Marvell® PXA3xx (88AP3xx) Processor Family

Vol. I: System and Timer Configuration Developers Manual

PXA30x Processor = 88AP300, 88AP301, 88AP302, 88AP303

PXA31x Processor = 88AP310, 88AP312

PXA320 Processor = 88AP320, 88AP322

Doc. No. MV-S301374-01, Rev. 2.0 Version -

April 6, 2009 Released

Document Conventions

	Note: Provides related information or information of special importance.
!	**Caution:** Indicates potential damage to hardware or software, or loss of data.
	Warning: Indicates a risk of personal injury.

Document Status

Draft	For internal use. This document has not passed a complete technical review cycle and ECN signoff process.
Preliminary Tapeout (Advance)	This document contains design specifications for a product in its initial stage of design and development. A revision of this document or supplementary information may be published at a later date. Marvell may make changes to these specifications at any time without notice. Contact Marvell Field Application Engineers for more information.
Preliminary Information	This document contains preliminary specifications. A revision of this document or supplementary information may be published at a later date. Marvell may make changes to these specifications at any time without notice. . Contact Marvell Field Application Engineers for more information.
Complete Information	This document contains specifications for a product in its final qualification stages. Marvell may make changes to these specifications at any time without notice. Contact Marvell Field Application Engineers for more information.

Milestone Indicator:
Draft = 0.xx
Advance = 1.xx
Preliminary = 2.xx
Complete = 3.xx

X . Y Z

Work in Progress Indicator
Zero means document is released.

Various Revisions Indicator

Doc Status: Released	Technical Publication: 3.30

For more information, visit our website at: www.marvell.com

Disclaimer
No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document.

Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications.

With respect to the products described herein, the user or recipient, in the absence of appropriate U.S. government authorization, agrees:

1) Not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2;

2) Not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and,

3) In the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML").

At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.

Copyright © 2009. Marvell International Ltd. All rights reserved. Marvell, the Marvell logo, Moving Forward Faster, Alaska, Fastwriter, Datacom Systems on Silicon, Libertas, Link Street, NetGX, PHYAdvantage, Prestera, Raising The Technology Bar, The Technology Within, Virtual Cable Tester, and Yukon are registered trademarks of Marvell. Ants, AnyVoltage, Discovery, DSP Switcher, Feroceon, GalNet, GalTis, Horizon, Marvell Makes It All Possible, RADLAN, UniMAC, and VCT are trademarks of Marvell. Intel XScale® is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries. All other trademarks are the property of their respective owners.

Contents

1 Introduction 23

1.1 About This Manual 23
1.1.1 Number Representation 24
1.1.2 Naming Conventions 24
1.1.3 Data Types 24
1.1.4 Related Documents 25

1.2 Product Overview 26
1.2.1 Application Subsystem Microarchitecture and Core 29
1.2.2 Application Subsystem Microarchitecture Features 29

1.3 XScale® Microarchitecture Compatibility 30
1.3.1 Multimedia Coprocessor 30
1.3.2 Internal Memory Architecture 30
1.3.3 General-Purpose I/O (GPIO) 31
1.3.4 Power Management 31
1.3.5 One-Wire Controller 31
1.3.6 DMA Controller 31
1.3.7 Interrupt Controller 32
1.3.8 Real-Time Clock (RTC) 32
1.3.9 Operating System Timers 33
1.3.10 Performance Monitor 33
1.3.11 Test 33
1.3.12 Power I2C Controller 33
1.3.13 External Memory Interfaces 34
1.3.13.1 Dynamic Memory Controller 34
1.3.13.2 Static Memory Controller 34
1.3.13.3 NAND Flash Controller 34
1.3.13.4 Internal SRAM Memory 35
1.3.13.5 Multimedia Card, SD Memory Card, and SDIO Card 35
1.3.14 Graphics and Input Controllers 36
1.3.14.1 LCD Panel Controller 36
1.3.14.2 Mini-LCD Panel Controller 36
1.3.14.3 Camera Image Capture Interface 37
1.3.14.4 Graphics Controller 38
1.3.14.5 Keypad Controller 39
1.3.14.6 Hardware Video Accelerator (PXA31x Processor Only) 39
1.3.14.7 ADC and Touchscreen Interface (PXA32x Processor Only) 39
1.3.15 Serial Ports 40
1.3.15.1 USB 1.1 Device Controller (PXA32x and PXA30x Processors Only) 40
1.3.15.2 USB 1.1 Host Controller 40
1.3.15.3 USB 2.0 High Speed Device Controller 40
1.3.15.4 Synchronous Serial Ports (SSP) 41
1.3.15.5 AC'97 CODEC Interface 41
1.3.15.6 UARTS 41
1.3.15.7 Consumer Infrared Controller 42
1.3.15.8 Pulse-Width Modulation Unit (PWM) 43
1.3.15.9 Universal Subscriber ID Controller 43
1.3.15.10 I2C Serial Bus Port 43

Copyright © 2009 Marvell

2 System Architecture Overview 45
2.1 Microarchitecture Implementation Options 45
2.2 Endianness 45
2.3 Memory Switch Versus System Bus 46
2.4 I/O Ordering 46
2.5 Accessing Peripherals on Internal Peripheral Bus 46
2.5.1 Programmed I/O Operations Using the Bridge 46
2.5.2 Data Transfer Using DMA 47
2.6 Peripheral Access on Internal System Buses 47
2.7 DMA/Peripheral Split Transactions 47
2.8 System Bus Arbiters 48
2.9 System Access Latencies 48
2.10 Achieving Optimum Performance 49
2.11 Semaphores 50
2.12 Interrupts 50
2.13 Reset 50
2.14 Selecting Peripherals Versus General-Purpose I/O 51
2.15 Power-On Reset and Boot Operation 51
2.16 Memory Map and Registers 52
2.16.1 Application Subsystem Microarchitecture Coprocessor Register Summary 53
2.16.2 Interrupt Controller Registers 56
2.16.3 Performance Monitoring Registers 56
2.16.4 Clock Configuration and Power Management Registers 57
2.16.5 Coprocessor Software Debug Registers 58
2.16.6 Coprocessor 15 58
2.16.6.1 Processor ID Register 58
2.16.6.2 Processor Cache Type Register 60
2.16.6.3 Auxiliary Control Register (P-Bit) 60
2.16.6.4 Coprocessor Access Register (CPAR) 60
2.16.6.5 Additions to Coprocessor 15 Functionality 60

3 Memory Switch 63
3.1 Features 63
3.2 I/O Pins 63
3.3 Interface Modules 63
3.3.1 Priority Control 64
3.3.2 The Memory Switch Concept 65

4 Pin Descriptions and Control 67
4.1 Overview 67
4.2 Features 67
4.3 PXA32x processor Pin List with Alternate Functions 68
4.4 PXA31x Processor Pin List with Alternate Functions 76
4.5 PXA30x Pin List with Alternate Functions 82
4.6 Signal Descriptions 90

Copyright © 2009 Marvell

4.7 Pin Control Unit Overview 107
4.7.1 Checking for Completion of a Multi-Function Pin Operation 107
4.7.2 Access to Nonexistent Registers or Pins 108
4.7.3 Pin Control Unit Address Map 108
4.8 Multi-Function Pin Description 124
4.9 Edge-Detect Operation 125
4.10 Low-Power Mode Operation 126
4.11 Register Description 127
4.11.1 Register Summary 128
4.11.2 Multi-Function Pin Registers (MFPR) 129

