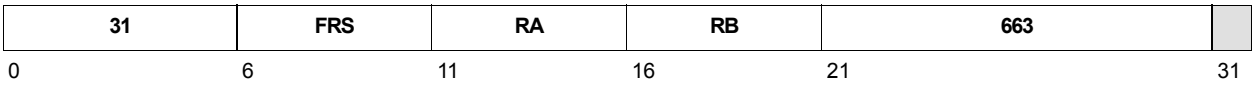
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfsx FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,4) ← SINGLE(FRS)

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are converted to single-precision format (see*Data Handling and Precision* on page 169) and stored into the word in storage addressed by EA.

Registers Altered

- None

Exceptions

An attempt to execute **stfsx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

33. Register Summary

This chapter provides an alphabetical listing and bit definitions for the registers provided by the PPC440EP that are not associated with the PPC440 processor. These registers include the DCRs, FPRs, MMIO registers, and all registers provided by the peripheral cores.

33.1 Reserved Fields

For all registers with fields marked as reserved, the reserved fields should be written as *zero* and read as *undefined*. That is, when writing to a reserved field, write a zero to that field. When reading from a reserved field, ignore that field.

The recommended coding practice is to perform the initial write to a register with reserved fields as described in the preceding paragraph, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register, alter desired fields with logical instructions, and then write the register.

33.2 Device Control Registers

DCRs may be used to control various on-chip system functions, such as the operation of on-chip buses, peripherals, and certain processor core behaviors. The DCR access instructions are **mtdcr** (move to device control register) and **mfdcr** (move from device control register), which move data between GPRs and the DCRs. Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR.

See *Table 4-3* on page 139 for a complete listing of the PPC440EP Device Control registers.

33.3 Memory Mapped Registers

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses.

See *Table 4-4* on page 149 for a complete listing of the PPC440EP Memory Mapped registers.

33.4 Alphabetical Listing of Chip Control and Peripheral Core Registers

The following pages list all the device control registers (DCRs), floating-point registers (FPRs), and memory mapped registers (MMIOs) implemented in the PPC440EP. The registers implemented for the PPC440 processor core are not included in this list (refer to the *PPC400 Processor User's Manual*). For each register, the following information is supplied:

- Register mnemonic
- Register type
- Register description
- Register number or address
- Access (Read/Clear, Read-Only, Read/Write, Write-Only)
- Cross-reference to detailed register information

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers

Register	Type	Description	Address or Number	Access	See Page
CPM0_ER	DCR	CPM Enable Register	0x00B0	R/W	301
CPM0_FR	DCR	CPM Force Register	0x00B1	R/W	303
CPM0_SR	DCR	CPM Status Register	0x00B2	R/W	304
CPR0_CFGADDR	DCR	Clock, Power-On, Reset Configuration Address Register	0x000C	R/W	294
CPR0_CFGDATA	DCR	Clock, Power-On, Reset Configuration Data Register	0x000D	R/W	294
CPR0_CLKUPD	DCR	Clocking Update Register	offset 0x0020	R/W	295
CPR0_ICFG	DCR	Initial Configuration Register	offset 0x0140	R/W	196
CPR0_MALD	DCR	MAL Clock Divisor Register	offset 0x0100	R/W	299
CPR0_OPBD0	DCR	OPB Clock Divisor Register	offset 0x00C0	R/W	298
CPR0_PERD0	DCR	Peripheral Clock Divisor Register	offset 0x00E0	R/W	298
CPR0_PLLC0	DCR	PLL Control Register	offset 0x0040	R/W	295
CPR0_PLLD0	DCR	PLL Divisor Register	offset 0x0060	R/W	296
CPR0_PRIMAD0	DCR	Primary Divisor Register A	offset 0x0080	R/W	297
CPR0_PRIMBD0	DCR	Primary Divisor Register B	offset 0x00A0	R/W	298
CPR0_SPCID	DCR	Synchronous PCI Clock Divisor Register	offset 0x0120	R/W	299
DMA2P30_ADR	DCR	DMA to PLB 3 Address Decode Register	0x0124	R/W	536
DMA2P30_CR0	DCR	DMA to PLB 3 Channel Control Register 0	0x0100	R/W	532
DMA2P30_CR1	DCR	DMA to PLB 3 Channel Control Register 1	0x0108	R/W	532
DMA2P30_CR2	DCR	DMA to PLB 3 Channel Control Register 2	0x0110	R/W	532
DMA2P30_CR3	DCR	DMA to PLB 3 Channel Control Register 3	0x0118	R/W	532
DMA2P30_CT0	DCR	DMA to PLB 3 Count Register 0	0x0101	R/W	534
DMA2P30_CT1	DCR	DMA to PLB 3 Count Register 1	0x0109	R/W	534
DMA2P30_CT2	DCR	DMA to PLB 3 Count Register 2	0x0111	R/W	534
DMA2P30_CT3	DCR	DMA to PLB 3 Count Register 3	0x0119	R/W	534
DMA2P30_DA0	DCR	DMA to PLB 3 Destination Address Register 0	0x0102	R/W	534
DMA2P30_DA1	DCR	DMA to PLB 3 Destination Address Register 1	0x010A	R/W	534
DMA2P30_DA2	DCR	DMA to PLB 3 Destination Address Register 2	0x0112	R/W	534
DMA2P30_DA3	DCR	DMA to PLB 3 Destination Address Register 3	0x011A	R/W	534
DMA2P30_POL	DCR	DMA to PLB 3 Polarity Configuration Register	0x0126	R/W	530
DMA2P30_SA0	DCR	DMA to PLB 3 Source Address Register 0	0x0103	R/W	534
DMA2P30_SA1	DCR	DMA to PLB 3 Source Address Register 1	0x010B	R/W	534
DMA2P30_SA2	DCR	DMA to PLB 3 Source Address Register 2	0x0113	R/W	534
DMA2P30_SA3	DCR	DMA to PLB 3 Source Address Register 3	0x011B	R/W	534
DMA2P30_SC0	DCR	DMA to PLB 3 Subchannel ID Register 0	0x0107	R/W	536
DMA2P30_SC1	DCR	DMA to PLB 3 Subchannel ID Register 1	0x010F	R/W	536
DMA2P30_SC2	DCR	DMA to PLB 3 Subchannel ID Register 2	0x0117	R/W	536
DMA2P30_SC3	DCR	DMA to PLB 3 Subchannel ID Register 3	0x011F	R/W	536

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
DMA2P30_SG0	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 0	0x0104	R/W	535
DMA2P30_SG1	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 1	0x010C	R/W	535
DMA2P30_SG2	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 2	0x0114	R/W	535
DMA2P30_SG3	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 3	0x011C	R/W	535
DMA2P30_SGC	DCR	DMA to PLB 3 Scatter/Gather Command Register	0x0123	R/W	535
DMA2P30_SLP	DCR	DMA to PLB 3 Sleep Mode Register	0x0125	R/W	531
DMA2P30_SR	DCR	DMA to PLB 3 Status Register	0x0120	R/W	531
DMA2P40_CR0	DCR	DMA to PLB 4 Channel Control Register 0	0x0300	R/W	513
DMA2P40_CR1	DCR	DMA to PLB 4 Channel Control Register 1	0x0308	R/W	513
DMA2P40_CR2	DCR	DMA to PLB 4 Channel Control Register 2	0x0310	R/W	513
DMA2P40_CR3	DCR	DMA to PLB 4 Channel Control Register 3	0x0318	R/W	513
DMA2P40_CT0	DCR	DMA to PLB 4 Count and Control Register 0	0x0301	R/W	514
DMA2P40_CT1	DCR	DMA to PLB 4 Count and Control Register 1	0x0309	R/W	514
DMA2P40_CT2	DCR	DMA to PLB 4 Count and Control Register 2	0x0311	R/W	514
DMA2P40_CT3	DCR	DMA to PLB 4 Count and Control Register 3	0x0319	R/W	514
DMA2P40_DAH0	DCR	DMA to PLB 4 Destination Address High Register 0	0x0304	R/W	516
DMA2P40_DAH1	DCR	DMA to PLB 4 Destination Address High Register 1	0x030C	R/W	516
DMA2P40_DAH2	DCR	DMA to PLB 4 Destination Address High Register 2	0x0314	R/W	516
DMA2P40_DAH3	DCR	DMA to PLB 4 Destination Address High Register 3	0x031C	R/W	516
DMA2P40_DAL0	DCR	DMA to PLB 4 Destination Address Low Register 0	0x0305	R/W	516
DMA2P40_DAL1	DCR	DMA to PLB 4 Destination Address Low Register 1	0x030D	R/W	516
DMA2P40_DAL2	DCR	DMA to PLB 4 Destination Address Low Register 2	0x0315	R/W	516
DMA2P40_DAL3	DCR	DMA to PLB 4 Destination Address Low Register 3	0x031D	R/W	516
DMA2P40_POL	DCR	DMA to PLB 4 Polarity Configuration Register	0x0326	R/W	519
DMA2P40_SAH0	DCR	DMA to PLB 4 Source Address High Register 0	0x0302	R/W	515
DMA2P40_SAH1	DCR	DMA to PLB 4 Source Address High Register 1	0x030A	R/W	515
DMA2P40_SAH2	DCR	DMA to PLB 4 Source Address High Register 2	0x0312	R/W	515
DMA2P40_SAH3	DCR	DMA to PLB 4 Source Address High Register 3	0x031A	R/W	515
DMA2P40_SAL0	DCR	DMA to PLB 4 Source Address Low Register 0	0x0303	R/W	515
DMA2P40_SAL1	DCR	DMA to PLB 4 Source Address Low Register 1	0x030B	R/W	515
DMA2P40_SAL2	DCR	DMA to PLB 4 Source Address Low Register 2	0x0313	R/W	515
DMA2P40_SAL3	DCR	DMA to PLB 4 Source Address Low Register 3	0x031B	R/W	515
DMA2P40_SGC	DCR	DMA to PLB 4 Scatter/Gather Command Register	0x0323	R/W	518
DMA2P40_SGH0	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 0	0x0306	R/W	516
DMA2P40_SGH1	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 1	0x030E	R/W	516
DMA2P40_SGH2	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 2	0x0316	R/W	516
DMA2P40_SGH3	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 3	0x031E	R/W	516
DMA2P40_SGL0	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 0	0x0307	R/W	516

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
DMA2P40_SGL1	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 1	0x030F	R/W	516
DMA2P40_SGL2	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 2	0x0317	R/W	516
DMA2P40_SGL3	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 3	0x031F	R/W	516
DMA2P40_SLP	DCR	DMA to PLB 4 Sleep Mode Register	0x0325	R/W	518
DMA2P40_SR	DCR	DMA to PLB 4 Status Register	0x0320	R/W	517
EBC0_B0AP	DCR	Peripheral Bank 0 Access Parameters	offset 0x0010	R/W	487
EBC0_B0CR	DCR	Peripheral Bank 0 Configuration Register	offset 0x0000	R/W	486
EBC0_B1AP	DCR	Peripheral Bank 1 Access Parameters	offset 0x0011	R/W	487
EBC0_B1CR	DCR	Peripheral Bank 1 Configuration Register	offset 0x0001	R/W	486
EBC0_B2AP	DCR	Peripheral Bank 2 Access Parameters	offset 0x0012	R/W	487
EBC0_B2CR	DCR	Peripheral Bank 2 Configuration Register	offset 0x0002	R/W	486
EBC0_B3AP	DCR	Peripheral Bank 3 Access Parameters	offset 0x0013	R/W	487
EBC0_B3CR	DCR	Peripheral Bank 3 Configuration Register	offset 0x0003	R/W	486
EBC0_B4AP	DCR	Peripheral Bank 4 Access Parameters	offset 0x0014	R/W	487
EBC0_B4CR	DCR	Peripheral Bank 4 Configuration Register	offset 0x0004	R/W	486
EBC0_B5AP	DCR	Peripheral Bank 5 Access Parameters	offset 0x0015	R/W	487
EBC0_B5CR	DCR	Peripheral Bank 5 Configuration Register	offset 0x0005	R/W	486
EBC0_BEAR	DCR	Peripheral Bus Error Address Register	offset 0x0020	R/W	489
EBC0_BESR0	DCR	Peripheral Bus Error Status Register 0	offset 0x0021	R/W	489
EBC0_BESR1	DCR	Peripheral Bus Error Status Register 1	offset 0x0022	R/W	490
EBC0_CFG	DCR	External Peripheral Control Register	offset 0x0023	R/W	485
EBC0_CFGADDR	DCR	Peripheral Bank Address Register	0x0012	R/W	483
EBC0_CFGDATA	DCR	Peripheral Bank Data Register	0x0013	R/W	483
EMAC0_GAHT1	MMIO	EMAC 0 Group Address Hash Table 1	0x0 EF60 0E40	R/W	621
EMAC0_GAHT2	MMIO	EMAC 0 Group Address Hash Table 2	0x0 EF60 0E44	R/W	621
EMAC0_GAHT3	MMIO	EMAC 0 Group Address Hash Table 3	0x0 EF60 0E48	R/W	621
EMAC0_GAHT4	MMIO	EMAC 0 Group Address Hash Table 4	0x0 EF60 0E4C	R/W	621
EMAC0_IAHR	MMIO	EMAC 0 Individual Address High	0x0 EF60 0E1C	R/W	619
EMAC0_IAHT1	MMIO	EMAC 0 Individual Address Hash Table 1	0x0 EF60 0E30	R/W	621
EMAC0_IAHT2	MMIO	EMAC 0 Individual Address Hash Table 2	0x0 EF60 0E34	R/W	621
EMAC0_IAHT3	MMIO	EMAC 0 Individual Address Hash Table 3	0x0 EF60 0E38	R/W	621
EMAC0_IAHT4	MMIO	EMAC 0 Individual Address Hash Table 4	0x0 EF60 0E3C	R/W	621
EMAC0_IALR	MMIO	EMAC 0 Individual Address Low	0x0 EF60 0E20	R/W	620
EMAC0_IPGVR	MMIO	EMAC 0 Inter-Packet Gap Value Register	0x0 EF60 0E58	R/W	622
EMAC0_ISER	MMIO	EMAC 0 Interrupt Status Enable Register	0x0 EF60 0E18	R/W	618
EMAC0_ISR	MMIO	EMAC 0 Interrupt Status Register	0x0 EF60 0E14	R/W	621
EMAC0_LSAH	MMIO	EMAC 0 Last Source Address Low	0x0 EF60 0E50	R	621
EMAC0_LSAL	MMIO	EMAC 0 Last Source Address High	0x0 EF60 0E54	R	621