5 General-Purpose I/O Unit 133

5.1 PXA3xx Processor Differences 134
5.2 Features 134
5.3 Signals 134
5.4 Register Descriptions 135
5.4.1 Register Descriptions 136
5.4.2 GPIO Pin-Level Registers (GPLRx) 139
5.4.2.1 GPIO Pin-Level Register 0 (GPLR0) 139
5.4.2.2 GPIO Pin-Level Register 1(GPLR1) 139
5.4.2.3 GPIO Pin-Level Register 2(GPLR2) 139
5.4.2.4 GPIO Pin-Level Register 3 (GPLR3) 139
5.4.3 GPIO Pin Direction Registers (GPDRx) 140
5.4.3.1 GPIO Pin Direction Register 0 (GPDR0) 140
5.4.3.2 GPIO Pin Direction Register 1 (GPDR1) 140
5.4.3.3 GPIO Pin Direction Register 2 (GPDR2) 140
5.4.3.4 GPIO Pin Direction Register 3 (GPDR3) 140
5.4.4 GPIO Pin Bit-Wise Set Direction Registers (GSDRx) 141
5.4.4.1 GPIO Pin Bit-Wise Set Direction Register 0 (GSDR0) 141
5.4.4.2 GPIO Pin Bit-Wise Set Direction Register 1 (GSDR1) 141
5.4.4.3 GPIO Pin Bit-Wise Set Direction Register 2 (GSDR2) 141
5.4.4.4 GPIO Pin Bit-Wise Set Direction Register 3 (GSDR3) 141
5.4.5 GPIO Pin Bit-Wise Clear Direction Registers (GCDRx) 142
5.4.5.1 GPIO Pin Bit-Wise Clear Direction Register 0 (GCDR0) 142
5.4.5.2 GPIO Pin Bit-Wise Clear Direction Register 1 (GCDR1) 142
5.4.5.3 GPIO Pin Bit-Wise Clear Direction Register 2 (GCDR2) 142
5.4.5.4 GPIO Pin Bit-Wise Clear Direction Register 3 (GCDR3) 142
5.4.6 GPIO Pin Output Set Registers (GPSRx) and Pin Output Clear Registers (GPCRx) 143
5.4.6.1 GPIO Pin Output Set Register 0 (GPSR0) 143
5.4.6.2 GPIO Pin Output Set Register 1 (GPSR1) 143
5.4.6.3 GPIO Pin Output Set Register 2 (GPSR2) 143
5.4.6.4 GPIO Pin Output Set Register 3 (GPSR3) 143
5.4.6.5 GPIO Pin Output Clear Register 0 (GPCR0) 143
5.4.6.6 GPIO Pin Output Clear Register 1 (GPCR1) 143
5.4.6.7 GPIO Pin Output Clear Register 2 (GPCR2) 144
5.4.6.8 GPIO Pin Output Clear Registers 3 (GPCR3) 144
5.4.7 GPIO Rising-Edge Detect-Enable Registers (GRERx) 145
5.4.7.1 GPIO Rising-Edge Detect-Enable Register 0 (GRER0) 145
5.4.7.2 GPIO Rising-Edge Detect-Enable Register 1 (GRER1) 145
5.4.7.3 GPIO Rising-Edge Detect-Enable Register 2 (GRER2) 145
5.4.7.4 GPIO Rising-Edge Detect-Enable Register 3 (GRER3) 145
5.4.8 GPIO Bit-Wise Set Rising-Edge (GSRERx) and GPIO Bit-wise Clear Rising-Edge (GCRERx)

Detect-Enable Registers146
5.4.8.1 GPIO Bit-Wise Set Rising-Edge Register 0 (GSRER0)....146
5.4.8.2 GPIO Bit-Wise Set Rising-Edge Register 1 (GSRER1)....146
5.4.8.3 GPIO Bit-Wise Set Rising-Edge Register 2 (GSRER2)....146
5.4.8.4 GPIO Bit-Wise Set Rising-Edge Register 3 (GSRER3)....146
5.4.8.5 GPIO Bit-wise Clear Rising-Edge (GCRER0) Detect-Enable Register 0....146
5.4.8.6 GPIO Bit-wise Clear Rising-Edge (GCRER1) Detect-Enable Register 1....146
5.4.8.7 GPIO Bit-wise Clear Rising-Edge (GCRER2) Detect-Enable Register 2....146
5.4.8.8 GPIO Bit-wise Clear Rising-Edge (GCRER3) Detect-Enable Register 3....146
5.4.9 GPIO Falling-Edge Detect-Enable Registers (GFERx)....147
5.4.9.1 GPIO Falling-Edge Detect-Enable Register 0 (GFER0)....148
5.4.9.2 GPIO Falling-Edge Detect-Enable Register 1 (GFER1)....148
5.4.9.3 GPIO Falling-Edge Detect-Enable Register 2 (GFER2)....148
5.4.9.4 GPIO Falling-Edge Detect-Enable Register 3 (GFER3)....148
5.4.10 GPIO Bit-Wise Set Falling-Edge (GSFERx) and GPIO Bit-wise Clear Falling-Edge (GCFERx) Detect-Enable Registers148
5.4.10.1 GPIO Bit-Wise Set Falling-Edge Register 0 (GSFER0)....149
5.4.10.2 GPIO Bit-Wise Set Falling-Edge Register 1 (GSFER1)....149
5.4.10.3 GPIO Bit-Wise Set Falling-Edge Register 2 (GSFER2)....149
5.4.10.4 GPIO Bit-Wise Set Falling-Edge Register 3 (GSFER3)....149
5.4.10.5 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 0 (GCFER0)....149
5.4.10.6 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 1 (GCFER1)....149
5.4.10.7 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 2 (GCFER2)....149
5.4.10.8 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 3 (GCFER3)....149
5.4.11 GPIO Edge Detect Status Registers (GEDRx)....150
5.4.11.1 GPIO Edge Detect Status Register 0 (GEDR0)....150
5.4.11.2 GPIO Edge Detect Status Register 1 (GEDR1)....150
5.4.11.3 GPIO Edge Detect Status Register 2 (GEDR02)....150
5.4.11.4 GPIO Edge Detect Status Register 3 (GEDR3)....150

6 Clock Controllers and Power Management....153

6.1 PXA3xx Processor Differences....153
6.2 Features....153
6.3 External Power Management and Clock Signals....155
6.4 Power States....161

Copyright © 2009 Marvell

6.5 Power Modes 164
6.5.1 S0/D0/C0 Run Mode 165
6.5.2 S0/D0/C1, Core Idle Mode 166
6.5.3 S0/D0CS/C0 Ring Oscillator Mode 167
6.5.3.1 Entering S0/D0CS/C0 167
6.5.3.2 Exiting S0/D0CS/C0 168
6.5.3.3 CP14 Commands in Ring Oscillator Mode 169
6.5.3.4 Non-CP14 Commands in Ring Oscillator Mode 169
6.5.4 S0/D1/C2, Standby with LCD Refresh Mode 169
6.5.4.1 Entering S0/D1/C2 Using Software 170
6.5.4.2 Entering S0/D1/C2 from a Wakeup Event from S0/D2/C2 170
6.5.4.3 Exiting S0/D1/C2 170
6.5.5 S0/D2/C2, Standby Mode 171
6.5.5.1 Entering S0/D1/C2 Using Software 171
6.5.5.2 Entering S0/D2/C2 due to a Wakeup Event 172
6.5.5.3 Exiting S0/D2/C2 172
6.5.6 S2/D3/C4, Sleep Mode 173
6.5.6.1 Entering S2/D3/C4 173
6.5.6.2 Exiting S2/D3/C4 174
6.5.7 S3/D4/C4, Deep Sleep Mode 174
6.5.7.1 Entering S3/D4/C4 174
6.5.7.2 Exiting S3/D4/C4 175
6.5.8 S4/D4/C4, Powered Off Mode 175
6.5.9 Coupling Voltage Change with Power Modes 175
6.5.10 Power Mode Status 176
6.5.11 Power Mode Latencies 177

6.6 Processor Clocks 180
6.6.1 Processor System Clocks 182
6.6.2 Power Mode Clocking 183
6.6.3 Clocks During Power Mode Changes 185
6.6.4 Functional Clock Gating 185

6.7 Operating Points 186
6.7.1 Core Frequency Changes 191
6.7.1.1 Run Mode Frequency Change 192
6.7.1.2 Turbo Mode Frequency Change 192
6.7.2 Peripheral Frequency Changes 192
6.7.2.1 Coupling Voltage Changes with Operating Point Changes 193

6.8 Voltage Management 195
6.8.1 Programming Restrictions for the PWR_I2C 196
6.8.2 External Voltage Regulator Requirements 196
6.8.3 Hardware-Controlled Voltage-Change Sequencer 196
6.8.4 Start-of-Day (SOD) Operation 197
6.8.5 Accessing PWR_I2C Registers Directly through Software 197

6.9 Internal Memory SRAM Power Management 198
6.9.1 Power Mode States 199
6.9.2 Auto Power Down 199

6.10 Power Management Operation 200
6.10.1 Configuring Hardware Driven PWR_I2C commands 200
6.10.2 Preparing for Power Mode Transitions 200
6.10.3 Boot ROM Requirements 201

6.11 Wakeup Events and Sources 201
6.11.1 Services Wakeups 206
6.11.2 Peripheral Controller Wakeups 206
6.11.3 Pin Control Wakeups 206
6.11.3.1 Edge Detect Operation 207
6.11.3.2 Generic Wakeups 207
6.11.3.3 First-Level Wakeups 207
6.11.3.4 Second-Level Generic Wakeups 208

6.12 Resets 208
6.12.1 Power-on Reset 209
6.12.2 Hardware Reset 210
6.12.3 GPIO Reset 211
6.12.4 Global Watchdog Reset 212
6.12.5 S3/D4/C4 Low-Power Mode Exit Reset 212

6.13 Registers 212
6.13.1 Core Clock Configuration Register (XCLKCFG) (CP14 Register 6) 214
6.13.2 Core PWRMODE Register (CP14 Register 7) 215
6.13.3 Power Management Unit Control Register (PMCR) 216
6.13.4 Power Management Unit Status Register (PSR) 217
6.13.5 Power Management Unit Scratch-Pad Register (PSPR) 218
6.13.6 Power Management Unit General Configuration Register (PCFR) 219
6.13.7 Power Manager Wake-Up Enable Register (PWER) 221
6.13.8 Power Manager Wake-Up Status Register (PWSR) 222
6.13.9 Power Manager EXT_WAKEUP<1:0> Control Register (PECR) 223
6.13.10 Power Manager Mask Event Register (PMER) 226
6.13.11 VCC_APPS Voltage Control Register (AVCR) 228
6.13.12 VCC_SRAM Voltage Control Register (SVCR) 229
6.13.13 Power Management Unit Voltage Change Control Register (PVCR) 231
6.13.14 Oscillator Configuration Register (OSCC) 233
6.13.15 Application Subsystem Power Status/Configuration Register (ASCR) 234
6.13.16 Application Subsystem Reset Status Register (ARSR) 235
6.13.17 Application Subsystem Wakeup from D3 Enable Register (AD3ER) 236
6.13.18 Application Subsystem Wakeup from D3 Status Register (AD3SR) 240
6.13.19 Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER) 244
6.13.20 Application Subsystem Wakeup from D2 to D0 Status Register (AD2D0SR) 247
6.13.21 Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER) 250
6.13.22 Application Subsystem Wakeup from D2 to D1 Status Register (AD2D1SR) 250
6.13.23 Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER) 251
6.13.24 Application Subsystem Wakeup from D1 to D0 Status Register (AD1D0SR) 254
6.13.25 Application Subsystem General Purpose Register (AGENP) 256
6.13.26 Application Subsystem D3 Configuration Register (AD3R) 257
6.13.27 Application Subsystem D2 Configuration Register (AD2R) 259
6.13.28 Application Subsystem D1 Configuration Register (AD1R) 260
6.13.29 Application Subsystem Clock Configuration Register (ACCR) 261
6.13.30 Application Subsystem Clock Status Register (ACSR) 267
6.13.31 Application Subsystem Interrupt Control/Status Register (AICSR) 270
6.13.32 D0 Mode Clock Enable Register A (D0CKEN_A) 271
6.13.33 D0 Mode Clock Enable Register B (D0CKEN_B) 276
6.13.34 AC '97 Clock Divisor Value Register (AC97_DIV) 278

7 1-Wire Bus Master Interface 281

7.1 Signals 282

Copyright © 2009 Marvell

7.2 Operation 282
7.2.1 Writing a Byte 282
7.2.2 Reading a Byte 282
7.2.3 I/O Signaling 283
7.2.3.1 Initialization Sequence 283
7.2.3.2 Write Time Slots 284
7.2.3.3 Read Time Slots 284
7.3 Interrupt Handling 285
7.4 Register Descriptions 286
7.4.1 1-Wire Command Register (W1CMDR) 286
7.4.2 1-Wire Transmit/Receive Buffer (W1TRR) 288
7.4.3 1-Wire Interrupt Register (W1INTR) 288
7.4.4 1-Wire Interrupt Enable Register (W1IER) 289
7.4.5 1-Wire Clock Divisor Register (W1CDR) 290

8 DMA Controller 293

8.1 PXA3xx Processor Differences 293
8.2 Features 293
8.3 Signal Descriptions 294
8.4 Operation 295
8.4.1 DMA Channels 295
8.4.1.1 DMA Channel-Priority Scheme 296
8.4.1.2 Channel States 296
8.4.2 DMA Descriptors 299
8.4.2.1 Descriptor-Fetch Transfer Operation 299
8.4.2.2 No-Descriptor-Fetch Transfer Operation 301
8.4.3 Transferring Data 302
8.4.3.1 Servicing Internal Peripherals 302
8.4.3.2 Servicing External Companion Chips (PXA32x Only) 303
8.4.3.3 Memory-to-Memory Moves 305
8.4.4 Programming Tips 305
8.4.4.1 Software Management Requirements 305
8.4.4.2 Programmed I/O Operations 306
8.4.4.3 Instruction Ordering 306
8.4.4.4 Misaligned Memory Accesses 306
8.4.5 How DMA Handles Trailing Bytes 307
8.4.6 Quick Reference to DMA Programming 309
8.4.7 Examples 315
8.5 Register Descriptions 319
8.5.1 Register Summary 319
8.5.2 DMA Request to Channel Map Register (DRCMRx) 328
8.5.3 DMA Descriptor Address Registers (DDADRx) 329
8.5.4 DMA Source Address Register (DSADRx) 330
8.5.5 DMA Target Address Registers (DTADRx) 331
8.5.6 DMA Command Registers (DCMDx) 332
8.5.7 DREQ Status Register 0 (DRQSR0) 336
8.5.8 DMA Channel Control/Status Registers (DCSRx) 337
8.5.9 DMA Interrupt Register (DINT) 344
8.5.10 DMA Alignment Register (DALGN) 344
8.5.11 DMA Programmed I/O Control Status Register (DPCSR) 345

9 Interrupt Controller 347
9.1 Features 347
9.2 Signal Descriptions 347
9.3 Operation 347
9.3.1 Accessing Interrupt Controller Registers 349
9.3.2 Enabling Coprocessor Access 350
9.3.3 Accessing the Coprocessor 350
9.3.4 Bit Positions and Peripheral IDs 351
9.4 Register Descriptions 353
9.4.1 Interrupt Controller Pending Registers (ICPR and ICPR2) 355
9.4.2 Interrupt Controller IRQ Pending Registers (ICIP and ICIP2) 363
9.4.3 Interrupt Controller FIQ Pending Registers (ICFP and ICFP2) 373
9.4.4 Interrupt Controller Mask Registers (ICMR and ICMR2) 382
9.4.5 Interrupt Controller Level Registers (ICLR and ICLR2) 391
9.4.6 Interrupt Controller Control Register (ICCR) 399
9.4.7 Interrupt Priority Registers 0 to 55 399
9.4.8 Interrupt Control Highest Priority Register (ICHP) 400