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
EMAC0_MR0	MMIO	EMAC 0 Mode Register 0	0x0 EF60 0E00	R/W	611
EMAC0_MR1	MMIO	EMAC 0 Mode Register 1	0x0 EF60 0E04	R/W	612
EMAC0_OCRX	MMIO	EMAC 0 Number of Octets Received	0x0 EF60 0E6C	R	625
EMAC0_OCTX	MMIO	EMAC 0 Number of Octets Transmitted	0x0 EF60 0E68	R	625
EMAC0_PTR	MMIO	EMAC 0 Pause Timer Register	0x0 EF60 0E2C	R/W	621
EMAC0_RMR	MMIO	EMAC 0 Receive Mode Register	0x0 EF60 0E10	R/W	615
EMAC0_RWMR	MMIO	EMAC 0 Receive Low/High Water Mark Register	0x0 EF60 0E64	R/W	624
EMAC0_STACR	MMIO	EMAC 0 STA Control Register	0x0 EF60 0E5C	R/W	622
EMAC0_TMR0	MMIO	EMAC 0 Transmit Mode Register 0	0x0 EF60 0E08	R/W	613
EMAC0_TMR1	MMIO	EMAC 0 Transmit Mode Register 1	0x0 EF60 0E0C	R/W	614
EMAC0_TRTR	MMIO	EMAC 0 Transmit Request Threshold Register	0x0 EF60 0E60	R/W	623
EMAC0_VTCI	MMIO	EMAC 0 VLAN TCI Register	0x0 EF60 0E28	R/W	620
EMAC0_VTPID	MMIO	EMAC 0 VLAN TPID Register	0x0 EF60 0E24	R/W	620
EMAC1_GAHT1	MMIO	EMAC 1 Group Address Hash Table 1	0x0 EF60 0F40	R/W	621
EMAC1_GAHT2	MMIO	EMAC 1 Group Address Hash Table 2	0x0 EF60 0F44	R/W	621
EMAC1_GAHT3	MMIO	EMAC 1 Group Address Hash Table 3	0x0 EF60 0F48	R/W	621
EMAC1_GAHT4	MMIO	EMAC 1 Group Address Hash Table 4	0x0 EF60 0F4C	R/W	621
EMAC1_IAHR	MMIO	EMAC 1 Individual Address High	0x0 EF60 0F1C	R/W	619
EMAC1_IAHT1	MMIO	EMAC 1 Individual Address Hash Table 1	0x0 EF60 0F30	R/W	621
EMAC1_IAHT2	MMIO	EMAC 1 Individual Address Hash Table 2	0x0 EF60 0F34	R/W	621
EMAC1_IAHT3	MMIO	EMAC 1 Individual Address Hash Table 3	0x0 EF60 0F38	R/W	621
EMAC1_IAHT4	MMIO	EMAC 1 Individual Address Hash Table 4	0x0 EF60 0F3C	R/W	621
EMAC1_IALR	MMIO	EMAC 1 Individual Address Low	0x0 EF60 0F20	R/W	620
EMAC1_IPGVR	MMIO	EMAC 1 Inter-Packet Gap Value Register	0x0 EF60 0F58	R/W	622
EMAC1_ISER	MMIO	EMAC 1 Interrupt Status Enable Register	0x0 EF60 0F18	R/W	618
EMAC1_ISR	MMIO	EMAC 1 Interrupt Status Register	0x0 EF60 0F14	R/W	621
EMAC1_LSAH	MMIO	EMAC 1 Last Source Address Low	0x0 EF60 0F50	R	621
EMAC1_LSAL	MMIO	EMAC 1 Last Source Address High	0x0 EF60 0F54	R	621
EMAC1_MR0	MMIO	EMAC 1 Mode Register 0	0x0 EF60 0F00	R/W	611
EMAC1_MR1	MMIO	EMAC 1 Mode Register 1	0x0 EF60 0F04	R/W	612
EMAC1_OCRX	MMIO	EMAC 1 Number of Octets Received	0x0 EF60 0F6C	R	625
EMAC1_OCTX	MMIO	EMAC 1 Number of Octets Transmitted	0x0 EF60 0F68	R	625
EMAC1_PTR	MMIO	EMAC 1 Pause Timer Register	0x0 EF60 0F2C	R/W	621
EMAC1_RMR	MMIO	EMAC 1 Receive Mode Register	0x0 EF60 0F10	R/W	615
EMAC1_RWMR	MMIO	EMAC 1 Receive Low/High Water Mark Register	0x0 EF60 0F64	R/W	624
EMAC1_STACR	MMIO	EMAC 1 STA Control Register	0x0 EF60 0F5C	R/W	622
EMAC1_TMR0	MMIO	EMAC 1 Transmit Mode Register 0	0x0 EF60 0F08	R/W	613
EMAC1_TMR1	MMIO	EMAC 1 Transmit Mode Register 1	0x0 EF60 0F0C	R/W	614

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
EMAC1_TRTR	MMIO	EMAC 1 Transmit Request Threshold Register	0x0 EF60 0F60	R/W	623
EMAC1_VTCI	MMIO	EMAC 1 VLAN TCI Register	0x0 EF60 0F28	R/W	620
EMAC1_VTPID	MMIO	EMAC 1 VLAN TPID Register	0x0 EF60 0F24	R/W	620
FPR0	FPR	Floating Point Register	na	na	162
FPR1	FPR	Floating Point Register	na	na	162
FPR2	FPR	Floating Point Register	na	na	162
FPR3	FPR	Floating Point Register	na	na	162
FPR4	FPR	Floating Point Register	na	na	162
FPR5	FPR	Floating Point Register	na	na	162
FPR6	FPR	Floating Point Register	na	na	162
FPR7	FPR	Floating Point Register	na	na	162
FPR8	FPR	Floating Point Register	na	na	162
FPR9	FPR	Floating Point Register	na	na	162
FPR10	FPR	Floating Point Register	na	na	162
FPR11	FPR	Floating Point Register	na	na	162
FPR12	FPR	Floating Point Register	na	na	162
FPR13	FPR	Floating Point Register	na	na	162
FPR14	FPR	Floating Point Register	na	na	162
FPR15	FPR	Floating Point Register	na	na	162
FPR16	FPR	Floating Point Register	na	na	162
FPR17	FPR	Floating Point Register	na	na	162
FPR18	FPR	Floating Point Register	na	na	162
FPR19	FPR	Floating Point Register	na	na	162
FPR20	FPR	Floating Point Register	na	na	162
FPR21	FPR	Floating Point Register	na	na	162
FPR22	FPR	Floating Point Register	na	na	162
FPR23	FPR	Floating Point Register	na	na	162
FPR24	FPR	Floating Point Register	na	na	162
FPR25	FPR	Floating Point Register	na	na	162
FPR26	FPR	Floating Point Register	na	na	162
FPR27	FPR	Floating Point Register	na	na	162
FPR28	FPR	Floating Point Register	na	na	162
FPR29	FPR	Floating Point Register	na	na	162
FPR30	FPR	Floating Point Register	na	na	162
FPR31	FPR	Floating Point Register	na	na	162
FPSCR	FPR	Floating Point Status and Control Register	na	na	162
GPIO0_IR	MMIO	GPIO0 Input Register	0x0 EF60 0B1C	R	685
GPIO0_ISR1H	MMIO	GPIO0 Input Select Register 1 High	0x0 EF60 0B34	R/W	685

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
GPIO0_ISR1L	MMIO	GPIO0 Input Select Register 1 Low	0x0 EF60 0B30	R/W	685
GPIO0_ISR2H	MMIO	GPIO0 Input Select Register 2 High	0x0 EF60 0B3C	R/W	685
GPIO0_ISR2L	MMIO	GPIO0 Input Select Register 2 Low	0x0 EF60 0B38	R/W	685
GPIO0_ISR3H	MMIO	GPIO0 Input Select Register 3 High	0x0 EF60 0B44	R/W	685
GPIO0_ISR3L	MMIO	GPIO0 Input Select Register 3 Low	0x0 EF60 0B40	R/W	685
GPIO0_ODR	MMIO	GPIO0 Open Drain Register	0x0 EF60 0B18	R/W	684
GPIO0_OR	MMIO	GPIO0 Output	0x0 EF60 0B00	R/W	683
GPIO0_OSRH	MMIO	GPIO0 Output Select Register High	0x0 EF60 0B0C	R/W	683
GPIO0_OSRL	MMIO	GPIO0 Output Select Register Low	0x0 EF60 0B08	R/W	683
GPIO0_RR1	MMIO	GPIO0 Receive Register 1	0x0 EF60 0B20	R/W	686
GPIO0_RR2	MMIO	GPIO0 Receive Register 2	0x0 EF60 0B24	R/W	686
GPIO0_RR3	MMIO	GPIO0 Receive Register 3	0x0 EF60 0B28	R/W	686
GPIO0_TCR	MMIO	GPIO0 Three-State Control Register	0x0 EF60 0B04	R/W	683
GPIO0_TSRH	MMIO	GPIO0 Three-State Select Register High	0x0 EF60 0B14	R/W	684
GPIO0_TSRL	MMIO	GPIO0 Three-State Select Register Low	0x0 EF60 0B10	R/W	684
GPIO1_IR	MMIO	GPIO1 Input Register	0x0 EF60 0C1C	R	685
GPIO1_ISR1H	MMIO	GPIO1 Input Select Register 1 High	0x0 EF60 0C34	R/W	685
GPIO1_ISR1L	MMIO	GPIO1 Input Select Register 1 Low	0x0 EF60 0C30	R/W	685
GPIO1_ISR2H	MMIO	GPIO1 Input Select Register 2 High	0x0 EF60 0C3C	R/W	685
GPIO1_ISR2L	MMIO	GPIO1 Input Select Register 2 Low	0x0 EF60 0C38	R/W	685
GPIO1_ISR3H	MMIO	GPIO1 Input Select Register 3 High	0x0 EF60 0C44	R/W	685
GPIO1_ISR3L	MMIO	GPIO1 Input Select Register 3 Low	0x0 EF60 0C40	R/W	685
GPIO1_ODR	MMIO	GPIO1 Open Drain Register	0x0 EF60 0C18	R/W	684
GPIO1_OR	MMIO	GPIO1 Output Register	0x0 EF60 0C00	R/W	683
GPIO1_OSRH	MMIO	GPIO1 Output Select Register High	0x0 EF60 0C0C	R/W	683
GPIO1_OSRL	MMIO	GPIO1 Output Select Register Low	0x0 EF60 0C08	R/W	683
GPIO1_RR1	MMIO	GPIO1 Receive Register 1	0x0 EF60 0C20	R/W	686
GPIO1_RR2	MMIO	GPIO1 Receive Register 2	0x0 EF60 0C24	R/W	686
GPIO1_RR3	MMIO	GPIO1 Receive Register 3	0x0 EF60 0C28	R/W	686
GPIO1_TCR	MMIO	GPIO1 Three-State Control Register	0x0 EF60 0C04	R/W	683
GPIO1_TSRH	MMIO	GPIO1 Three-State Select Register High	0x0 EF60 0C14	R/W	684
GPIO1_TSRL	MMIO	GPIO1 Three-State Select Register Low	0x0 EF60 0C10	R/W	684
GPT0_COMP0	MMIO	GPT Compare Timer 0	0x0 EF60 0080	R/W	280
GPT0_COMP1	MMIO	GPT Compare Timer 1	0x0 EF60 0084	R/W	280
GPT0_COMP2	MMIO	GPT Compare Timer 2	0x0 EF60 0088	R/W	280
GPT0_COMP3	MMIO	GPT Compare Timer 3	0x0 EF60 008C	R/W	280
GPT0_COMP4	MMIO	GPT Compare Timer 4	0x0 EF60 0090	R/W	280
GPT0_COMP5	MMIO	GPT Compare Timer 5	0x0 EF60 0094	R/W	280

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
GPT0_COMP6	MMIO	GPT Compare Timer 6	0x0 EF60 0098	R/W	280
GPT0_DCIS	MMIO	Down Count Timer Interrupt Status	0x0 EF60 011C	R/W	281
GPT0_DCT0	MMIO	Down Count Timer	0x0 EF60 0110	R/W	280
GPT0_IE	MMIO	GPT Interrupt Enable	0x0 EF60 0024	R/W	279
GPT0_IM	MMIO	GPT Interrupt Mask	0x0 EF60 0018	R/W	278
GPT0_ISC	MMIO	GPT Interrupt Status (Clear bits if write 1)	0x0 EF60 0020	R/W	278
GPT0_ISS	MMIO	GPT Interrupt Status (Set bits if write 1)	0x0 EF60 001C	R/W	278
GPT0_MASK0	MMIO	GPT Compare Mask 0	0x0 EF60 00C0	R/W	280
GPT0_MASK1	MMIO	GPT Compare Mask 1	0x0 EF60 00C4	R/W	280
GPT0_MASK2	MMIO	GPT Compare Mask 2	0x0 EF60 00C8	R/W	280
GPT0_MASK3	MMIO	GPT Compare Mask 3	0x0 EF60 00CC	R/W	280
GPT0_MASK4	MMIO	GPT Compare Mask 4	0x0 EF60 00D0	R/W	280
GPT0_MASK5	MMIO	GPT Compare Mask 5	0x0 EF60 00D4	R/W	280
GPT0_MASK6	MMIO	GPT Compare Mask 6	0x0 EF60 00D8	R/W	280
GPT0_TBC	MMIO	GPT Time Base Counter	0x0 EF60 0000	R/W	277
IIC0_CLKDIV	MMIO	IIC 0 Clock Divide	0x0 EF60 070C	R/W	670
IIC0_CNTL	MMIO	IIC 0 Control	0x0 EF60 0706	R/W	664
IIC0_DIRECTCNTL	MMIO	IIC 0 Direct Control	0x0 EF60 0710	R/W	674
IIC0_EXTSTS	MMIO	IIC 0 Extended Status	0x0 EF60 0709	R/W	668
IIC0_HMADR	MMIO	IIC 0 High Master Address	0x0 EF60 0705	R/W	664
IIC0_HSADR	MMIO	IIC 0 High Slave Address	0x0 EF60 070B	R/W	670
IIC0_INTR	MMIO	IIC 0 Interrupt	0x0 EF60 0711	R	675
IIC0_INTRMSK	MMIO	IIC 0 Interrupt Mask	0x0 EF60 070D	R/W	671
IIC0_LMADR	MMIO	IIC 0 Low Master Address	0x0 EF60 0704	R/W	663
IIC0_LSADR	MMIO	IIC 0 Low Slave Address	0x0 EF60 070A	R/W	669
IIC0_MDBUF	MMIO	IIC 0 Master Data Buffer	0x0 EF60 0700	R/W	662
IIC0_MDCNTL	MMIO	IIC 0 Mode Control	0x0 EF60 0707	R/W	665
IIC0_SDBUF	MMIO	IIC 0 Slave Data Buffer	0x0 EF60 0702	R/W	662
IIC0_STS	MMIO	IIC 0 Status	0x0 EF60 0708	R/W	666
IIC0_XFRCNT	MMIO	IIC 0 Transfer Count	0x0 EF60 070E	R/W	672
IIC0_XTCNTLSS	MMIO	IIC 0 Extended Control and Slave Status	0x0 EF60 070F	R/W	673
IIC1_CLKDIV	MMIO	IIC 1 Clock Divide	0x0 EF60 080C	R/W	670
IIC1_CNTL	MMIO	IIC 1 Control	0x0 EF60 0806	R/W	664
IIC1_DIRECTCNTL	MMIO	IIC 1 Direct Control	0x0 EF60 0810	R/W	674
IIC1_EXTSTS	MMIO	IIC 1 Extended Status	0x0 EF60 0809	R/W	668
IIC1_HMADR	MMIO	IIC 1 High Master Address	0x0 EF60 0805	R/W	664
IIC1_HSADR	MMIO	IIC 1 High Slave Address	0x0 EF60 080B	R/W	670
IIC1_INTR	MMIO	IIC 1 Interrupt	0x0 EF60 0811	R	675