10 Real-Time Clock (RTC) 403
10.1 Features 403
10.2 Signals 403
10.3 Operation 404
10.3.1 Timer Module 407
10.3.2 Wristwatch Module 407
10.3.2.1 Programming Wristwatch Registers 408
10.3.2.2 Allowable Values for Wristwatch Register Fields 408
10.3.2.3 Effects of Data Written to Wristwatch Register Fields 409
10.3.3 Stopwatch Module 411
10.3.3.1 Interval Interrupts 411
10.3.4 Periodic Interrupt Module 412
10.3.5 Trimmer Module 412
10.3.5.1 Trim Procedure 413
10.3.5.2 RTTR Value Calculations 413
10.4 Register Descriptions 415
10.4.1 RTC Timer Trim Register (RTTR) 416
10.4.2 RTC Status Register (RTSR) 417
10.4.3 RTC Alarm Register (RTAR) 419
10.4.4 Wristwatch Day Alarm Registers (RDARx) 419
10.4.5 Wristwatch Year Alarm Registers (RYARx) 420
10.4.6 Stopwatch Alarm Registers (SWARx) 421
10.4.7 Periodic Interrupt Alarm Register (PIAR) 422
10.4.8 RTC Counter Register (RCNR) 423
10.4.9 RTC Day Counter Register (RDCR) 423
10.4.10 RTC Year Counter Register (RYCR) 424
10.4.11 Stopwatch Counter Register (SWCR) 424
10.4.12 Periodic Interrupt Counter Register (RTCPICR) 425

11 Operating System Timers 427
11.1 Features 427
11.2 Signals 427

Copyright © 2009 Marvell

11.3 Operation 428
11.3.1 Block Diagram 428
11.3.2 Compares and Matches 429
11.3.3 Marvell® PXA25x Processor Compatibility 429
11.3.4 Timer Channels 429
11.3.5 Counter Resolutions 429
11.3.5.1 Clock Generation for Counter 0 Register 429
11.3.5.2 Clock Generation for Channels 4–7 429
11.3.5.3 Clock Generation for Channels 8–11 430
11.3.6 External Synchronization (EXT_SYNC<1:0>) 430
11.3.6.1 Output Generation for CHOUT<1:0> 430
11.3.7 Snapshot Mode 431
11.3.8 Operation in Low-Power Modes 431

11.4 Register Descriptions 431
11.4.1 OS Match Control Registers (OMCRx) 433
11.4.2 OS Timer Match Registers (OSMRx) 438
11.4.3 OS Timer Watchdog Match Enable Register (OWER) 439
11.4.4 OS Timer Interrupt Enable Register (OIER) 440
11.4.5 OS Timer Count Register 0 (OSCR0) 440
11.4.6 OS Timer Count Registers (OSCR[4:11]) 441
11.4.7 OS Timer Status Register (OSSR) 441
11.4.8 OS Timer Snapshot Register (OSNR) 442

12 Performance Monitoring and Debug 443

12.1 Features 443

12.2 Software 443

12.3 Signals 443

12.4 Operation 443
12.4.1 Processor-Level Performance Events 444
12.4.2 Core-Level Performance Events 448
12.4.3 Debug Functionality 450

12.5 Register Descriptions 450
12.5.1 Register Summary 450
12.5.1.1 Event Select Registers (PML_ESL_[7:0]) 451
12.5.2 Processor Debug Unit (MDU) Configuration Registers 452
12.5.2.1 MDU XScale® Breakpoint Register (MDU_CPU_BP) 452
12.5.2.2 MDU 2DG Stop Register (MDU_2DG_EVENT) 453
12.5.2.3 MDU CW Match Signal Register (MDU_CW_MATCH) 454
12.5.3 Multi-Function Pin Monitor Register (MFP_MONITOR) 455

13 System Bus Arbiters 459

13.1 Features 459

13.2 Signals 459

13.3 Operation 459
13.3.1 Programmable Weights 459
13.3.2 Bus Parking 460
13.3.3 Bus Locking 460
13.3.4 System Considerations: System Bus Access Latency 461

13.4 Register Descriptions 461
13.4.1 Register Summary 461
13.4.2 System Bus Arbiter Control Registers (ARB_CNTRL_1 and ARB_CNTRL_2) 462
13.4.2.1 System Bus Arbiter Control Register 1 (ARB_CNTRL_1) 462
13.4.2.2 System Bus Arbiter Control Register 2 (ARB_CNTRL_2) 462

14 JTAG 469

14.1 Features 469

14.2 Signals 469

14.3 Operation 470
14.3.1 TAP Controller Reset 470
14.3.2 Instruction Register 471
14.3.3 Test Data Registers 472
14.3.3.1 Bypass Register 472
14.3.3.2 Boundary Scan Register (BSR) 472
14.3.3.3 Device Identification (ID) Register 473
14.3.3.4 XScale® Data-Specific Registers 473
14.3.4 TAP Controller 473

14.4 Register Descriptions 476

15 Power Manager I2C Interface 477

15.1 I/O Signals 477

15.2 Feature List 477

15.3 Functional Description 477
15.3.1 Operational Blocks 478
15.3.2 PWR_I2C Bus Interface Modes 478
15.3.3 Start and Stop Bus States 479
15.3.3.1 START Condition 480
15.3.3.2 No START or STOP Condition 481
15.3.3.3 STOP Condition 481
15.3.4 Data Transfer Sequence 481
15.3.4.1 Data and Addressing Management 481
15.3.4.2 PWR_I2C Acknowledge 483
15.3.4.3 Master Operations 484
15.3.4.4 Master Programming Examples 487
15.3.4.5 Initialize the Unit 487
15.3.4.6 Write 1 Byte as a Master 487
15.3.4.7 Read 1 Byte as a Master 488
15.3.4.8 Write 2 Bytes and Repeated Start Read 1 Byte as a Master 488
15.3.4.9 Read 2 Bytes as a Master - Send STOP Using the Abort 488
15.3.5 Slave Operations 489
15.3.6 Slave Mode Programming Examples 492
15.3.6.1 Initialize Unit 492
15.3.6.2 Write n Bytes as a Slave 492
15.3.6.3 Read n Bytes as a Slave 493
15.3.6.4 General Call Address 493
15.3.7 Glitch Suppression Logic 494
15.3.8 Reset Conditions 494

Copyright © 2009 Marvell

15.4 Register Descriptions 495
15.4.1 Power Manager I2C Control Register (PICR) 495
15.4.2 Power Manager I2C Status Register (PISR) 498
15.4.3 Power Manager I2C Slave Address Register (PISAR) 500
15.4.4 Power Manager I2C Data Buffer Register (PIDBR) 500
15.4.5 Power Manager I2C Bus Monitor Register (PIBMR) 501