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
IIC1_INTRMSK	MMIO	IIC 1 Interrupt Mask	0x0 EF60 080D	R/W	671
IIC1_LMADR	MMIO	IIC 1 Low Master Address	0x0 EF60 0804	R/W	663
IIC1_LSADR	MMIO	IIC 1 Low Slave Address	0x0 EF60 080A	R/W	669
IIC1_MDBUF	MMIO	IIC 1 Master Data Buffer	0x0 EF60 0800	R/W	662
IIC1_MDCNTL	MMIO	IIC 1 Mode Control	0x0 EF60 0807	R/W	665
IIC1_SDBUF	MMIO	IIC 1 Slave Data Buffer	0x0 EF60 0802	R/W	662
IIC1_STS	MMIO	IIC 1 Status	0x0 EF60 0808	R/W	666
IIC1_XFRCNT	MMIO	IIC 1 Transfer Count	0x0 EF60 080E	R/W	672
IIC1_XTCNTLSS	MMIO	IIC 1 Extended Control and Slave Status	0x0 EF60 080F	R/W	673
MAL0_CFG	DCR	MAL Configuration Register	0x0180	R/W	567
MAL0_ESR	DCR	MAL Error Status Register	0x0181	R/Clear	570
MAL0_IER	DCR	MAL Interrupt Enable Register	0x0182	R/W	572
MAL0_RCBS0	DCR	MAL RX Channel 0 Buffer Size Register	0x01E0	R/W	574
MAL0_RCBS1	DCR	MAL RX Channel 1 Buffer Size Register	0x01E1	R/W	574
MAL0_RXCARR	DCR	MAL Rx Channel Active Register (Reset)	0x0191	R/W	569
MAL0_RXCASR	DCR	MAL Rx Channel Active Register (Set)	0x0190	R/W	569
MAL0_RXCTP0R	DCR	MAL RX Channel 0 Table Pointer Register	0x01C0	R/W	573
MAL0_RXCTP1R	DCR	MAL RX Channel 1 Table Pointer Register	0x01C1	R/W	573
MAL0_RXDEIR	DCR	MAL Rx Descriptor Error Interrupt Register	0x0193	R/Clear	572
MAL0_RXEOBISR	DCR	MAL Rx End of Buffer Interrupt Status Register	0x0192	R/Clear	570
MAL0_TXCARR	DCR	MAL Tx Channel Active Register (Reset)	0x0185	R/W	569
MAL0_TXCASR	DCR	MAL Tx Channel Active Register (Set)	0x0184	R/W	569
MAL0_TXCTP0R	DCR	MAL TX Channel 0 Table Pointer Register	0x01A0	R/W	573
MAL0_TXCTP1R	DCR	MAL TX Channel 1 Table Pointer Register	0x01A1	R/W	573
MAL0_TXCTP2R	DCR	MAL TX Channel 2 Table Pointer Register	0x01A2	R/W	573
MAL0_TXCTP3R	DCR	MAL TX Channel 3 Table Pointer Register	0x01A3	R/W	573
MAL0_TXDEIR	DCR	MAL Tx Descriptor Error Interrupt Register	0x0187	R/Clear	572
MAL0_TXEOBISR	DCR	MAL Tx End of Buffer Interrupt Status Register	0x0186	R/Clear	570
NDFC0_ADDR	MMIO	NDFC Address Register	0x0 xxxx 0004	R/W	499
NDFC0_B0CR	MMIO	NDFC Bank Configuration Register 0	0x0 xxxx 0030	R/W	502
NDFC0_B1CR	MMIO	NDFC Bank Configuration Register 1	0x0 xxxx 0034	R/W	502
NDFC0_B2CR	MMIO	NDFC Bank Configuration Register 2	0x0 xxxx 0038	R/W	502
NDFC0_B3CR	MMIO	NDFC Bank Configuration Register 3	0x0 xxxx 003C	R/W	502
NDFC0_CMD	MMIO	NDFC Command Register	0x0 xxxx 0000	R/W	499
NDFC0_CR	MMIO	NDFC Configuration Register	0x0 xxxx 0040	R/W	503
NDFC0_DATA	MMIO	NDFC Data Register	0x0 xxxx 0008	R/W	500
NDFC0_ECC0	MMIO	NDFC ECC Register 0	0x0 xxxx 0010	R	501
NDFC0_ECC1	MMIO	NDFC ECC Register 1	0x0 xxxx 0014	R	501

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
NDFC0_ECC2	MMIO	NDFC ECC Register 2	0x0 xxxx 0018	R	501
NDFC0_ECC3	MMIO	NDFC ECC Register 3	0x0 xxxx 001C	R	501
NDFC0_ECC4	MMIO	NDFC ECC Register 4	0x0 xxxx 0020	R	501
NDFC0_ECC5	MMIO	NDFC ECC Register 5	0x0 xxxx 0024	R	501
NDFC0_ECC6	MMIO	NDFC ECC Register 6	0x0 xxxx 0028	R	501
NDFC0_ECC7	MMIO	NDFC ECC Register 7	0x0 xxxx 002C	R	501
NDFC0_HWCTL	MMIO	NDFC Direct Hardware Control Register	0x0 xxxx 0048	R/W	505
NDFC0_REVID	MMIO	NDFC Revision ID Register	0x0 xxxx 0050	R	506
NDFC0_SR	MMIO	NDFC Status Register	0x0 xxxx 0044	R	505
OPBA0_CR	MMIO	OPB Arbiter Control Register	0x0 EF60 0A01	R/W	107
OPBA0_PR	MMIO	OPB Arbiter Priority Register	0x0 EF60 0A00	R/W	106
OPB2PLB30_BCTRL	DCR	OPB to PLB 3 Bridge Control Register	0x00A8	R/W	108
OPB2PLB30_BSTAT	DCR	OPB to PLB 3 Bridge Status Register	0x00A9	R	108
OPB2PLB30_REVID	DCR	OPB to PLB 3 Bridge Revision ID Register	0x00AA	R	108
P3P4BI0_BEARH	DCR	PLB3 to PLB4 Bridge Error Address High	0x0033	R/Clear	87
P3P4BI0_BEARL	DCR	PLB3 to PLB4 Bridge Error Address Low	0x0032	R/Clear	87
P3P4BI0_BESR0	DCR	PLB3 to PLB4 Bridge Error Status 0	0x0030	R/Clear	87
P3P4BI0_BESR1	DCR	PLB3 to PLB4 Bridge Error Status 1	0x0034	R/Clear	90
P3P4BI0_CFG	DCR	PLB3 to PLB4 Bridge Configuration	0x0036	R/Clear/Set	91
P3P4BI0_PEIR	DCR	PLB3 to PLB4 Bridge Parity Error Interrupt	0x0038	R/Clear/Set	93
P3P4BI0_PICR	DCR	PLB3 to PLB4 Bridge Priority	0x0037	R/Clear/Set	92
P3P4BI0_REVID	DCR	PLB3 to PLB4 Bridge Revision ID	0x003A	R/Clear	93
P4P3BO0_BEARH	DCR	PLB4 to PLB3 Bridge Error Address High	0x0023	R/Clear	80
P4P3BO0_BEARL	DCR	PLB4 to PLB3 Bridge Error Address Low	0x0022	R/Clear	80
P4P3BO0_BESR0	DCR	PLB4 to PLB3 Bridge Error Status 0	0x0020	R/Clear	81
P4P3BO0_BESR1	DCR	PLB4 to PLB3 Bridge Error Status 1	0x0024	R/Clear	83
P4P3BO0_CFG	DCR	PLB4 to PLB3 Bridge Configuration	0x0026	R/Clear/Set	84
P4P3BO0_PEIR	DCR	PLB4 to PLB3 Bridge Parity Error Interrupt	0x0028	R/Clear/Set	86
P4P3BO0_PICR	DCR	PLB4 to PLB3 Bridge Priority	0x0027	R/Clear/Set	85
P4P3BO0_REVID	DCR	PLB4 to PLB3 Bridge Revision ID	0x002A	R/Clear	86
PCIC0_BIST	MMIO	PCI Built In Self Test Control	offset 0x8000 000F	R	391
PCIC0_BRDGOPT1	MMIO	PCI Bridge Options 1	offset 0x8000 004A	R/W	396
PCIC0_BRDGOPT2	MMIO	PCI Bridge Options 2	offset 0x8000 0060	R/W	401
PCIC0_CACHELS	MMIO	PCI Cache Line Size	offset 0x8000 000C	R	390
PCIC0_CAP	MMIO	PCI Capabilities Pointer	offset 0x8000 0034	R	393
PCIC0_CAPID	MMIO	Capability Identifier	offset 0x8000 0058	R	399
PCIC0_CFGADDR	MMIO	PCI Space Configuration Address	0x0 EEC0 0000	R/W	386
PCIC0_CFGDATA	MMIO	PCI Space Configuration Data	0x0 EEC0 0004	R/W	387

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PCIC0_CLS	MMIO	PCI Class	offset 0x8000 0009	R/W	390
PCIC0_CMD	MMIO	PCI Command	offset 0x8000 0004	R/W	388
PCIC0_DATA	MMIO	Data	offset 0x8000 005F	R	401
PCIC0_DEVID	MMIO	PCI Device ID	offset 0x8000 0002	R/W	387
PCIC0_ERREN	MMIO	Error Enable	offset 0x8000 0048	R/W	394
PCIC0_ERRSTS	MMIO	Error Status	offset 0x8000 0049	R/W	395
PCIC0_HDTYPE	MMIO	PCI Header Type	offset 0x8000 000E	R	391
PCIC0_ICS	MMIO	PCI Interrupt Control/Status	offset 0x8000 0044	R/W	394
PCIC0_INTLN	MMIO	PCI Interrupt Line	offset 0x8000 003C	R/W	393
PCIC0_INTPN	MMIO	PCI Interrupt Pin	offset 0x8000 003D	R	393
PCIC0_LATTIM	MMIO	PCI Latency Timer	offset 0x8000 000D	R/W	390
PCIC0_MAXLTNCY	MMIO	PCI Maximum Latency	offset 0x8000 003F	R	394
PCIC0_MINGNT	MMIO	PCI Minimum Grant	offset 0x8000 003E	R	394
PCIC0_NEXTIPTR	MMIO	Next Item Pointer	offset 0x8000 0059	R	400
PCIC0_PLBBEAR	MMIO	PLB Slave Error Address Register	offset 0x8000 0054	R/W	399
PCIC0_PLBBESR0	MMIO	PLB Slave Error Syndrome 0	offset 0x8000 004C	R/W	396
PCIC0_PLBBESR1	MMIO	PLB Slave Error Syndrome 1	offset 0x8000 0050	R/W	398
PCIC0_PMC	MMIO	Power Management Capabilities	offset 0x8000 005A	R	400
PCIC0_PMCSR	MMIO	Power Management Control Status	offset 0x8000 005C	R/W	400
PCIC0_PMCSRBSE	MMIO	PMCSR PCI-to-PCI Bridge Support Extensions	offset 0x8000 005E	R	401
PCIC0_PMSCRR	MMIO	Power Management State Change Request Register	offset 0x8000 0064	R/W	402
PCIC0_PTM1BAR	MMIO	PCI PTM 1 BAR	offset 0x8000 0014	R/W	391
PCIC0_PTM2BAR	MMIO	PCI PTM 2 BAR	offset 0x8000 0018	R/W	392
PCIC0_REVID	MMIO	PCI Revision ID	offset 0x8000 0008	R/W	390
PCIC0_SBSYSID	MMIO	PCI Subsystem ID	offset 0x8000 002E	R/W	393
PCIC0_SBSYSVID	MMIO	PCI Subsystem Vendor ID	offset 0x8000 002C	R/W	393
PCIC0_STATUS	MMIO	PCI Status	offset 0x8000 0006	R/W	388
PCIC0_VENDID	MMIO	PCI Vendor ID	offset 0x8000 0000	R/W	387
PCIL0_PMM0LA	MMIO	PMM 0 Local Address	0x0 EF40 0000	R/W	381
PCIL0_PMM0MA	MMIO	PMM 0 Mask/Attribute	0x0 EF40 0004	R/W	381
PCIL0_PMM0PCIHA	MMIO	PMM 0 PCI High Address	0x0 EF40 000C	R/W	382
PCIL0_PMM0PCILA	MMIO	PMM 0 PCI Low Address	0x0 EF40 0008	R/W	382
PCIL0_PMM1LA	MMIO	PMM 1 Local Address	0x0 EF40 0010	R/W	382
PCIL0_PMM1MA	MMIO	PMM 1 Mask/Attribute	0x0 EF40 0014	R/W	382
PCIL0_PMM1PCIHA	MMIO	PMM 1 PCI High Address	0x0 EF40 001C	R/W	383
PCIL0_PMM1PCILA	MMIO	PMM 1 PCI Low Address	0x0 EF40 0018	R/W	383
PCIL0_PMM2LA	MMIO	PMM 2 Local Address	0x0 EF40 0020	R/W	383
PCIL0_PMM2MA	MMIO	PMM 2 Mask/Attribute	0x0 EF40 0024	R/W	384

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PCIL0_PMM2PCIHA	MMIO	PMM 2 PCI High Address	0x0 EF40 002C	R/W	384
PCIL0_PMM2PCILA	MMIO	PMM 2 PCI Low Address	0x0 EF40 0028	R/W	384
PCIL0_PTM1LA	MMIO	PTM 1 Local Address	0x0 EF40 0034	R/W	385
PCIL0_PTM1MS	MMIO	PTM 1 Memory Size/Attribute	0x0 EF40 0030	R/W	385
PCIL0_PTM2LA	MMIO	PTM 2 Local Address	0x0 EF40 003C	R/W	386
PCIL0_PTM2MS	MMIO	PTM 2 Memory Size/Attribute	0x0 EF40 0038	R/W	385
PLB32OPB0_BEAR	DCR	PLB 3 to OPB 0 Bridge Error Address Register	0x0092	R	104
PLB32OPB0_BESR0	DCR	PLB 3 to OPB 0 Bridge Error Status Register 0	0x0090	R/Clear	102
PLB32OPB0_BESR1	DCR	PLB 3 to OPB 0 Bridge Error Status Register 1	0x0094	R/Clear	104
PLB32OPB0_REVID	DCR	PLB 3 to OPB 0 Bridge Revision ID Register	0x0093	R	104
PLB3A0_ACR	DCR	PLB3 Arbiter Control Register	0x0077	R/W	101
PLB3A0_BEAR	DCR	PLB3 Arbiter Bus Error Address Register	0x0076	R/W	100
PLB3A0_BESR	DCR	PLB3 Arbiter Bus Error Status Register	0x0074	R/Clear	99
PLB3A0_REVID	DCR	PLB3 Arbiter Revision ID Register	0x0072	R	98
PLB42OPB1_BEARH	DCR	PLB 4 to OPB 1 Bridge Error Address Register High	0x0203	R	96
PLB42OPB1_BEARL	DCR	PLB 4 to OPB 1 Bridge Error Address Register Low	0x0202	R	96
PLB42OPB1_BESR0	DCR	PLB 4 to OPB 1 Bridge Error Status Register 0	0x0200	R/Clear	94
PLB42OPB1_BESR1	DCR	PLB 4 to OPB 1 Bridge Error Status Register 1	0x0204	R/Clear	96
PLB42OPB1_CFG	DCR	PLB 4 to OPB 1 Bridge Configuration Register	0x0206	R/Clear	97
PLB42OPB1_LATENCY	DCR	PLB 4 to OPB 1 Bridge Burst Latency Register	0x0208	R/Clear	98
PLB42OPB1_REVID	DCR	PLB 4 to OPB 1 Bridge Revision ID Register	0x020A	R	98
PLB4A0_ACR	DCR	PLB4A0 Arbiter Control Register	0x0081	R/W	73
PLB4A0_CCR	DCR	PLB4 Crossbar Control Register	0x0088	R/W	79
PLB4A0_EARH	DCR	PLB4A0 Error Address Register High	0x0085	R	78
PLB4A0_EARL	DCR	PLB4A0 Error Address Register Low	0x0084	R	78
PLB4A0_ESRH*	DCR	PLB4A0 Error Status Register High (*diagnostic use only)	0x0087	Set(W)	na
PLB4A0_ESRL	DCR	PLB4A0 Error Status Register Low	0x0082	R/Clear	77
PLB4A0_ESRL*	DCR	PLB4A0 Error Status Register Low (*diagnostic use only)	0x0086	Set(W)	77
PLB4A0_REVID	DCR	PLB4 Crossbar ID/Revision Register	0x0080	R	73
PLB4A1_ACR	DCR	PLB4A1 Arbiter Control Register	0x0089	R/W	73
PLB4A1_EARH	DCR	PLB4A1 Error Address Register High	0x008D	R	78
PLB4A1_EARL	DCR	PLB4A1 Error Address Register Low	0x008C	R	78
PLB4A1_ESRH*	DCR	PLB4A1 Error Status Register High (*diagnostic use only)	0x008F	Set(W)	na
PLB4A1_ESRL	DCR	PLB4A1 Error Status Register Low	0x008A	R/Clear	77
PLB4A1_ESRL*	DCR	PLB4A1 Error Status Register Low (*diagnostic use only)	0x008E	Set(W)	77
PPM0_CCR	DCR	Cycle Control Register	offset 0x0003	R/W	124
PPM0_CFGADDR	DCR	Performance Monitor Configuration Address Register	0x0016	R/W	121
PPM0_CFGDATA	DCR	Performance Monitor Configuration Data Register	0x0017	R/W	121

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PPM0_CR	DCR	Control Register	offset 0x0002	R/W	123
PPM0_DCMNR0	DCR	Duration Counter Minimum Register 0	offset 0x0035	R/W	132
PPM0_DCMNR1	DCR	Duration Counter Minimum Register 1	offset 0x0036	R/W	132
PPM0_DCMXR0	DCR	Duration Counter Maximum Register 0	offset 0x0033	R/W	132
PPM0_DCMXR1	DCR	Duration Counter Maximum Register 1	offset 0x0034	R/W	132
PPM0_DCOTR0	DCR	Duration Counter Occurrence Total Register 0	offset 0x0039	R/W	133
PPM0_DCOTR1	DCR	Duration Counter Occurrence Total Register 1	offset 0x003A	R/W	133
PPM0_DCSR0	DCR	Duration Counter Selection Register 0	offset 0x0031	R/W	131
PPM0_DCSR1	DCR	Duration Counter Selection Register 1	offset 0x0032	R/W	131
PPM0_DCTVR0	DCR	Duration Counter Total Value Register 0	offset 0x0037	R/W	133
PPM0_DCTVR1	DCR	Duration Counter Total Value Register 1	offset 0x0038	R/W	133
PPM0_GCR0	DCR	Generic Pipeline Event Counter Register 0	offset 0x002D	R/W	131
PPM0_GCR1	DCR	Generic Pipeline Event Counter Register 1	offset 0x002E	R/W	131
PPM0_GCR2	DCR	Generic Pipeline Event Counter Register 2	offset 0x002F	R/W	131
PPM0_GCR3	DCR	Generic Pipeline Event Counter Register 3	offset 0x0030	R/W	131
PPM0_GCSR0	DCR	Generic Event Counter Selection Register 0	offset 0x0019	R/W	130
PPM0_GCSR1	DCR	Generic Event Counter Selection Register 1	offset 0x001A	R/W	130
PPM0_GCSR2	DCR	Generic Event Counter Selection Register 2	offset 0x001B	R/W	130
PPM0_GCSR3	DCR	Generic Event Counter Selection Register 3	offset 0x001C	R/W	130
PPM0_ISR	DCR	Interrupt Status Register	offset 0x0000	R	121
PPM0_LAMR	DCR	Lower Address Mask Register	offset 0x0007	R/W	126
PPM0_LAR	DCR	Lower Address Register	offset 0x0005	R/W	125
PPM0_MCR0	DCR	Master Event Counter Register 0	offset 0x001D	R/W	130
PPM0_MCR1	DCR	Master Event Counter Register 1	offset 0x001E	R/W	130
PPM0_MCR2	DCR	Master Event Counter Register 2	offset 0x001F	R/W	130
PPM0_MCR3	DCR	Master Event Counter Register 3	offset 0x0020	R/W	130
PPM0_MCSR0	DCR	Master Event Counter Selection Register 0	offset 0x0009	R/W	126
PPM0_MCSR1	DCR	Master Event Counter Selection Register 1	offset 0x000A	R/W	126
PPM0_MCSR2	DCR	Master Event Counter Selection Register 2	offset 0x000B	R/W	126
PPM0_MCSR3	DCR	Master Event Counter Selection Register 3	offset 0x000C	R/W	126
PPM0_RIDR	DCR	Revision ID Register	offset 0x0008	R	126
PPM0_SCR0	DCR	Slave Event Counter Register 0	offset 0x0025	R/W	130
PPM0_SCR1	DCR	Slave Event Counter Register 1	offset 0x0026	R/W	130
PPM0_SCR2	DCR	Slave Event Counter Register 2	offset 0x0027	R/W	130
PPM0_SCR3	DCR	Slave Event Counter Register 3	offset 0x0028	R/W	130
PPM0_SCSR0	DCR	Slave Event Counter Selection Register 0	offset 0x0011	R/W	128
PPM0_SCSR1	DCR	Slave Event Counter Selection Register 1	offset 0x0012	R/W	128
PPM0_SCSR2	DCR	Slave Event Counter Selection Register 2	offset 0x0013	R/W	128

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PPM0_SCSR3	DCR	Slave Event Counter Selection Register 3	offset 0x0014	R/W	128
PPM0_UAMR	DCR	Upper Address Mask Register	offset 0x0006	R/W	125
PPM0_UAR	DCR	Upper Address Register	offset 0x0004	R/W	125
SDR0_AMP0	DCR	Alternate PLB4 Master Priority Register	offset 0x0240	R/W	66
SDR0_AMP1	DCR	Alternate PLB3 Master Priority Register	offset 0x0241	R/W	67
SDR0_CFGADDR	DCR	System DCR Configuration Address Register	0x000E	R/W	196
SDR0_CFGDATA	DCR	System DCR Configuration Data Register	0x000F	R/W	196
SDR0_CP440	DCR	440CPU Control Register	offset 0x0180	R/W	68
SDR0_CUST0	DCR	Register0 Reserved for Customer Use	offset 0x4000	R/W	221
SDR0_CUST1	DCR	Register1 Reserved for Customer Use	offset 0x4002	R/W	222
SDR0_DDRDL0	DCR	DDR Delay Line Register	offset 0x00E0	R/W	336
SDR0_EBC0	DCR	EBC Configuration Register	offset 0x0100	R/W	222
SDR0_ECID0	DCR	Electronic Chip ID Register 0	offset 0x0080	R/W	197
SDR0_ECID1	DCR	Electronic Chip ID Register 1	offset 0x0081	R/W	197
SDR0_ECID2	DCR	Electronic Chip ID Register 2	offset 0x0082	R/W	197
SDR0_ECID3	DCR	Electronic Chip ID Register 3	offset 0x0083	R/W	197
SDR0_EMAC0REJCNT	DCR	EMAC0 RX Packet Reject Counter	offset 0x4303	R	592
SDR0_EMAC0RXST	DCR	EMAC0 RX Status Register	offset 0x4301	R/W	589
SDR0_EMAC0TXST	DCR	EMAC0 TX Status Register	offset 0x4302	R/W	591
SDR0_EMAC1REJCNT	DCR	EMAC1 RX Packet Reject Counter	offset 0x4306	R	592
SDR0_EMAC1RXST	DCR	EMAC1 RX Status Register	offset 0x4304	R/W	589
SDR0_EMAC1TXST	DCR	EMAC1 TX Status Register	offset 0x4305	R/W	591
SDR0_HSF	DCR	DDR Hardware Self Refresh Register	offset 0x4400	R/W	201
SDR0_JTAGID	DCR	JTAG ID Register	offset 0x00C0	R/W	na[1]
SDR0_MALRBL	DCR	MAL Receive Burst Length Register	offset 0x02A0	R/W	566
SDR0_MALRBS	DCR	Reserved	offset 0x02E0	R/W	566
SDR0_MALTBL	DCR	MAL Transmit Burst Length Register	offset 0x0280	R/W	566
SDR0_MALTBS	DCR	Reserved	offset 0x02C0	R/W	567
SDR0_MFR	DCR	Miscellaneous Function Register	offset 0x4300	R/W	200
SDR0_MIRQ0	DCR	Master Interrupt Request Register 0 (PLB3)	offset 0x0260	R/W	69
SDR0_PCI0	DCR	PCI Control Register	offset 0x0300	R/W	378
SDR0_PFC0	DCR	Pin Function Control Register 0	offset 0x4100	R/W	197
SDR0_PFC1	DCR	Pin Function Control Register 1	offset 0x4101	R/W	198
SDR0_PINSTP	DCR	Pin Strapping Register	offset 0x0040	R	217
SDR0_SDCS0	DCR	Serial Device Controller Settings Register	offset 0x0060	R/W	217
SDR0_SDSTP0	DCR	Serial Device Strap Register 0	offset 0x0020	R	218
SDR0_SDSTP1	DCR	Serial Device Strap Register 1	offset 0x0021	R	219
SDR0_SDSTP2	DCR	Read Only Version of SDR0_CUST0	offset 0x4001	R	220

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
SDR0_SDSTP3	DCR	Read Only Version of SDR0_CUST1	offset 0x4003	R	221
SDR0_SLPIPE0	DCR	PLB Slave Address Pipeline Disabling Register	offset 0x0220	R/W	71
SDR0_SRST0	DCR	Individual Core Reset Control Register 0	offset 0x0200	R/W	199
SDR0_SRST1	DCR	Individual Core Reset Control Register 1	offset 0x0201	R/W	199
SDR0_UART0	DCR	UART Configuration Register 0	offset 0x0120	R/W	640
SDR0_UART1	DCR	UART Configuration Register 1	offset 0x0121	R/W	641
SDR0_UART2	DCR	UART Configuration Register 2	offset 0x0122	R/W	642
SDR0_UART3	DCR	UART Configuration Register 3	offset 0x0123	R/W	642
SDR0_USB0	DCR	USB Configuration Register	offset 0x0320	R/W	695
SDRAM0_B0CR	DCR	DDR SDRAM Bank 0 Configuration	offset 0x0040	R/W	325
SDRAM0_B1CR	DCR	DDR SDRAM Bank 1 Configuration	offset 0x0044	R/W	325
SDRAM0_B2CR	DCR	DDR SDRAM Bank 2 Configuration	offset 0x0048	R/W	325
SDRAM0_B3CR	DCR	DDR SDRAM Bank 3 Configuration	offset 0x004C	R/W	325
SDRAM0_BEAR	DCR	Bus Error Address Register	offset 0x0010	R	319
SDRAM0_BESR0	DCR	Bus Error Status Register 0	offset 0x0000	R/W	317
SDRAM0_BESR1	DCR	Bus Error Status Register 1	offset 0x0008	R/W	318
SDRAM0_CFG0	DCR	DDR SDRAM Controller Options 0	offset 0x0020	R/W	321
SDRAM0_CFG1	DCR	DDR SDRAM Controller Options 1	offset 0x0021	R/W	322
SDRAM0_CFGADDR	DCR	DDR-SDRAM Address Register	0x0010	R/W	313
SDRAM0_CFGDATA	DCR	DDR-SDRAM Data Register	0x0011	R/W	313
SDRAM0_CID	DCR	Controller ID Register	offset 0x00A4	R/W	341
SDRAM0_CLKTR	DCR	DDR SDRAM Clock Timing Register	offset 0x0082	R/W	334
SDRAM0_DEVOPT	DCR	DDR SDRAM Device Options	offset 0x0022	R/W	323
SDRAM0_DLYCAL	DCR	Delay Line Calibration Register	offset 0x0084	R/W	339
SDRAM0_ECCESR	DCR	ECC Error Status	offset 0x0098	R/W	340
SDRAM0_MCSTS	DCR	DDR SDRAM Memory Controller Status	offset 0x0024	R	323
SDRAM0_MIRQ	DCR	Master Write Interrupt	offset 0x0011	R/W	319
SDRAM0_PMIT	DCR	Power Management Idle Timer	offset 0x0034	R/W	324
SDRAM0_RID	DCR	Revision ID Register	offset 0x00A8	R	341
SDRAM0_RTR	DCR	Refresh Timer Register	offset 0x0030	R/W	324
SDRAM0_SLIO	DCR	PLB Slave Interface Options	offset 0x0018	R/W	320
SDRAM0_TR0	DCR	DDR SDRAM Timing Register 0	offset 0x0080	R/W	326
SDRAM0_TR1	DCR	DDR SDRAM Timing Register 1	offset 0x0081	R/W	328
SDRAM0_UABBA	DCR	PLB UA Bus Base Address	offset 0x0038	R/W	324
SDRAM0_WDDCTR	DCR	Write Data, DQS, DM Clock Timing Register	offset 0x0083	R/W	337
SPI0_CDM	MMIO	SPI Clock Divisor Modulus Register	0x0 EF60 0906	R/W	657
SPI0_CR	MMIO	SPI Control Register	0x0 EF60 0903	R/W	656
SPI0_MODE	MMIO	SPI Mode Register	0x0 EF60 0900	R/W	658

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
SPI0_RxD	MMIO	SPI Receive Data Register	0x0 EF60 0901	R	656
SPI0_SR	MMIO	SPI Status Register	0x0 EF60 0904	R	657
SPI0_TxD	MMIO	SPI Transmit Data Register	0x0 EF60 0902	R/W	656
UART0_DLL	MMIO	UART 0 Baud-rate Divisor Latch LSB	0x0 EF60 0300	R/W	639
UART0_DLM	MMIO	UART 0 Baud-rate Divisor Latch MSB	0x0 EF60 0301	R/W	639
UART0_FCR	MMIO	UART 0 FIFO Control Register	0x0 EF60 0302	W	635
UART0_IER	MMIO	UART 0 Interrupt Enable Register	0x0 EF60 0301	R/W	633
UART0_IIR	MMIO	UART 0 Interrupt Identification Register	0x0 EF60 0302	R	634
UART0_LCR	MMIO	UART 0 Line Control Register	0x0 EF60 0303	R/W	635
UART0_LSR	MMIO	UART 0 Line Status Register	0x0 EF60 0305	R/W	637
UART0_MCR	MMIO	UART 0 Modem Control Register	0x0 EF60 0304	R/W	636
UART0_MSR	MMIO	UART 0 Modem Status Register	0x0 EF60 0306	R/W	638
UART0_RBR	MMIO	UART 0 Receiver Buffer Register	0x0 EF60 0300	R	633
UART0_SCR	MMIO	UART 0 Scratch Register	0x0 EF60 0307	R/W	639
UART0_THR	MMIO	UART 0 Transmitter Holding Register	0x0 EF60 0300	W	633
UART1_DLL	MMIO	UART 1 Baud-rate Divisor Latch LSB	0x0 EF60 0400	R/W	639
UART1_DLM	MMIO	UART 1 Baud-rate Divisor Latch MSB	0x0 EF60 0401	R/W	639
UART1_FCR	MMIO	UART 1 FIFO Control Register	0x0 EF60 0402	W	635
UART1_IER	MMIO	UART 1 Interrupt Enable Register	0x0 EF60 0401	R/W	633
UART1_IIR	MMIO	UART 1 Interrupt Identification Register	0x0 EF60 0402	R	634
UART1_LCR	MMIO	UART 1 Line Control Register	0x0 EF60 0403	R/W	635
UART1_LSR	MMIO	UART 1 Line Status Register	0x0 EF60 0405	R/W	637
UART1_MCR	MMIO	UART 1 Modem Control Register	0x0 EF60 0404	R/W	636
UART1_MSR	MMIO	UART 1 Modem Status Register	0x0 EF60 0406	R/W	638
UART1_RBR	MMIO	UART 1 Receiver Buffer Register	0x0 EF60 0400	R	633
UART1_SCR	MMIO	UART 1 Scratch Register	0x0 EF60 0407	R/W	639
UART1_THR	MMIO	UART 1 Transmitter Holding Register	0x0 EF60 0400	W	633
UART2_DLL	MMIO	UART 2 Baud-rate Divisor Latch LSB	0x0 EF60 0500	R/W	639
UART2_DLM	MMIO	UART 2 Baud-rate Divisor Latch MSB	0x0 EF60 0501	R/W	639
UART2_FCR	MMIO	UART 2 FIFO Control Register	0x0 EF60 0502	W	635
UART2_IER	MMIO	UART 2 Interrupt Enable Register	0x0 EF60 0501	R/W	633
UART2_IIR	MMIO	UART 2 Interrupt Identification Register	0x0 EF60 0502	R	634
UART2_LCR	MMIO	UART 2 Line Control Register	0x0 EF60 0503	R/W	635
UART2_LSR	MMIO	UART 2 Line Status Register	0x0 EF60 0505	R/W	637
UART2_MCR	MMIO	UART 2 Modem Control Register	0x0 EF60 0504	R/W	636
UART2_MSR	MMIO	UART 2 Modem Status Register	0x0 EF60 0506	R/W	638
UART2_RBR	MMIO	UART 2 Receiver Buffer Register	0x0 EF60 0500	R	633
UART2_SCR	MMIO	UART 2 Scratch Register	0x0 EF60 0507	R/W	639