16 Memory Map 505

16.1 Overview 505
16.2 PXA3xx Processor Differences 505
16.3 Memory-Mapped Registers Summary 509
16.4 Boot ROM Space 511

Copyright © 2009 Marvell

Figures

Figure 1: PXA30x Processor Block Diagram 27
Figure 2: PXA31x Processor Block Diagram 28
Figure 3: PXA32x Processor Block Diagram 29
Figure 4: Physical Address Map Decode Regions....53
Figure 5: Memory Switch Block Diagram....64
Figure 6: Memory Switch Concept....65
Figure 7: Clocks and Power Management Units....155
Figure 8: Services and Application Subsystem Power States 162
Figure 9: Processor Power Modes....165
Figure 10: Clock Generation Diagram 181
Figure 11: Organization and Memory Mapping of SRAM Arrays....199
Figure 12: 1-Wire Bus Master Block Diagram....281
Figure 13: 1-Wire Initialization Sequence (Reset and Presence Pulses)....284
Figure 14: 1-Wire Write Slots....284
Figure 15: 1-Wire Read Time Slots....285
Figure 16: DMAC Block Diagram....295
Figure 17: DREQ Timing Requirements 295
Figure 18: Flow of Descriptor Fetch Transfer Operation....300
Figure 19: Flow Chart for Descriptor Branching....301
Figure 20: No-Descriptor-Fetch Transfer Channel State Diagram....302
Figure 21: Descriptor Chain for Software Implementation of Full and Empty Bits 319
Figure 22: Descriptor Behavior on End-of-Receive (EOR) 343
Figure 23: Interrupt Controller Block Diagram 349
Figure 24: RTC Block Diagram 405
Figure 25: Operational Flow of the RTC Modules....406
Figure 26: RTC Wristwatch Module 407
Figure 27: Operating System Timers Block Diagram....428
Figure 28: Example: Reset of OSCR6 Based on Rising Edge of EXT_SYNC1 430
Figure 29: TAP Controller State Diagram 474
Figure 30: Start and Stop Conditions....480
Figure 31: Data Format of First Byte in Master Transaction 482
Figure 32: Acknowledge on the PWR_I2C Bus 484
Figure 33: Master-Receiver Read from Slave-Transmitter 486
Figure 34: Master-Receiver Read from Slave-Transmitter/Repeated Start*....487
Figure 35: A Complete Data Transfer 487
Figure 36: Master-Transmitter Write to Slave-Receiver....491
Figure 37: Master-Receiver Read to Slave-Transmitter....491
Figure 38: Master-Receiver Read to Slave-Transmitter, Repeated START** 492
Figure 39: General Call Address....494

Copyright © 2009 Marvell

Copyright © 2009 Marvell

Tables

Revision History for Volume I....22
Table 1: Supplemental Documentation....25
Table 2: Little-Endian Value Encoding....45
Table 3: Naturally Aligned Transfers....49
Table 4: Effect of Each Type of Reset on Internal Register State....51
Table 5: Coprocessor Register Summary....53
Table 6: Performance Monitoring Registers, Access: Coprocessor 14....56
Table 7: Devices Operating in S0/D0CS/C0 Mode....57
Table 8: Processor ID Register....58
Table 9: Coprocessor: New CPU ID and JTAG ID Values....59
Table 10: Processor CPAR Register....60
Table 11: PXA32x processor Alternate Function Table....68
Table 12: PXA31x Processor Alternate Function Table....76
Table 13: PXA30x Processor Alternate Function Table....83
Table 14: PXA3xx Processor Signal Descriptions....90
Table 15: PXA32x Processor Pad Control Addresses....108
Table 16: PXA31x Processor Pad Control Addresses....114
Table 17: PXA30x Processor Pad Control Addresses....119
Table 18: Low-Power Mode States....126
Table 19: SLEEP_SEL and RDH Multi-function Pin State Summary....127
Table 20: Pins and Control Register Summary....128
Table 21: MFPR Bit Definitions....129
Table 22: Pullup/Pulldown State Table....131
Table 23: PXA3xx Processors Feature Differences....134
Table 24: GPIO Controller Interface Signals Summary....134
Table 25: GPIO Register Summary....136
Table 26: GPLRx Bit Definitions....140
Table 27: GPDRx Bit Definitions....141
Table 28: GSDRx Bit Definitions....142
Table 29: GCDRx Bit Definitions....143
Table 30: GPSRx Bit Definitions....144
Table 31: GPCRx Bit Definitions....144
Table 32: GRERx Bit Definitions....145
Table 33: GSRERx Bit Definitions....147
Table 34: GCRERx Bit Definitions....147
Table 35: GFERx Bit Definitions....148
Table 36: GSFERx Bit Definitions....149
Table 37: GCFERx Bit Definitions....150
Table 38: GEDRx Bit Definitions....151
Table 39: PXA3xx Processors Feature Differences....153

Table 40: PXA3xx Processor Family Signals 156
Table 41: Services Power States (S-States) 162
Table 42: Application Subsystem Power States (D-States) 163
Table 43: Application Subsystem Core Power States (C-States) 164
Table 45: Low Power Mode Status Registers 176
Table 46: Low-Power Mode Entry Latencies 177
Table 47: Low Power Mode Exit Latencies 178
Table 48: PXA31x Low Power Mode Exit Latencies 179
Table 49: System Clocks 182
Table 50: Power Mode Clock Frequencies 183
Table 51: PXA31x Application Subsytem Core PLL, Turbo and Run Mode Output Frequencies 186
Table 52: PXA30x Application Subsytem Core PLL, Turbo and Run Mode Output Frequencies 186
Table 53: PXA32x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 130 MHz) 186
Table 54: PXA32x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 156 MHz) 187
Table 55: PXA31x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 130 MHz) 189
Table 56: PXA31x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 156 MHz) 190
Table 57: PXA30x Core PLL, Turbo and Run Mode Output Frequencies 191
Table 59: Hardware PWR_I2C command encoding 197
Table 60: Processor Wakeup Events 202
Table 61: PXA32x Processor Generic Wakeup[13] Sources 208
Table 62: Reset Status Registers 209
Table 63: Effect of Each Type of Reset on Internal Register State 209
Table 64: Core Coprocessor Register Summary 213
Table 65: Services Unit Register Summary 213
Table 66: Application Subsystem Unit Register Summary 213
Table 67: XCLKCFG Bit Definitions 215
Table 68: PWRMODE Bit Definitions 216
Table 69: PMCR Bit Definitions 217
Table 70: PSR Bit Definitions 217
Table 71: PSPR Bit Definitions 218
Table 72: PCFR Bit Definitions 219
Table 73: PWER Bit Definitions 222
Table 74: PWSR Bit Definitions 223
Table 75: PECR Bit Definitions 224
Table 76: PMER Bit Definitions 227
Table 77: AVCR Bit Definitions 229
Table 78: SVCR Bit Definitions 230
Table 79: PVCR Bit Definitions 232
Table 80: OSCC Bit Definitions 233
Table 81: ASCR Bit Definitions 235
Table 82: ARSR Bit Definitions 236
Table 83: AD3ER Bit Definitions 237
Table 84: AD3SR Bit Definitions 241
Table 85: AD2D0ER Bit Definitions 245