Table 33-1. Alphabetical Listing of Chip Control and Peripheral Core Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
UART2_THR	MMIO	UART 2 Transmitter Holding Register	0x0 EF60 0500	W	633
UART3_DLL	MMIO	UART 3 Baud-rate Divisor Latch LSB	0x0 EF60 0600	R/W	639
UART3_DLM	MMIO	UART 3 Baud-rate Divisor Latch MSB	0x0 EF60 0601	R/W	639
UART3_FCR	MMIO	UART 3 FIFO Control Register	0x0 EF60 0602	W	635
UART3_IER	MMIO	UART 3 Interrupt Enable Register	0x0 EF60 0601	R/W	633
UART3_IIR	MMIO	UART 3 Interrupt Identification Register	0x0 EF60 0602	R	634
UART3_LCR	MMIO	UART 3 Line Control Register	0x0 EF60 0603	R/W	635
UART3_LSR	MMIO	UART 3 Line Status Register	0x0 EF60 0605	R/W	637
UART3_MCR	MMIO	UART 3 Modem Control Register	0x0 EF60 0604	R/W	636
UART3_MSR	MMIO	UART 3 Modem Status Register	0x0 EF60 0606	R/W	638
UART3_RBR	MMIO	UART 3 Receiver Buffer Register	0x0 EF60 0600	R	633
UART3_SCR	MMIO	UART 3 Scratch Register	0x0 EF60 0607	R/W	639
UART3_THR	MMIO	UART 3 Transmitter Holding Register	0x0 EF60 0600	W	633
UIC0_CR	DCR	UIC 0 Critical Register	0x00C3	R/W	235
UIC0_ER	DCR	UIC 0 Enable Register	0x00C2	R/W	230
UIC0_MSR	DCR	UIC 0 Masked Status Register	0x00C6	R	248
UIC0_PR	DCR	UIC 0 Polarity Register	0x00C4	R/W	239
UIC0_SR	DCR	UIC 0 Status Register	0x00C0	R/Clear	226
UIC0_SRS	DCR	UIC 0 Status Register Set (reserved for debug only)	0x00C1	W/Set	na
UIC0_TR	DCR	UIC 0 Triggering Register	0x00C5	R/W	244
UIC0_VCR	DCR	UIC 0 Vector Configuration Register	0x00C8	W	253
UIC0_VR	DCR	UIC 0 Vector Register	0x00C7	R	254
UIC1_CR	DCR	UIC 1 Critical Register	0x00D3	R/W	237
UIC1_ER	DCR	UIC 1 Enable Register	0x00D2	R/W	233
UIC1_MSR	DCR	UIC 1 Masked Status Register	0x00D6	R	250
UIC1_PR	DCR	UIC 1 Polarity Register	0x00D4	R/W	241
UIC1_SR	DCR	UIC 1 Status Register	0x00D0	R/Clear	228
UIC1_SRS	DCR	UIC 1 Status Register Set (reserved for debug only)	0x00D1	W/Set	na
UIC1_TR	DCR	UIC 1 Triggering Register	0x00D5	R/W	246
UIC1_VCR	DCR	UIC 1 Vector Configuration Register	0x00D8	W	253
UIC1_VR	DCR	UIC 1 Vector Register	0x00D7	R	255
ZMII0_FER	MMIO	ZMII Function Enable Register	0x0 EF60 0D00	R/W	579
ZMII0_SMIISR	MMIO	ZMII SMII Status Register	0x0 EF60 0D08	R/W	581
ZMII0_SSR	MMIO	ZMII Speed Select Register	0x0 EF60 0D04	R/W	580

34. Signal Summary

This chapter provides information about the PPC440EP I/O signals.

34.1 Signals Listed Alphabetically

The *Signals Listed Alphabetically* table in the *PPC440EP Embedded Processor Data Sheet* provides a complete list of all the PPC440EP I/O signals in alphabetical order.

34.2 Signals by Interface Category

The *Signal Functional Description* table in the *PPC440EP Embedded Processor Data Sheet* provides a complete list of all the PPC440EP I/O signals grouped together by interface category. The functional characteristics of the signal are described.

Appendix A. Floating Point Instruction Summary

This appendix contains PPC440 FPU instructions summarized alphabetically and by opcode.

Instruction Set – Alphabetical lists all PPC440 FPU mnemonics, including extended mnemonics, alphabetically. A short functional description is included for each mnemonic.

Instructions Sorted by Opcode, lists all PPC440 FPU instructions, sorted by primary and secondary opcodes. Extended mnemonics are not included in the opcode list.

Instruction Formats, illustrates the PPC440 FPU instruction forms (allowed arrangements of fields within instructions).

A.1 Instruction Set – Alphabetical

Table A-1 summarizes the PPC440 FPU instruction set. The instructions are described in detail in *Floating Point Instruction Set* on page 755 which is also alphabetized under the root form. It also describes the instruction operands and notation.

Table A-1. PPC440 FPU Instruction Syntax Summary

Mnemonic	Operands	Function	Other Registers Changed	Page
fabs	FRT, FRB	Place (FRB), with bit 0 set to zero, into FRT.		759
fadd	FRT, FRA, FRB	Add (FRA) to (FRB). Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	760
fadds	FRT, FRA, FRB	Add (FRA) to (FRB). Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	761
fcmpo	BF, FRA, FRB	(FRA) is compared to (FRB). Place result in CR[BF] and FPSCR[FPCC].	CR[BF FPSCR[FPCC, FX, VXSNAN, VXVC]	762
fcmpu	BF, FRA, FRB	(FRA) is compared to (FRB). Place result in CR[BF] and FPSCR[FPCC].	CR[BF] FPSCR[FPCC, FX, VXSNAN]	763
fctiw	FRT, FRB	Convert (FRB) to a 32-bit signed integer. Place result in $FRT_{32:63}$. $FRT_{0:31}$ are undefined.	FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI] FPSCR[FPRF] undefined	764
fctiwz	FRT, FRB	Convert (FRB) to a 32-bit signed integer. Place result in $FRT_{32:63}$. $FRT_{0:31}$ are undefined.	FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI] FPSCR[FPRF] undefined	765
fdiv	FRT, FRA, FRB	Divide (FRA) by (FRB). Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]	766

Table A-1. PPC440 FPU Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
fdivs	FRT, FRA, FRB	Divide (FRA) by (FRB). Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]	767
fmadd	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	768
fmadds	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	769
fmsub	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	771
fmsubs	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	772
fmul	FRT, FRA, FRC	Multiply (FRA) by (FRC). Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]	773
fmuls	FRT, FRA, FRC	Multiply (FRA) by (FRC). Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]	774
fnabs	FRT, FRB	Place (FRB), with bit 0 set to one, into FRT.		775
fneg	FRT, FRB	Place (FRB), with bit 0 inverted, into FRT.		776
fnmadd	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Negate result and place it in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	777
fnmadds	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Negate result, round it to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	769
fnmsub	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Negate result and place it in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	779
fnmsubs	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Negate result, round it to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	780

Table A-1. PPC440 FPU Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
fres	FRT, FRB	A single-precision estimate of the reciprocal of the floating-point operand in FRB is placed into FRT.	FPSCR[FX, OX, UX, ZX, VXSNAN], FPSCR[FPRF, FR, FI] undefined	781
frsp	FRT, FRB	Round (FRB) to single precision and place the result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN]	782
frsqrte	FRT, FRB	A double-precision estimate of the reciprocal of the square root of the floating-point operand in FRB is placed into FRT.	FPSCR[FX, ZX, VXSNAN, VXSQRT], FPSCR[FPRF, FR, FI] undefined	783
fsel	FRT, FRA, FRC, FRB	The floating-point operand in FRA is compared to zero.		785
fsub	FRT, FRA, FRB	The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).		786
fsubs	FRT, FRA, FRB	The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).	FPSCR[FPRF, FR, FI FX, OX, UX, XX, VXSNAN]	787
lfd	FRT, D(RA)	Load the double word from EA = (RA|0) + EXTS(D) into FRT.		788
lfdu	FRT, D(RA)	Load the double word from EA = (RA|0) + EXTS(D) into FRT. Update the base address, (RA) ← EA		789
lfdux	FRT, RA, RB	Load the double word from EA = (RA|0) + (RB) into FRT. Update the base address, (RA) ← EA.		790
lfdx	FRT, RA, RB	Load the double word from EA = (RA|0) + (RB) into FRT.		791
lfs	FRT, D(RA)	Load the word from EA = (RA|0) + EXTS(D) into FRT. The word is interpreted as a single precision operand.		792
lfsu	FRT, D(RA)	Load the word from EA = (RA|0) + EXTS(D) into FRT. The word is interpreted as a single precision operand. Update the base address, (RA) ← EA.		793

Table A-1. PPC440 FPU Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lfsux	FRT, RA, RB	Load the word from EA = (RA|0) + (RB) into FRT. The word is interpreted as a single precision operand. Update the base address, (RA) ← EA.		794
lfsx	FRT, RA, RB	Load the word from EA = (RA|0) + (RB). The word is interpreted as a single precision operand.		795
mcrfs	BF, BFA	Copy the contents of the FPSCR specified by BFA to the CR field specified by BF. Place result in CR[BF] and FPSCR[FPCC].	CR[BF See the instruction description for details of changed FPSCR fields	796
mffs	FRT	Place (FPSCR) into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.		797
mtfsb0	BT	Set the FPSCR bit specified by BT to 0.	$FPSCR_{BT}$	798
mtfsb1	BT	Set the FPSCR bit specified by BT to 1.	$FPSCR_{BT}$	799
mtfsf	FLM, FRB	Place $FRB_{32:63}$ in the FPSCR under control of the mask specified by FLM.	$FPSCR_{BF}$	800
mtfsfi	FLM, FRB	Place $FRB_{32:63}$ in the FPSCR under control of the mask specified by FLM.	FPSCR fields altered by mask.	801
stfd	FRS, D(RA)	Store double word (FRS) at EA = (RA|0) + EXTS(D).		802
stfdu	FRS, D(RA)	Store double word (FRS) at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		803
stfdux	FRS, RA, RB	Store double word (FRS) at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		804
stfdx	FRS, RA, RB	Store double word (FRS) at EA = (RA|0) + (RB).		805
stfiwx	FRS, RA, RB	Store word $FRS_{32:63}$ at EA = (RA|0) + (RB).		806
stfs	FRS, D(RA)	Convert (FRS) to single-precision format and store at EA = (RA|0) + EXTS(D).		807
stfsu	FRS, D(RA)	Convert (FRS) to single-precision format and store at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		808
stfsux	FRS, RA, RB	Convert (FRS) to single-precision format and store at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		809
stfsx	RS, RA, RB	Convert (FRS) to single-precision format and store at EA = (RA|0) + (RB).		810

A.2 Instructions Sorted by Opcode

All instructions are four bytes long and word aligned. All instructions have a primary opcode field (shown as field OPCD in *Figure A-1* through *Figure A-10*, beginning on page 839) in bits 0:5. Some instructions also have a secondary opcode field (shown as field XO in *Figure A-1* through *Figure A-10*). PPC440 FPU instructions, sorted by primary and secondary opcode, are listed in *Table A-2*.

The "Form" indicated in the table refers to the arrangement of valid field combinations within the four-byte instruction. See *Instruction Formats* on page 836, for the field layouts of each form.

Form X has a 10-bit secondary opcode field, while form XO uses only the low-order 9-bits of that field. Form XO uses the high-order secondary opcode bit (the tenth bit) as a variable; therefore, every form XO instruction really consumes two secondary opcodes from the 10-bit secondary-opcode space. The implicitly consumed secondary opcode is listed in parentheses for form XO instructions in the table below.

Table A-2. PPC440 FPU Instructions by Opcode

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	535	X	**lfsx**	FRT, RA, RB	795
31	567	X	**lfsux**	FRT, RA, RB	794
31	599	X	**lfdx**	FRT, RA, RB	791
31	631	X	**lfdux**	FRT, RA, RB	790
31	695	X	**stfsux**	FRS, RA, RB	809
31	727	X	**stfdx**	FRS, RA, RB	805
31	759	X	**stfdux**	FRS, RA, RB	804
31	983	X	**stfiwx**	FRS, RA, RB	806
48		D	**lfs**	FRT, D(RA)	792
49		D	**lfsu**	FRT, D(RA)	793
50		D	**lfd**	FRT, D(RA)	788
51		D	**lfdu**	FRT, D(RA)	789
52		D	**stfs**	FRT, D(RA)	807
53		D	**stfsu**	FRT, D(RA)	808
54		D	**stfd**	FRS, D(RA)	802
55		D	**stfdu**	FRS, D(RA)	803
59	18	A	**fdivs**	FRT, FRA, FRB	767
59	20	A	**fsubs**	FRT, FRA, FRB	787
59	21	A	**fadds**	FRT, FRA, FRB	761
59	25	A	**fmuls**	FRT, FRA, FRC	774
59	28	A	**fmsubs**	FRT, FRA, FRC, FRB	772
59	29	A	**fmadds**	FRT, FRA, FRC, FRB	778

Table A-2. PPC440 FPU Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
59	30	A	**fnmsubs**	FRT, FRA, FRC, FRB	780
59	31	A	**fnmadds**	FRT, FRA, FRC, FRB	778
63	12	X	**frsp**	FRT, FRB	782
63	14	X	**fctiw**	FRT, FRB	764
63	15	X	**fctiwz**	FRT, FRB	765
63	18	A	**fdiv**	FRT, FRA, FRB	766
63	20	A	**fsub**	FRT, FRA, FRB	786
63	21	A	**fadd**	FRT, FRA, FRB	760
63	25	A	**fmul**	FRT, FRA, FRC	773
63	28	A	**fmsub**	FRT, FRA, FRC, FRB	771
63	29	A	**fmadd**	FRT, FRA, FRC, FRB	768
63	30	A	**fnmsub**	FRT, FRA, FRC, FRB	779
63	31	A	**fnmadds**	FRT, FRA, FRC, FRB	778
63	38	X	**mtfsb1**	BT	799
63	40	X	**fneg**	FRT, FRB	776
63	70	X	**mtfsb0**	BT	798
63	72	X	**fmr**	FRT, FRB	770
63	134	X	**mtfsfi**	BF, U	801
63	136	X	**fnabs**	FRT, FRB	775
63	264	X	**fabs**	FRT, FRB	759
63	583	X	**mffs**	FRT	797
63	711	XFL	**mtfsf**	BF, U	800

A.3 Instruction Formats

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions have an extended opcode in another field. Remaining instruction bits contain additional fields. All instruction fields belong to one of the following categories:

- Defined

  These instructions contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

These fields contain operands, such as GPR selectors and immediate values, that can vary from execution to execution. The instruction format diagrams specify the operands in the variable fields.

- Reserved

  Bits in reserved fields should be set to 0. In the instruction format diagrams, /, //, or /// indicate reserved fields.

If any bit in a defined field does not contain the expected value, the instruction is illegal and an illegal instruction exception occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid; its result is architecturally undefined. The PPC440 FPU executes all invalid instruction forms without causing an illegal instruction exception.

A.3.1 Instruction Fields

PPC440 FPU instructions contain various combinations of the following fields, as indicated in the instruction format diagrams that follow the field definitions. Numbers, enclosed in parentheses, that follow the field names indicate bit positions; bit fields are indicated by starting and stopping bit positions separated by colons.

AA (30) Absolute address bit.

- 0 The immediate field represents an address relative to the current instruction address (CIA). The effective address (EA) of the branch is either the sum of the LI field sign-extended to 32 bits and the branch instruction address, or the sum of the BD field sign-extended to 32 bits and the branch instruction address.
- 1 The immediate field represents an absolute address. The EA of the branch is either the LI field or the BD field, sign-extended to 32 bits.

BA (11:15) Specifies a bit in the CR used as a source of a CR-logical instruction.

BB (16:20) Specifies a bit in the CR used as a source of a CR-logical instruction.

BD (16:29) An immediate field specifying a 14-bit signed twos complement branch displacement. This field is concatenated on the right with 0b00 and sign-extended to 32 bits.

BF (6:8) Specifies a field in the CR used as a target in a compare or **mcrf** instruction.

BFA (11:13) Specifies a field in the CR used as a source in a **mcrf** instruction.

BI (11:15) Specifies a bit in the CR used as a source for the condition of a conditional branch instruction.

BO (6:10) Specifies options for conditional branch instructions.

BT (6:10) Specifies a bit in the CR used as a target as the result of a CR-Logical instruction.

D (16:31) Specifies a 16-bit signed twos-complement integer displacement for load/store instructions.

DCRN (11:20) Specifies a device control register (DCR).

FXM (12:19) Field mask used to identify CR fields to be updated by the **mtcrf** instruction.

IM (16:31) An immediate field used to specify a 16-bit value (either signed integer or unsigned).

LI (6:29) An immediate field specifying a 24-bit signed twos complement branch displacement; this field is concatenated on the right with b'00' and sign-extended to 32 bits.

LK (31) Link bit.

- 0 Do not update the link register (LR).
- 1 Update the LR with the address of the next instruction.

MB (21:25) Mask begin.

Used in rotate-and-mask instructions to specify the beginning bit of a mask.

ME (26:30) Mask end.

Used in rotate-and-mask instructions to specify the ending bit of a mask.

NB (16:20) Specifies the number of bytes to move in an immediate string load or store.

OPCD (0:5) Primary opcode. Primary opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The OPCD field name does not appear in instruction descriptions.

OE (21) Enables setting the OV and SO fields in the fixed-point exception register (XER) for extended arithmetic.

RA (11:15) A GPR used as a source or target.

RB (16:20) A GPR used as a source.

Rc (31) Record bit.

0 Do not set the CR.

1 Set the CR to reflect the result of an operation.

RS (6:10) A GPR used as a source.

RT (6:10) A GPR used as a target.

SH (16:20) Specifies a shift amount.

SPRF (11:20) Specifies a special purpose register (SPR).

TO (6:10) Specifies the conditions on which to trap, as described under **tw** and **twi** instructions.

XO (21:30) Extended opcode for instructions without an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

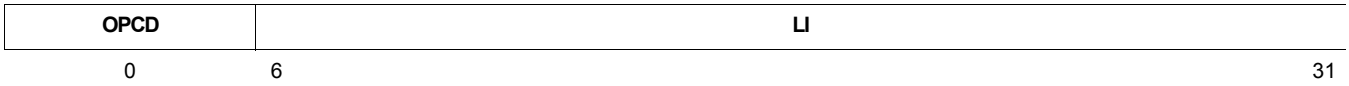
XO (22:30) Extended opcode for instructions with an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

A.3.2 Instruction Format Diagrams

The instruction formats (also called "forms") illustrated in *Figure A-1* through *Figure A-10* are valid combinations of instruction fields. *Table A-2* on page 835 indicates which "form" is utilized by each PPC440 FPU opcode. Fields indicated by slashes (/, //, or ///) are reserved. The figures are adapted from the PowerPC User Instruction Set Architecture.

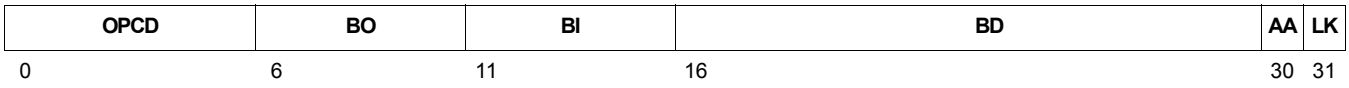
A.3.2.1 I-Form

Figure A-1. I Instruction Format



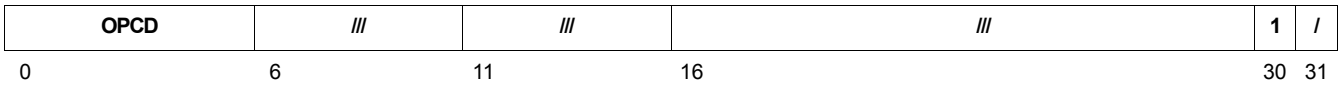
A.3.2.2 B-Form

Figure A-2. B Instruction Format



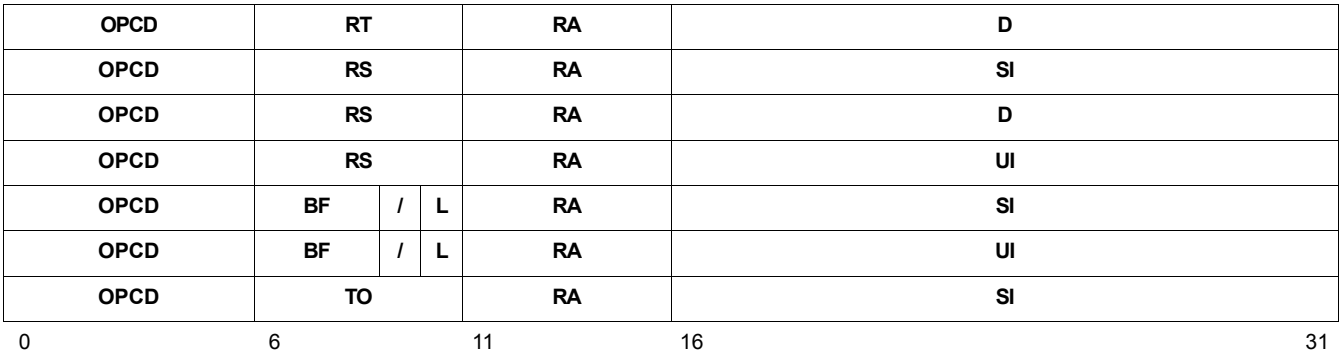
A.3.2.3 SC-Form

Figure A-3. SC Instruction Format



A.3.2.4 D-Form

Figure A-4. D Instruction Format



A.3.2.5 X-Form

Figure A-5. X Instruction Format

OPCD	RT	RA	RB	XO	Rc
OPCD	RT	RA	RB	XO	/
OPCD	RT	RA	NB	XO	/
OPCD	RT	RA	WS	XO	/
OPCD	RT	///	RB	XO	/
OPCD	RT	///	///	XO	/
OPCD	RS	RA	RB	XO	Rc
OPCD	RS	RA	RB	XO	1
OPCD	RS	RA	RB	XO	/
OPCD	RS	RA	NB	XO	/
OPCD	RS	RA	WS	XO	/
OPCD	RS	RA	SH	XO	Rc
OPCD	RS	RA	///	XO	Rc
OPCD	RS	///	RB	XO	/
OPCD	RS	///	///	XO	/
OPCD	BF / L	RA	RB	XO	/
OPCD	BF //	BFA //	///	XO	Rc
OPCD	BF //	///	///	XO	/
OPCD	BF //	///	U	XO	Rc
OPCD	BF //	///	///	XO	/
OPCD	TO	RA	RB	XO	/
OPCD	BT	///	///	XO	Rc
OPCD	///	RA	RB	XO	/
OPCD	///	///	///	XO	/
OPCD	///	///	E //	XO	/
0	6	11	16	21	31

A.3.2.6 XL-Form

Figure A-6. XL Instruction Format

OPCD	BT	BA	BB	XO	/
OPCD	BC	BI	///	XO	LK
OPCD	BF //	BFA //	///	XO	/
OPCD	///	///	///	XO	/
0	6	11	16	21	31

A.3.2.7 XFL-Form

Figure A-7. XFX Instruction Format

OPCD	/	FLM	/	FRB	XO	Rc
0	6	15	16	21		31

A.3.2.8 XFX-Form

Figure A-8. XFX Instruction Format

OPCD	RT	SPRF	XO	/	
OPCD	RT	DCRF	XO	/	
OPCD	RT	/ FXM /	XO	/	
OPCD	RS	SPRF	XO	/	
OPCD	RS	DCRF	XO	/	
0	6	11 16	21	31	

A.3.2.9 X0-Form

Figure A-9. XO Instruction Format

OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	///	/	XO	Rc
0	6	11	16	21	22	31

A.3.2.10 M-Form

Figure A-10. M Instruction Format

OPCD	RS	RA	RB	MB	ME	Rc
OPCD	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

Index

A

addressing 137, 159
addressing modes 60, 160
ANSI/IEEE Standard 754-1985 159

B

bits
  guard (G) 172
  round (R) 172
  sticky (X) 172
boot from NDFC 494
bootstrap controller 56
Buffer Descriptor Status/Control Fields 556
bus timeout error 473

C

chip reset results 189
clock and power management 56, 301
  registers 301
clocking 283
  registers 294
  system 284
    bypass PLL 288
    choosing system clock ratios 289
    clocks for offchip use 287
    CPU feedback example 289, 290
    CPU PLB frequency 288
    feedback selection 285
    IIC bootstrap controller clocking 287
    input system clock 284
    M value for SYS PLL 285
    PerClk feedback example 291
    SYS PLL strapping 288
    SYS PLL TUNE setting 287
    system clock ratio examples 289
    VCO frequency 285
computional instructions 175
condition register 270
condition registers 61
conventions
  notational 45
core reset results 189
CoreConnect features 53
CPM0_ER 301
CPM0_FR 303
CPM0_SR 304
CPR0 registers 195
CPR0_CFGADDR 294
CPR0_CFGDATA 294
CPR0_CLKUPD 295
CPR0_ICFG 196
CPR0_MALD 299
CPR0_OPBD0 298
CPR0_PERD0 298
CPR0_PLLC0 295
CPR0_PLLD0 296
CPR0_PRIMAD0 297
CPR0_PRIMBD0 298
CPR0_SPCID 299

D

data addressing modes 160
data storage addressing modes 160
DCRs 139
  defined 811
DDR_SDRAM 56, 311
  commands and operations 347
  device configuration 314
  DIMM support 355
  error checking and correction 355
  initialization 342
  interface signals 311
  page management 344
  PLB slave interface options 343
  PLB to memory address code 342
  power management 360
  registers
    address map 314
denormalization 169
development tool support 62
device control registers 61, 139, 811
Device Control Registers. *See also* DCRs
device control registers. See DCRs 139
device-paced transfers 472
  bus timeout error 473
diagram, USB interface block 694, 696
direct memory access (DMA) 57
DMA
  memory-to-memory mode transfers
    initiated by software 544
DMA controller 509
  configuration and status registers 511
  transfers 509
DMA operations
  arbitration transfer priorities 536
  channel priorities 519
  data parity 519, 537
  errors 520, 537
  interrupts 521, 538
  peripheral and deviced paced memory bursts 519
  programming 523
  scatter gather transfers 522
DMA2P30_ADR 536
DMA2P30_CRx 532
DMA2P30_CTx 534
DMA2P30_DAx 534
DMA2P30_SAx 534

DMA2P30_SCx 536
DMA2P30_SGC 535
DMA2P30_SGx 535
DMA2P30_SLP 531
DMA2P30_SR 531
DMA2P40_CRx 513
DMA2P40_CTx 514
DMA2P40_DAHx 516
DMA2P40_DALx 516
DMA2P40_POL 519
DMA2P40_SAHx 515
DMA2P40_SALx 515
DMA2P40_SGC 518
DMA2P40_SGHx 516
DMA2P40_SGLx 516
DMA2P40_SLP 518
DMA2P40_SR 517
duration counter logic 115

E

EBC
  signals 465
EBC error reporting 488
EBC registers 483
EBC0_BEAR 489
EBC0_BESR0 489
EBC0_BESR1 490
EBC0_BxAP 487
EBC0_BxCR 486
EBC0_CFG 485
ECC 496
EMAC programming notes 626
EMAC to PHY bridge 58, 575
EMACx_GAHTx 621
EMACx_IAHR 619
EMACx_IAHTx 621
EMACx_IALR 620
EMACx_IPGVR 622
EMACx_ISER 618
EMACx_ISR 616
EMACx_LSAH 621
EMACx_LSAL 622
EMACx_MR0 611
EMACx_MR1 612
EMACx_OCRX 625
EMACx_OCTX 625
EMACx_PTR 621
EMACx_RMR 615
EMACx_RWMR 624
EMACx_STACR 622
EMACx_TMR0 613
EMACx_TMR1 614
EMACx_TRTR 623
EMACx_VTCI 620
EMACx_VTPID 620
error correction code (implemented in NDFC) 496
error handling, PCI interface 403
error reporting, EBC 488
Ethernet MAC 583
  features 584
  flow control 601
  MAL-MAC packet transfer flow 625
  MII 625
  operations 585
  receive operation 599
  registers 609
  transmit operation 593
  VLAN support 604
ethernet media access controller 58
exception
  floating-point 161
  inexact 161, 180, 260, 267
  invalid operation 263
  overflow 161, 180, 260, 265
  underflow 161, 180, 260, 266
  updating the CR 270
  updating the FPR 269
  zero divide 161, 260, 265
exception priorities for floating-point load and store instructions 268
exception syndrome register (ESR) 259
exceptions 257
execution model
  multply-add 64-bit 174
execution model for multiply-add type instructions 174
external bus controller 57
  signals 465
external bus master 477
  arbitration 478
  interface 477

F

fabs 759
fadd 760
fadds 761
fcmpo 762
fctiw 764
fctiwz 765
fdiv 766
fdivs 767
features 49
FEX 163
field
  n 270
Flash controller, NAND 493
floating point registers 162
floating point unit 55
floating point unit features 49
floating-point
  denormalization 169
  denormalized number 167
  infinity 167
  normalization 169
  Not a Number 168

sign 168
zero 167
floating-point accumulator 174
floating-point compare and select instruction set index 181
floating-point compare instructions
comparison sets 181
floating-point instructions 758
floating-point multiply-add instructions 180
floating-point operands 176
double precision format 176
single format 176
floating-point programming model 159
floating-point rounding and conversion instruction set index 181
floating-point status and control register 181
instruction set index 182
floating-point unavailable interrupt 262
fmadd 768
fmadds 769
fmr 770
fmsub 771
fmsubs 772
fmsubs. 772
fmul 773
fmuls 774
fnabs 775
fneg 776
fnmadd 777
fnmadds 778
fnmsub 779
fnmsubs 780
fpcmpu 763
FPR0:FPR31 162
FPSCR 162
fres 781
fres. 781
frsp 782
frsqrte 783
fsel 785
fsub 786
fsubs 787
functional block diagrams 112

G

general purpose registers 61
general purpose timers 58, 275
registers 276
generic pipeline event counter logic 114
GPIO operations 59
GPIO0_IRO 679
GPT 58
GPT0_COMP0:GPT0_COMP6 280
GPT0_DCIS 281
GPT0_DCT0 280
GPT0_IE 279
GPT0_IM 278
GPT0_ISS, GPT0_ISC 278
GPT0_MASK0:GPT0_MASK6 280
GPT0_TBC 277

H

HcBulkCurrentED 712
HcBulkHeadED 711
HcCommandStatus 703
HcControl 701
HcControlCurrentED 710
HcControlHeadED 709
HcDoneHeadED 713
HcFmInterval 714
HcFmNumber 716
HcFmRemaining 715
HcHCCA 707
HcInterruptDisable 706
HcInterruptEnable 705
HcInterruptStatus 704
HcLSThreshold 718
HcPeriodCurrentED 708
HcPeriodicStart 717
HcRevision 700
HcRhDescriptorA 719
HcRhDescriptorB 720
HcRhPortStatus 723
HcRhStatus 721