Copyright © 2009 Marvell

Table 86: AD2D0SR Bit Definitions 248
Table 87: AD2D1ER Bit Definitions 250
Table 88: AD2D1SR Bit Definitions 251
Table 89: AD1D0ER Bit Definitions 252
Table 90: AD1D0SR Bit Definitions 254
Table 91: AGENP Bit Definitions 257
Table 92: AD3R Bit Definitions 258
Table 93: AD2R Bit Definitions 259
Table 94: AD1R Bit Definitions 260
Table 95: ACCR Bit Definitions....262
Table 96: ACSR Bit Definitions....268
Table 97: AICSR Bit Definitions....270
Table 98: D0CKEN_A Bit Definitions: Clock Enable Mappings for Units....272
Table 99: D0CKEN_B Bit Definitions: Clock Enable Mappings for Units....276
Table 100: 278
Table 101: AC97_DIV Bit Definitions....279
Table 97: 1-Wire Signal 282
Table 98: 1-Wire Bus Register Summary 286
Table 99: W1CMDR Bit Definitions....287
Table 100: W1TRR Bit Definitions 288
Table 101: W1INTR Bit Definitions 288
Table 102: W1IER Bit Definitions 290
Table 103: W1CDR Bit Definitions....291
Table 104: PXA3xx Processors Feature Differences 293
Table 105: DMAC External Signal List 294
Table 106: Channel Priority 296
Table 107: Channel States Based on Software Configuration 297
Table 108: Configuration for Peripheral Bus Peripheral Related Data Transfers 309
Table 109: Configuration for Memory-to-Memory Data Transfers 310
Table 110: System Bus Peripheral Related Data Transfer Configuration....310
Table 111: Configuration for Companion Processor (CC)-Related Data Transfers....311
Table 112: DMA Quick Reference for Internal Peripherals....311
Table 113: DMA Controller Registers 320
Table 114: DRCMR[0:63], DRCMR[64:99] Bit Definitions 328
Table 115: DDADR[0:31] Bit Definitions 329
Table 116: DSADR[0:31] Bit Definitions 330
Table 117: DTADR0–31 Bit Definitions 331
Table 118: DCMD[0:31] Bit Definitions 332
Table 119: DRQSR0 Bit Definitions....336
Table 120: DCSR[0:31] Bit Definitions....337
Table 121: DINT Bit Definitions 344
Table 122: DALGN Bit Definitions 345
Table 123: DPCSR Bit Definitions 346
Table 124: Interrupt Controller Register Mapping....350

Table 125: Summary of Bit Positions for Primary Sources 351
Table 127: Interrupt Controller Register Summary by Coprocessor Addresses 354
Table 128: ICPR Bit Definitions 355
Table 129: ICPR2 Bit Definitions 360
Table 130: ICIP Bit Definitions 364
Table 131: ICIP2 Bit Definitions 369
Table 132: ICFP Bit Definitions 373
Table 133: ICFP2 Bit Definitions 378
Table 134: ICMR Bit Definitions 383
Table 135: ICMR2 Bit Definitions 387
Table 136: ICLR Bit Definitions 391
Table 137: ICLR2 Bit Definitions 395
Table 138: ICCR Bit Definitions 399
Table 139: IPR0/55 Bit Definitions 400
Table 140: ICHP Bit Definitions 401
Table 141: Real-Time Clock Controller I/O Signal 404
Table 142: RTC Controller Alarm Bit Location Summary 407
Table 143: Valid and Invalid Data For The Wristwatch Register Fields 408
Table 144: Valid Data for Day of Month (DOM) Field In RYCR 409
Table 145: RTC Controller Register Summary 415
Table 146: RTTR Bit Definitions 416
Table 147: RTSR Bit Definitions 417
Table 148: RTAR Bit Definitions 419
Table 149: RDARx Bit Definitions 420
Table 150: RYARx Bit Definitions 421
Table 151: SWARx Bit Definitions 421
Table 152: PIAR Bit Definitions 422
Table 153: RCNR Bit Definitions 423
Table 154: RDCR Bit Definitions 423
Table 155: RYCR Bit Definitions 424
Table 156: SWCR Bit Definitions 425
Table 157: RTCPICR Bit Definitions 425
Table 158: Operating System Timers Interface Signals Summary 427
Table 159: OS Timers Register Summary 432
Table 160: OMCR4/5/6/7 Bit Definitions 433
Table 161: OMCR8/10 Bit Definitions 435
Table 162: OMCR9/11 Bit Definitions 437
Table 163: OSMR0–11 Bit Definitions 439
Table 164: OWER Bit Definitions 440
Table 165: OIER Bit Definitions 440
Table 166: OSCR0 Bit Definitions 441
Table 167: OSCR[4:11] Bit Definitions 441
Table 168: OSSR Bit Definitions 442
Table 169: OSNR Bit Definitions 442

Copyright © 2009 Marvell

Table 170: ASSP Processor Performance Monitor Events 444
Table 171: Core Processor Performance Monitor Events 448
Table 172: Performance Monitoring and Multicore Debug Register Summary....451
Table 173: PML_ESEL_[7:0] Bit Definitions 451
Table 174: MDU_XSCALE_BP Bit Definitions....452
Table 175: MDU_2DG_EVENT Bit Definitions 453
Table 176: MDU_CW_MATCH Bit Definitions....454
Table 177: MFP_MONITOR Bit Definitions 456
Table 178: Internal System Bus Arbiter Register Summary 461
Table 179: ARB_CNTRL_1 Bit Definitions 462
Table 180: ARB_CNTRL_2 Bit Definitions 464
Table 181: TAP Controller Pin Definitions 469
Table 182: JTAG Instruction Description 471
Table 183: JTAG Device Identification Register 473
Table 184: TAP Controller States 474
Table 185: I2C Interface Signals Summary 477
Table 186: PWR_I2C Bus Definitions....477
Table 187: PWR_I2C Bus Interface Operating Modes 478
Table 188: START and STOP Bit Definitions 480
Table 189: Master Transactions 484
Table 190: Slave Transactions 490
Table 191: General Call Address Second Byte Definitions 494
Table 192: Power Manager PWR_I2C Register Summary....495
Table 193: PICR Bit Definitions 496
Table 194: PISR Bit Definitions 498
Table 195: PISAR Bit Definitions....500
Table 196: PIDBR Bit Definitions....501
Table 197: PIBMR Bit Definitions 502
Table 198: PXA3xx Processors Feature Differences 505
Table 199: Memory Map (Part 1) — From 0x0000_0000 to 0x7FFF_FFFF....506
Table 200: Memory Map (Part Two) — From 0x8000_0000 to 0xFFFF FFFF....508
Table 201: Register Address Summary 509
Table 202: Boot ROM Locations....512
Table 203: IAS Revision Changes....512
Table 204: DM Revision Changes....512

Revision History for Volume I

Rev	Date	Description
A	2-08-08	Initial upload of combined PXA3xx Processor Family Developer Manual, Volume I
B	2-15-08	Chapter 4: Added User Settings row to Table 21 Chapter 6: Added steps 7 and 9 to Section 6.5.3.1 Entering S0/D0CS/C0 Modified step 7 in Section 6.5.3.2 Exiting S0/D0CS/C0 to read that bits in ACSR are cleared (not set) Added Reserved bit status for Bit 20 (PXA31x only) in Table 94 for D0CKEN_A Bit Definitions: Clock Enable Mappings for Units Added SKU definitions to front cover; “88AP3xx” to header in master pages
C	3-21-08	Chapter 4, Chapter 6, Chapter 14: Revised existing references to EXT_WAKEUP<0> to now read EXT_WAKEUP<1:0>
2.0	10-9-08	Updated for TSMC requirements

Copyright © 2009 Marvell

1 Introduction

The PXA3xx (88AP3xx) Processor Family (also referred to throughout this volume as the *processor*) consists of high-performance, low-power microprocessors providing a rich feature set optimized for wireless handheld applications. The PXA3xx Processor Family complies with the ARM* Instruction Set Architecture V5TE. It is one of the XScale® technology building blocks featuring:

- A super-pipelined RISC microarchitecture, providing performance for demanding applications.
- Wireless Intel SpeedStep® Power Manager technology, enabling dynamic scaling of computing performance and power consumption based on application requirements.

This chapter presents an overview of the processor. It also describes documentation conventions and related documents referenced throughout the four-volume set.

1.1 About This Manual

The PXA3xx Processor Developers Manual consists of four volumes.

Note

The individual product numbers for the PXA3xx processor SKUs are as follows:
PXA30x = 88AP300, 88AP301, 88AP302, 88AP303
PXA31x = 88AP310, 88AP311, 88AP312
PXA32X = 88AP320, 88AP322

These SKU numbers are not referenced within this revision of the Developers Manual (all four volumes).