I

IIC bootstrap controller 207
configuration 207
IIC bus 59
IIC bus interface 659
addressing modes 659
interrupt handling 676
registers 661
IICx_CLKDIV 670
IICx_CNTL 664
IICx_DIRECTCNTL 674
IICx_EXTSTS 668
IICx_HMADR 664
IICx_HSADR 670
IICx_INTRMSK 671
IICx_LMADR 663
IICx_LSADR 669
IICx_MDBUF 662
IICx_MDCNTL 665
IICx_SDBUF 662
IICx_STS 666
IICx_XFRCNT 672
IICx_XTCNTLSS 673
initialization 195
instruction fields 837
instruction formats 755, 836
diagrams 838
instruction forms 836, 838

instruction set portability 755
instructions
alphabetical, including extended mnemonics 831
by category 176
categories 755
computational 175
fabs 759
fadd 760
fadds 761
fcmpo 762
fcmpu 763
fctiw 764
fctiwz 765
fdiv 766
fdivs 767
fmadd 768
fmadds 769
fmr 770
fmsub 771
fmsubs 772
fmsubs. 772
fmul 773
fmuls 774
fnabs 775
fneg 776
fnmadd 777
fnmadds 778
fnmsub 779
fnmsubs 780
format diagrams 838
formats 836
forms 836, 838
fres 781
fres. 781
frsp 782
frsqrte 783
fsel 785
fsub 786
fsubs 787
lfd 788
lfdu 789
lfdux 790
lfdx 791
lfs 792
lfsu 793
lfsux 794
lfsx 795
mcrfs 796
mffs 797
mtfsb0 798
mtfsb0. 798
mtfsb1 799
mtsfs 800
mtsfsi 801
noncomputational 175
opcodes 835
stfd 802
stfdu 803
stfdux 804
stfdx 805
stfiwx 806
stfs 807
stfsu 808
stfsx 810
inter integrated circuit bus 59
internal buses 55
interrupt
type
floating-point unavailable 262
interrupts 257
invalid operation exception (SNaN) 260
invalid operation exception bit 180

J

JTAG 56

L

lfd 788
lfdu 789
lfdux 790
lfdx 791
lfs 792
lfsu 793
lfsux 794
lfsx 795
linear data access region 507

M

machine state registers 61
MAL 57
channel active set and reset registers 569
end of buffer interrupt status register 570
error handling 560
initialization 559
programming notes 559
receive status/control field format 557, 558
transmit status/control field format 557
MAL interrupts 560
MAL0_CFG 567
MAL0_ESR 570
MAL0_IER 572
MAL0_RCBSx 574
MAL0_RXCTPnR 573
MAL0_RXDEIR 572
MAL0_RXEOBISR 570
MAL0_TXCASR 569
MAL0_TXCTPnR 573
MAL0_TXDEIR 572
MAL0_TXEOBISR 570
master event counter logic 113
mcrfs 796
memory access layer 57, 545

buffer descriptor 550
descriptor buffer status fields, descriptor buffer control fields 556
features 545
interfaces and channel assignments 547
receive software interface 555
transmit and receive operations 548
transmit software interface 552
memory interface
bus attachment
alternative 466
external bus master 477
SRAM
burst mode 469
bus timeout error 473
device-paced transfers 472
memory management. *See also* MMU
memory map 137, 159
memory map, NDFC register 498
memory mapped registers 62, 149
memory organization 137, 159
mffs 797
monitoring PLB events 116
mtfsb0 798
mtfsb0. 798
mtfsb1 799
mtsfs 800
mtsfsi 801

N

NAND Flash controller 57, 493
NAND linear data access region 507
NDFC interface
overview 493
NDFC registers 497
NDFC, boot from 494
NDFC0_ADDR 499
NDFC0_B0CR:NDFC0_B3CR 502
NDFC0_CMD 499
NDFC0_CR 503
NDFC0_DATA 500
NDFC0_ECC0:NDFC0_ECC7 501
NDFC0_HWCTL 505
NDFC0_REVID 506
NDFC0_SR 505
noncomputional instructions 175
notation 45, 756, 837
notational conventions 45

O

on-chip buses 63
DCR bus 109
OPB arbiter registers 106
OPB bus 105
OPB master assignments 105
OPB to PLB3 registers 107
PLB4 arbiter registers 73
PLB4 to OPB Bridge registers 94
processor local bus 63
master and slave assignments 64
master priority assignments 64
OPB2PLB30_BCTRL 108
OPB2PLB30_BSTAT 108
OPB2PLB30_REVID 108
OPBAx_CR 107
OPBAx_PR 106
OPBMC 725
opcodes 835

P

P3P4BI0_BEARH 87
P3P4BI0_BEARL 87
P3P4BI0_BESR0 87
P3P4BI0_BESR1 90
P3P4BI0_CFG 91
P3P4BI0_PEIR 93
P3P4BI0_PICR 92
P3P4BI0_REVID 93
P4P3BO0_BEARH 80
P4P3BO0_BEARL 80
P4P3BO0_BESR0 81
P4P3BO0_BESR1 83
P4P3BO0_CFG 84
P4P3BO0_PEIR 86
P4P3BO0_PICR 85
P4P3BO0_REVID 86
PCI bridge 56
PCI bridge completion ordering rules 377
PCI bridge configuration registers 379
PCI bridge register summary 379
PCI bridge reset and initialization 408
PCI bridge timing diagrams 411
PCI configuration registers
offsets 380
PCI interface 363
arbiter 366
block diagram 364
bridge address mapping 367
bridge functional blocks 365
bridge transaction handling 370
byte ordering 364
error handling 403
master commands 371
target map configuration 370
PCI power management interface 407
PCIC0_BIST 391
PCIC0_BRDGOPT1 396
PCIC0_BRDGOPT2 401
PCIC0_CACHELS 390
PCIC0_CAP 393
PCIC0_CAPID 399
PCIC0_CFGADDR 386

PCIC0_CFGDATA 387
PCIC0_CLS 390
PCIC0_CMD 388
PCIC0_DATA 401
PCIC0_DEVID 387
PCIC0_ERREN 394
PCIC0_ERRSTS 395
PCIC0_HDTYPE 391
PCIC0_ICS 394
PCIC0_INTLN 393
PCIC0_INTPN 393
PCIC0_LATTIM 390
PCIC0_MAXLTNCY 394
PCIC0_MINGNT 394
PCIC0_NEXTIPTR 400
PCIC0_PLBBEAR 399
PCIC0_PLBBESR0 396
PCIC0_PLBBESR1 398
PCIC0_PMC 400
PCIC0_PMCSR 400
PCIC0_PMCSRBSE 401
PCIC0_PMSCRR 402
PCIC0_PTM1BAR 391
PCIC0_PTM2BAR 392
PCIC0_REVID 390
PCIC0_SBSYSID 393
PCIC0_SBSYSVID 393
PCIC0_STATUS 388
PCIC0_VENDID 387
PCIL0_PMM0LA 381
PCIL0_PMM0MA 381
PCIL0_PMM0PCIHA 382
PCIL0_PMM0PCILA 382
PCIL0_PMM1LA 382
PCIL0_PMM1MA 382
PCIL0_PMM1PCIHA 383
PCIL0_PMM1PCILA 383
PCIL0_PMM2LA 383
PCIL0_PMM2MA 384
PCIL0_PMM2PCIHA 384
PCIL0_PMM2PCILA 384
PCIL0_PTM1LA 385
PCIL0_PTM1MS 385
PCIL0_PTM2LA 386
PCIL0_PTM2MS 385
PCI-to-PLB half-bridge 366
PCI-to-PLB transaction handling 374
peripheral component interconnect interface 363
peripheral features 50
peripheral functions and interfaces 55
PLB master commands 375
PLB performance monitor 111
  features 111
  monitoring PLB events 116
  registers 119
PLB3 arbiter
  registers 98
PLB3 to OPB bridge
  registers 102
PLB3 to OPB bridge error status register 0 102
PLB3 to OPB bridge error status register 1 104
PLB3 to PLB4 bridge
  registers 86
PLB32OPB0_BEAR 104
PLB32OPB0_BESR0 102
PLB32OPB0_BESR1 104
PLB3A0_ACR 101
PLB3A0_BEAR 100
PLB3A0_BESR 99
PLB3A0_REVID 98
PLB4 to PLB3 bridge
  registers 80
PLB42OPB0_BEARH 96
PLB42OPB0_BEARL 96
PLB42OPB0_BESR0 94
PLB42OPB0_BESR1 96
PLB42OPB0_CFG 97
PLB42OPB0_LATENCY 98
PLB42OPB0_REVID 98
PLB4A0_CCR 79
PLB4A0_REVID 73
PLB4An_ACR 73
PLB4An_EARH 78
PLB4An_EARL 78
PLB4An_ESRL 77
PLB-to-PCI half-bridge 365
portability, instruction set 755
PPC440GP
  signals 829
PPM0_CCR 124
PPM0_CFGADDR 121
PPM0_CFGDATA 121
PPM0_CR 123
PPM0_DCMNR0:PPM0_DCMNR1 132
PPM0_DCMXR0:PPM0_DCMXR1 132
PPM0_DCOTR0:PPM0_DCOTR1 133
PPM0_DCSR0:PPM0_DCSR1 131
PPM0_DCTVR0:PPM0_DCTVR1 133
PPM0_GCR0:PPM0_GCR3 131
PPM0_GCSR0:PPM0_GCSR3 130
PPM0_ISR 121
PPM0_LAMR 126
PPM0_LAR 125
PPM0_MCR0:PPM0_MCR3 130
PPM0_MCSR0:PPM0_MCSR3 126
PPM0_RIDR 126
PPM0_SCR0:PPM0_SCR3 130
PPM0_SCSR0:PPM0_SCSR3 128
PPM0_UAMR 125
PPM0_UAR 125
primary opcodes 835
processor core features 49
pseudocode 756

Q

QNaN 269

R

registers 60, 161
clock and power management 301
clocking 294
CPM0_ER 301
CPM0_FR 303
CPM0_SR 304
CPR0_CFGADDR 294
CPR0_CFGDATA 294
CPR0_CLKUPD 295
CPR0_ICFG 196
CPR0_MALD 299
CPR0_OPBD0 298
CPR0_PERD0 298
CPR0_PLLC0 295
CPR0_PLLD0 296
CPR0_PRIMAD0 297
CPR0_PRIMBD0 298
CPR0_SPCID 299
DCR 139
DMA2P30_ADR 536
DMA2P30_CRx 532
DMA2P30_CTx 534
DMA2P30_DAx 534
DMA2P30_POL 530
DMA2P30_SAx 534
DMA2P30_SCx 536
DMA2P30_SGC 535
DMA2P30_SGx 535
DMA2P30_SLP 531
DMA2P30_SR 531
DMA2P40_CRx 513
DMA2P40_CTx 514
DMA2P40_DAHx 516
DMA2P40_DALx 516
DMA2P40_POL 519
DMA2P40_SAHx 515
DMA2P40_SALx 515
DMA2P40_SGC 518
DMA2P40_SGHx 516
DMA2P40_SGLx 516
DMA2P40_SLP 518
DMA2P40_SR 517
EBC 483
EBC0_BEAR 489
EBC0_BESR0 489
EBC0_BESR1 490
EBC0_BxAP 487
EBC0_BxCR 486
EBC0_CFG 485
EBC0_CFGADDR 483
EBC0_CFGDATA 483
EMACx_GAHTx 621
EMACx_IAHR 619
EMACx_IAHTx 621
EMACx_IALR 620
EMACx_IPGVR 622
EMACx_ISER 618
EMACx_ISR 616
EMACx_LSAH 621
EMACx_LSAL 622
EMACx_MR0 611
EMACx_MR1 612
EMACx_OCRX 625
EMACx_OCTX 625
EMACx_PTR 621
EMACx_RMR 615
EMACx_RWMR 624
EMACx_STACR 622
EMACx_TMR0 613
EMACx_TMR1 614
EMACx_TRTR 623
EMACx_VTCI 620
EMACx_VTPID 620
error handling 562
Ethernet MAC 609
FPR0:FPR31 162
FPSCR 162
GPT0_COMP0:GPT0_COMP6 280
GPT0_DCIS 281
GPT0_DCT0 280
GPT0_IE 279
GPT0_IM 278
GPT0_ISS, GPT0_ISC 278
GPT0_MASK0:GPT0_MASK6 280
GPT0_TBC 277
HcBulkCurrentED 712
HcBulkHeadED 711
HcCommandStatus 703
HcControl 701
HcControlCurrentED 710
HcControlHeadED 709
HcDoneHeadED 713
HcFmInterval 714
HcFmNumber 716
HcFmRemaining 715
HcHCCA 707
HcInterruptDisable 706
HcInterruptEnable 705
HcInterruptStatus 704
HcLSThreshold 718
HcPeriodCurrentED 708
HcPeriodicStart 717
HcRevision 700
HcRhDescriptorA 719
HcRhDescriptorB 720
HcRhPortStatus 723
HcRhStatus 721
IIC bus interface 661
IICx_CLKDIV 670
IICx_CNTL 664
IICx_DIRECTCNTL 674
IICx_EXTSTS 668
IICx_HMADR 664
IICx_HSADR 670
IICx_INTRMSK 671
IICx_LMADR 663
IICx_LSADR 669

IICx_MDBUF 662
IICx_MDCNTL 665
IICx_SDBUF 662
IICx_STS 666
IICx_XFRCNT 672
IICx_XTCNTLSS 673
MAL 567
MAL device control 566
MAL0_CFG 567
MAL0_ESR 570
MAL0_IER 572
MAL0_RCBSx 574
MAL0_RXCTPnR 573
MAL0_RXDEIR 572
MAL0_RXEOBISR 570
MAL0_TXCASR 569
MAL0_TXCTPnR 573
MAL0_TXDEIR 572
MAL0_TXEOBISR 570
NDFC0_ADDR 499
NDFC0_B0CR:NDFC0_B3CR 502
NDFC0_CMD 499
NDFC0_CR 503
NDFC0_DATA 500
NDFC0_ECC0:NDFC0_ECC7 501
NDFC0_HWCTL 505
NDFC0_REVID 506
NDFC0_SR 505
OPB2PLB30_BCTRL 108
OPB2PLB30_BSTAT 108
OPB2PLB30_REVID 108
OPBAx_CR 107
OPBAx_PR 106
OPBMC 725
P3P4BI0_BEARH 87
P3P4BI0_BEARL 87
P3P4BI0_BESR0 87
P3P4BI0_BESR1 90
P3P4BI0_CFG 91
P3P4BI0_PEIR 93
P3P4BI0_PICR 92
P3P4BI0_REVID 93
P4P3BO0_BEARH 80
P4P3BO0_BEARL 80
P4P3BO0_BESR0 81
P4P3BO0_BESR1 83
P4P3BO0_CFG 84
P4P3BO0_PEIR 86
P4P3BO0_PICR 85
P4P3BO0_REVID 86
PCIC0_BIST 391
PCIC0_BRDGOPT1 396
PCIC0_BRDGOPT2 401
PCIC0_CACHELS 390
PCIC0_CAP 393
PCIC0_CAPID 399
PCIC0_CFGADDR 386
PCIC0_CFGDATA 387
PCIC0_CLS 390
PCIC0_CMD 388
PCIC0_DATA 401
PCIC0_DEVID 387
PCIC0_ERREN 394
PCIC0_ERRSTS 395
PCIC0_HDTYPE 391
PCIC0_ICS 394
PCIC0_INTLN 393
PCIC0_INTPN 393
PCIC0_LATTIM 390
PCIC0_MAXLTNCY 394
PCIC0_MINGNT 394
PCIC0_NEXTIPTR 400
PCIC0_PLBBEAR 399
PCIC0_PLBBESR0 396
PCIC0_PLBBESR1 398
PCIC0_PMC 400
PCIC0_PMCSR 400
PCIC0_PMCSRBSE 401
PCIC0_PMSCRR 402
PCIC0_PTM1BAR 391
PCIC0_PTM2BAR 392
PCIC0_REVID 390
PCIC0_SBSYSID 393
PCIC0_SBSYSVID 393
PCIC0_STATUS 388
PCIC0_VENDID 387
PCIL0_PMM0LA 381
PCIL0_PMM0MA 381
PCIL0_PMM0PCIHA 382
PCIL0_PMM0PCILA 382
PCIL0_PMM1LA 382
PCIL0_PMM1MA 382
PCIL0_PMM1PCIHA 383
PCIL0_PMM1PCILA 383
PCIL0_PMM2LA 383
PCIL0_PMM2MA 384
PCIL0_PMM2PCIHA 384
PCIL0_PMM2PCILA 384
PCIL0_PTM1LA 385
PCIL0_PTM1MS 385
PCIL0_PTM2LA 386
PCIL0_PTM2MS 385
PLB performance monitor 119
PLB3 arbiter 98
PLB3 to OPB bridge 102
PLB3 to PLB4 bridge 86
PLB32OPB0_BEAR 104
PLB32OPB0_BESR0 102
PLB32OPB0_BESR1 104
PLB3A0_ACR 101
PLB3A0_BEAR 100
PLB3A0_BESR 99
PLB3A0_REVID 98
PLB4 to PLB3 bridge 80
PLB42OPB0_BEARH 96
PLB42OPB0_BEARL 96
PLB42OPB0_BESR0 94
PLB42OPB0_BESR1 96

PLB42OPB0_CFG 97
PLB42OPB0_LATENCY 98
PLB42OPB0_REVID 98
PLB4A0_CCR 79
PLB4A0_REVID 73
PLB4An_ACR 73
PLB4An_EARH 78
PLB4An_EARL 78
PLB4An_ESRL 77
PPM0_CCR 124
PPM0_CFGADDR 121
PPM0_CFGDATA 121
PPM0_CR 123
PPM0_DCMNR0:PPM0_DCMNR1 132
PPM0_DCMXR0:PPM0_DCMXR1 132
PPM0_DCOTR0:PPM0_DCOTR1 133
PPM0_DCSR0:PPM0_DCSR1 131
PPM0_DCTVR0:PPM0_DCTVR1 133
PPM0_GCR0:PPM0_GCR3 131
PPM0_GCSR0:PPM0_GCSR3 130
PPM0_ISR 121
PPM0_LAMR 126
PPM0_LAR 125
PPM0_MCR0:PPM0_MCR3 130
PPM0_MCSR0:PPM0_MCSR3 126
PPM0_RIDR 126
PPM0_SCR0:PPM0_SCR3 130
PPM0_SCSR0:PPM0_SCSR3 128
PPM0_UAMR 125
PPM0_UAR 125
SDR0_AMP0 66
SDR0_AMP1 67
SDR0_CP440 68
SDR0_CUST0 221
SDR0_CUST1 222
SDR0_DDRDL0 336
SDR0_EBC0 222
SDR0_ECID0 197
SDR0_ECID1 197
SDR0_ECID2 197
SDR0_ECID3 197
SDR0_EMACxREJCNT 592
SDR0_EMACxRXST 589
SDR0_EMACxTXST 591
SDR0_HSF 201
SDR0_MALRBL 566
SDR0_MFR 200
SDR0_MIRQ0 69
SDR0_PCI0 378
SDR0_PFC0 197
SDR0_PFC1 198
SDR0_PINSTP 217
SDR0_SDCS0 217
SDR0_SDSTP0 218
SDR0_SDSTP1 219
SDR0_SDSTP2 220
SDR0_SDSTP3 221
SDR0_SLPIPE0 71, 199
SDR0_SRSTx 199
SDR0_UART0 640
SDR0_UART1 641
SDR0_UART2 642
SDR0_UART3 642
SDR0_USB0 695
SDRAM0_B0CR:SDRAM0_B3CR 325
SDRAM0_BEAR 319
SDRAM0_BESR0 317
SDRAM0_BESR1 318
SDRAM0_CFG0 321
SDRAM0_CFG1 322
SDRAM0_CID 341
SDRAM0_CLKTR 334
SDRAM0_DEVOPT 323
SDRAM0_DLYCAL 339
SDRAM0_ECCESR 340
SDRAM0_MCSTS 323
SDRAM0_MIRQ 319
SDRAM0_PMIT 324
SDRAM0_RID 341
SDRAM0_RTR 324
SDRAM0_SLIO 320
SDRAM0_TR0 326
SDRAM0_TR1 328
SDRAM0_UABBA 324
SDRAM0_WDDCTR 337
serial peripheral interface 655
SPI0_CDM 657
SPI0_CR 656
SPI0_MODE 658
SPI0_RxD 656
SPI0_SR 657
SPI0_TxD 656
types 162
  DCR 811
UART controller 632
UARTx_DLL 639
UARTx_DLM 639
UARTx_FCR 635
UARTx_IER 633
UARTx_IIR 634
UARTx_LCR 635
UARTx_LSR 637
UARTx_MCR 636
UARTx_MSR 638
UARTx_RBR 633
UARTx_SCR 639
UARTx_THR 633
UIC controller 226
UIC0_CR 235
UIC0_ER 230
UIC0_MSR 248
UIC0_PR 239
UIC0_SR 226
UIC0_TR 244
UIC0_VCR 253
UIC0_VR 254
UIC1_CR 237
UIC1_ER 233

UIC1_MSR 250
UIC1_PR 241
UIC1_SR 228
UIC1_TR 246
UIC1_VCR 253
UIC1_VR 255
USB2D0_FADDR 733
USB2D0_FRAME 739
USB2D0_INCSR 742
USB2D0_INCSR0 741
USB2D0_INDEX 739
USB2D0_INMAXP 744
USB2D0_INTRIN 731
USB2D0_INTRINE 734
USB2D0_INTROUT 734
USB2D0_INTROUTE 737
USB2D0_INTRUSB 736
USB2D0_INTRUSBE 735
USB2D0_OUTCOUNT0 748
USB2D0_OUTCSR 745
USB2D0_OUTMAXP 747
USB2D0_POWER 731
USB2D0_TESTMODE 738
ZMII bridge 579
ZMII0_FER 579
ZMII0_SMIISR 581
ZMII0_SSR 580
registers, device control 139, 811
registers, memory mapped 149
registers, summary 161
reset 55
reset types
chip 189
core 189
system 190
resets
signals 189
types 189

S

SDR0 registers 195
SDR0_AMP0 66
SDR0_AMP1 67
SDR0_CP440 68
SDR0_CUST0 221
SDR0_CUST1 222
SDR0_DDRDL0 336
SDR0_EBC0 222
SDR0_ECID0 197
SDR0_ECID2 197
SDR0_ECID3 197
SDR0_EMACxREJCNT 592
SDR0_EMACxRXST 589
SDR0_EMACxTXST 591
SDR0_HSF 201
SDR0_MALRBL 566
SDR0_MFR 200
SDR0_MIRQ0 69
SDR0_PCI0 378
SDR0_PFC0 197
SDR0_PFC1 198
SDR0_PINSTP 217
SDR0_SCID1 197
SDR0_SDCS0 217
SDR0_SDSTP0 218
SDR0_SDSTP1 219
SDR0_SDSTP2 220
SDR0_SDSTP3 221
SDR0_SLPIPE0 71, 199
SDR0_SRSTx 199
SDR0_UART0 640
SDR0_UART1 641
SDR0_UART2 642
SDR0_UART3 642
SDR0_USB0 695
SDRAM0_B0CR:SDRAM0_B3CR 325
SDRAM0_BEAR 319
SDRAM0_BESR0 317
SDRAM0_BESR1 318
SDRAM0_CFG0 321
SDRAM0_CFG1 322
SDRAM0_CID 341
SDRAM0_CLKTR 334
SDRAM0_DEVOPT 323
SDRAM0_DLYCAL 339
SDRAM0_ECCESR 340
SDRAM0_MCSTS 323
SDRAM0_MIRQ 319
SDRAM0_PMIT 324
SDRAM0_RID 341
SDRAM0_RTR 324
SDRAM0_SLIO 320
SDRAM0_TR0 326
SDRAM0_TR1 328
SDRAM0_UABBA 324
SDRAM0_WDDCTR 337
secondary opcodes 835
serial peripheral interface 59, 653
exchange of data 655
interrupt operation 654
loopback operation 654
registers 655
serial port operations 59
UART controller 629
signals 62, 829
EBC 465
resets 189
slave event counter logic 113
special purpose registers 61
SPI 653
SPI0_CDM 657
SPI0_CR 656
SPI0_MODE 658
SPI0_RxD 656
SPI0_SR 657
SPI0_TxD 656

stfd 802
stfdu 803
stfdux 804
stfdx 805
stfiwx 806
stfs 807
stfsu 808
stfsx 810
supported configurability 60
system reset results 190

T

time base registers 61
timing diagrams, PCI bridge 411

U

UART controller
  clocking 630
  DMA operation 645
  features 629
  FIFO 643
  registers 632
  sleep mode 645
UART receiver DMA mode 649
UART transmitter DMA mode 646
UARTx_DLL 639
UARTx_DLM 639
UARTx_FCR 635
UARTx_IER 633
UARTx_IIR 634
UARTx_LCR 635
UARTx_LSR 637
UARTx_MCR 636
UARTx_MSR 638
UARTx_RBR 633
UARTx_SCR 639
UARTx_THR 633
UIC 58
UIC controller 223
  features 223
  interrupt assignments 224
  programming 226
  registers 226
UIC0_CR 235
UIC0_ER 230
UIC0_MSR 248
UIC0_PR 239
UIC0_SR 226
UIC0_TR 244
UIC0_VCR 253
UIC0_VR 254
UIC1_CR 237
UIC1_ER 233
UIC1_MSR 250
UIC1_PR 241
UIC1_SR 228
UIC1_TR 246
UIC1_VCR 253
UIC1_VR 255
universal interrupt controller 58
universal serial bus interface 59
USB 693
USB 1.1 Host Interface 695
USB 1.1 Host registers 697
USB 2.0 Device Interface 727
USB 2.0 Device registers 730
USB controller data flow 695
USB interface 59, 693
USB2D0_FADDR 733
USB2D0_FRAME 739
USB2D0_INCSR 742
USB2D0_INCSR0 741
USB2D0_INDEX 739
USB2D0_INMAXP 744
USB2D0_INTRIN 731
USB2D0_INTRINE 734
USB2D0_INTROUT 734
USB2D0_INTROUTE 737
USB2D0_INTRUSB 736
USB2D0_INTRUSBE 735
USB2D0_OUTCOUNT0 748
USB2D0_OUTCSR 745
USB2D0_OUTMAXP 747
USB2D0_POWER 731
USB2D0_TESTMODE 738

Z

ZMII bridge
  features 575
  interface signals 576
  interfaces 577
  registers 579
ZMII0_FER 579
ZMII0_SMIISR 581
ZMII0_SSR 580

Revision Log

Revision Date	Version	Description
04/20/2006	1.19	Correct UICx_VCR and UICx_VR register names and address correlation. Misc. updates. Remove references to PerWE that were missed in 06/10/2004 purge. Move USB information from supplements into this document.
11/17/2005	1.18	Clean up document formatting and structure. Add note in DMA section indicating that EBC does not support data parity checking. Add additional info to Clocking section.
10/27/2005		Change default boot configuration from Option A to default configuration X (Errata DC_6). Correct list of registers that must be modified to change boot configuration (Doc Issue 135).
10/13/2005	1.17	Move processor overview to PPC440 Processor User's Manual.
08/15/2005		Reduced size by removing all redundant and duplicate material and separating processor core into separate book.
07/18/2005		Misc. updates and corrections.
07/08/2005		Implement 440GR corrections that are applicable to the 440EP.
06/29/2005		Implement corrections in GPT, EBC, and SPI chapters.
06/27/2005		Remove Confidential from title page header and all footers.
04/27/2005		Misc. updates and corrections.
03/15/2005		Correction correct update errors in 03/09 version.
03/09/2005		Misc. updates to registers and addresses. Duplication reduction. Corrections for AE's Doc Issues.
02/25/2005		Adding six EMAC registers to support Ethernet packet reject. Updated index markers. Delete references to PerPar signals in DMA chapter. Misc. updates to registers and addresses.
01/25/2005		More Pass 2 updates and corrections for the new arbiter.
12/13/2004		Pass 2 updates plus replacement of PLB4 arbiter with Crossbar arbiter.
10/25/2004		Misc. corrections to formatting. Remove references to USB Host 2.0. Replace USB details with references to USB supplement documents.
10/06/2004		Restore missing DMA-to-PL3 registers.
08/31/2004		Convert to AMCC template.
07/19/2004		Corrected SDR0_MFR register definition for ZMII Mode.
06/18/2004		Clarified indirect addressing involving **mtdcr** and **mfdcr** instructions used in conjunction with CPR0_, SDR0_, SDRAM0_ AND EBC0_ configuration registers. Reworked address and access columns in register descriptions for DDR SDRAM Registers (SDRAM0_xxxxx), contained in Section 18, and External Bus Controller Registers (EBC0_xxxxx in Section 20. Confirmed appropriate DCR number/offset distinctions are made in CPR0_ and SDR0 register descriptions. Made equivalent changes to Section 33, Register Summary Introduction tables 4-6 and 4-7.

Revision Date	Version	Description
06/15/2004		Maintenance update. Corrected EBC chapter: textual errors in descriptions of EBC0_BESR0 and EBC0_BESR1 registers and changed register objects.
06/10/2004		Corrected EBC chapter to remove all references to the PerWE signal, which is not pinned out on the PPC440EP.
04/29/2004		Corrected multiple errors SDR0_CUST1.reg and SDSTPn.reg files, including references in ZMII.fm and EBC.fm
04/28/2004		Corrected multiple errors in documentation of DMA to PLB3 Controller registers DMA2P30
04/27/2004		Corrected address of register SPI0_CDM in three files. Changes impacted the following files: Section 27, Serial Peripheral Interface Section 33, Register Summary (Table 33-4. PPC440EP Memory Mapped Registers)
03/19/2004		Confidential Preliminary version (SA14-2737-04) of the user's manual.
02/15/04-3/15/04		Overview chapter has been reviewed and updated to reflect all functional units implemented in the chip. On-chip bus chapter has been updated to reflect PLB3 and PLB4 naming conventions including register descriptions. PLB performance chapter has been included in the book with corresponding registers Programming chapter has been reviewed/updated to reflect updated address map, MMIO and DCR tables Reset and initialization chapter has been reviewed and updated as per designer review/comments Bootstrap controller chapter updated to reflect designer review/comments Timer facilities chapter updated to reflect designer review/comments Clocking chapter updated to include clock update register Clock and power management chapter reviewed/updated to include PPM field in registers Universal interrupt controller chapter has been updated to include PLB performance monitor UIC assignments documented in every UIC register External bus controller chapter has been updated to reflect chip selects and banks DMA controller chapter has been updated to reflect PLB4 and PLB3 DMA information including register naming conventions UART chapter updated to reflect DMA channels USB chapter has ben updated to include USB2.0 device information including register description. All register summaries have been reviewed and updated to reflect register naming and address mapping information. Register field names maintained to be consistent with field definitions and naming conventions. General review/comments updated as needed Reviewed and updated index markers for the book
01/15/04		Confidential Preliminary version (SA14-2737-03) of the user's manual. This version is being distributed among alpha customers under NDA.
09/22/03		Confidential Preliminary version (SA14-2737-02) of the user's manual distributed among reviewers.
05/30/03		First Confidential draft (SA14-2737-01) of the user's manual based on the chip spec.