Note

ABOUT THIS MANUAL: Any functional or operational differences among the PXA3xx processors are highlighted in a table below the *Overview* section of individual chapters in all four volumes. **Be sure to review carefully the register tables for those functions or peripherals that do have differences.** Some peripherals do not have any differences across the PXA3xx processor family and therefore those chapters will not have a Differences table.

- This volume, *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual*, presents an overview of the processors. It describes product features and device-specific configuration information, such as the memory map and signal multiplexing. It also gives an overview of clocking and power management features. This volume describes system-wide functions, such as the memory switch, clock control, power management, 1-Wire bus, DMA controller, general-purpose I/O (GPIO) controller, real-time clock, operating system timers, system-bus arbitration, and interrupt control.
- *PXA3xx Processor Family Vol II: Memory Configuration Developers Manual* describes the memory interfaces and configuration of the memory controllers. The dynamic memory controller, static memory controller, external memory pin interface (EMPI), data flash controller,

Copyright © 2009 Marvell

data flash interface (DFI), internal memory, and MultiMediaCard/SD/SDIO controller are all discussed in this volume.

- *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual* describes the configuration of the LCD controller and mini-LCD controller, Intel® Quick Capture Camera Interface, and keypad controller.
- *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual* describes the configuration of the serial controllers, including USB 1.1 client and host controllers, USB 2.0, synchronous serial protocol (SSP) ports, AC '97 controller, UARTs, consumer infrared port, pulse width modulator controllers, universal subscriber ID interfaces, and the I^2C bus interface unit.

The *PXA3xx Processor Family Developers Manual* is written for experienced programmers of ARM* Architecture V5TE-compliant processors. This manual assumes a working knowledge of the vocabulary and principles of embedded-systems programming. The core and the Intel® Wireless MMX™ 2 media enhancement technology are not described. Refer to Table 1 for more information.

1.1.1 Number Representation

All numbers in this document are decimal (base 10**)** unless designated otherwise. Hexadecimal numbers have a prefix of 0x, and binary numbers have a prefix of 0b. For example, 107 is represented as 0x6B in hexadecimal and 0b110_1011 in binary.

1.1.2 Naming Conventions

All signal and register-bit names appear in uppercase. Active low signals are prefixed with a lowercase "n".

Signal pin ranges are enclosed in angle brackets: EXTERNAL_ADDRESS<31:0>. Single numbered pins do not use the brackets: EXTERNAL_ADDRESS22.

Bit fields, which consist of two or more bits within a register, are enclosed in square brackets: REGISTER[BIT_FIELD].

In register definition tables, values shown in the "Reset" row have the following meanings:

- **0** = bit clear
- **1** = bit set
- **?** = bit is undefined

Abbreviations in the "Access" column have the following meanings:

- **R** = read only
- **W** = write only
- **R/W** = read and write
- There are two special cases:
- **R/WC** = R/W; to clear the bit, write 0b1 to it.
- **RC** = Read only; the bit is automatically cleared after it is read.

1.1.3 Data Types

In the context of the ARM* Architecture V5TE, a word consists of 32 bits. As a result, the following naming convention applies to the different data types in the PXA30x processor or PXA31x processor:

- 8 bits = byte (abbreviation **B**)
- 16 bits = half word (abbreviation **H**)
- 32 bits = word (abbreviation **W**)
- 64 bits = double word (abbreviation **D**)

Copyright © 2009 Marvell

1.1.4 Related Documents

Table 1 lists supplemental documentation for users of the PXA30x processor or PXA31x processor. Contact an a Marvell representative for the latest revision of Marvell documents without order numbers.

Table 1: Supplemental Documentation

Title
PXA310 Processor and PXA300 Processor Design Guides
PXA300 Processor and PXA310 Processor Electrical, Mechanical, and Thermal Specification (EMTS)
ARM Architecture Version 5T Specification (Document number ARM DDI 0100D-10) and *ARM Architecture Reference Manual* (ARM DDI 0100 or ISBN 0-201-73719-1)
ARM Architecture Reference Manual
ARM Multi-ICE System Design Considerations, Application Note 72 (ARM DAI 0072A)
Intel® Wireless MMX™ Technology Developers Guide
Intel XScale® Core Developers Manual
Intel® Mobile Scalable Link Specification
Mobile DDR SDRAM Specification
I^2C-*Bus Specification*, Philips Semiconductors, http://www.phillipssemiconductors.com
UARTs are functionally compatible with the 16550A and 16750 industry standards. The 16550A was originally produced by National Semiconductor Inc. The 16750 is produced as the TL16C750 by Texas Instruments.
Infrared Data Association http:/www.irda.org
Audio Codec '97 Component Specification, http://www.intel.com/labs/media/audio
I^2S *Bus Specification*, February 1986, Philips Semiconductors, http://www.phillipssemiconductors.com
Book of iButton Standards, http://www.ibutton.com/ibuttons/standard.pdf
Universal Serial Bus Specification, Revision 1.1; *On-The-Go Supplement to Universal Serial Bus Specification Revision 2.0; Pull-Up/Pull-Down Resistors Engineering Change Notice to Universal Serial Bus 2.0 Specification;* http://www.usb.org
OpenHCI—Open Host Controller Interface Specification for USB, Release 1.0a, http://www.usb.org
SD Memory Card Specifications Part I, Physical Layer Specification, Version 1.01, *and Secure Digital Input/Output (SDIO) Card Specification*, Version 1.0 (Draft 4), SD Association, http://www.sdcard.org
MultiMediaCard System Specification Version 3.2, http://www.mmca.org
GSM 11.11 Specification of the Subscriber Identity Module-Mobile Equipment (SIM-ME) Interface, Version 3.16.0, http://www.etsi.org. See also ISO standard 7816-3
JTAG- IEEE Std. 1149.1-1990 Standard Test Access Port and Boundary-Scan Architecture, http://standards.ieee.org

1.2 Product Overview

The PXA3xx Processor Family products are integrated system-on-a-chip microprocessors for high-performance, low-power portable handheld and handset devices. They incorporate the XScale® microarchitecture with on-the-fly voltage and frequency scaling and sophisticated power management to provide industry leading MIPS/mW performance across its wide range of operating frequencies. The processors comply with the ARM* Architecture V5TE instruction set (excluding floating point instructions) and follow the ARM* programmers model. The multimedia coprocessor provides enhanced Intel® Wireless MMX 2™ instructions to accelerate audio and video processing. The processors are available in a discrete package configuration. They provide a high degree of backward compatibility with the Marvell PXA27x Processor Family, but they offer significant performance and feature set enhancements.

The processor memory architecture offers greater flexibility and higher performance than previous core products. This architecture supports two dedicated memory interfaces for high-speed DDR SDRAM, VLIO devices, and NAND flash devices. The flexibility enables high performance “store-and-download” as well as “execute-in-place” system architectures. The processor memory architecture features a memory switch that allows multiple simultaneous memory transactions among different sources and targets. For example, the processor architecture allows memory traffic between the core and DDR SDRAM to move in parallel with DMA-generated traffic between the LCD controller and internal SRAM. These transactions block each other in an architecture with a single shared system bus.

The processor incorporates an internal Boot ROM and a Marvell® Wireless Trusted Transaction Technology module to provide flexible boot-loading options while maintaining platform security. They have up to six 128-Kbyte banks of internal SRAM for a combination of display frame buffer, program code, or multimedia data. Each bank can be configured to retain its contents when the processor enters a low-power mode.

The processor provides OS timer channels and synchronous serial ports (SSPs) that accept an external network clock input so that they can be synchronized to the cellular network.

An integrated LCD panel controller supports active and passive displays. It permits color depths of up to 18-bits per pixels (24-bits per pixel for smart panels). The LCD controller also supports hardware cursor and two display overlays.

The processor incorporates a comprehensive set of system and peripheral functions that make it useful in a variety of low-power applications. Figure 1 illustrates the system-on-a-chip PXA30x processor, Figure 2 illustrates the PXA31x processor, and Figure 3 illustrates the PXA32x processor. The diagram shows a multi-port memory switch and system bus architecture with the core attached, along with an LCD controller and USB 1.1 controllers, and internal memory. The key features of all of the sub-blocks are described in this chapter; and more details are provided in the respective chapters.

Copyright © 2009 Marvell

Figure 1: PXA30x Processor Block Diagram

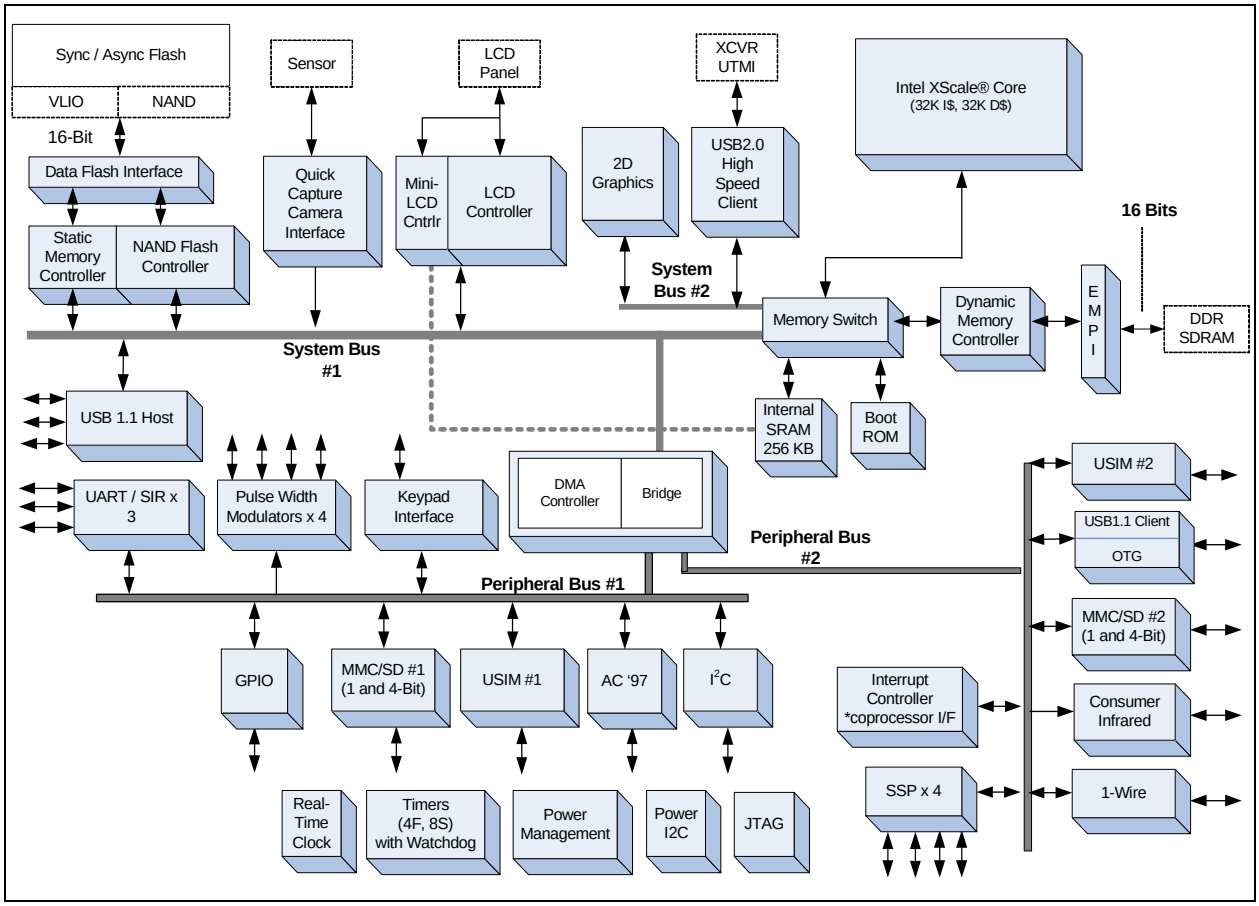
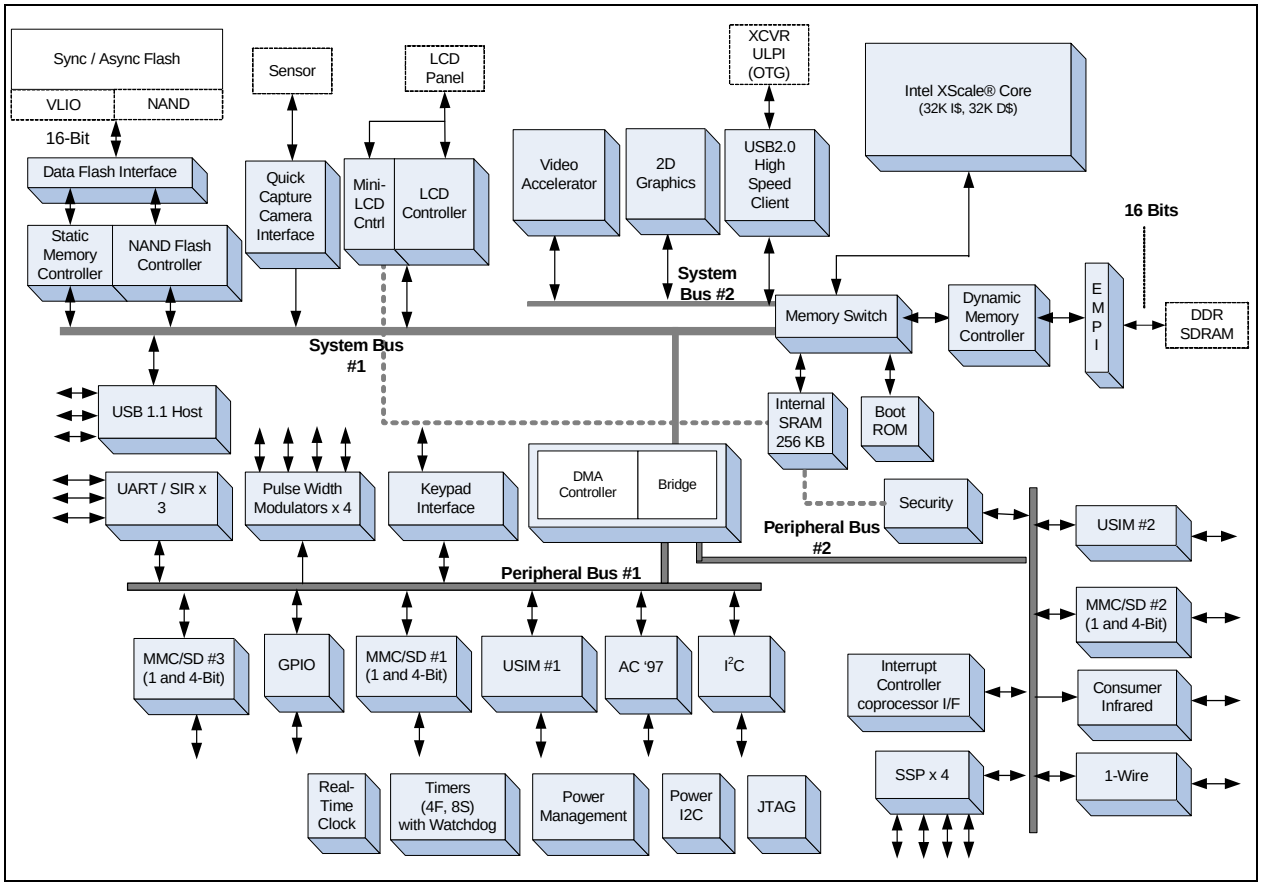
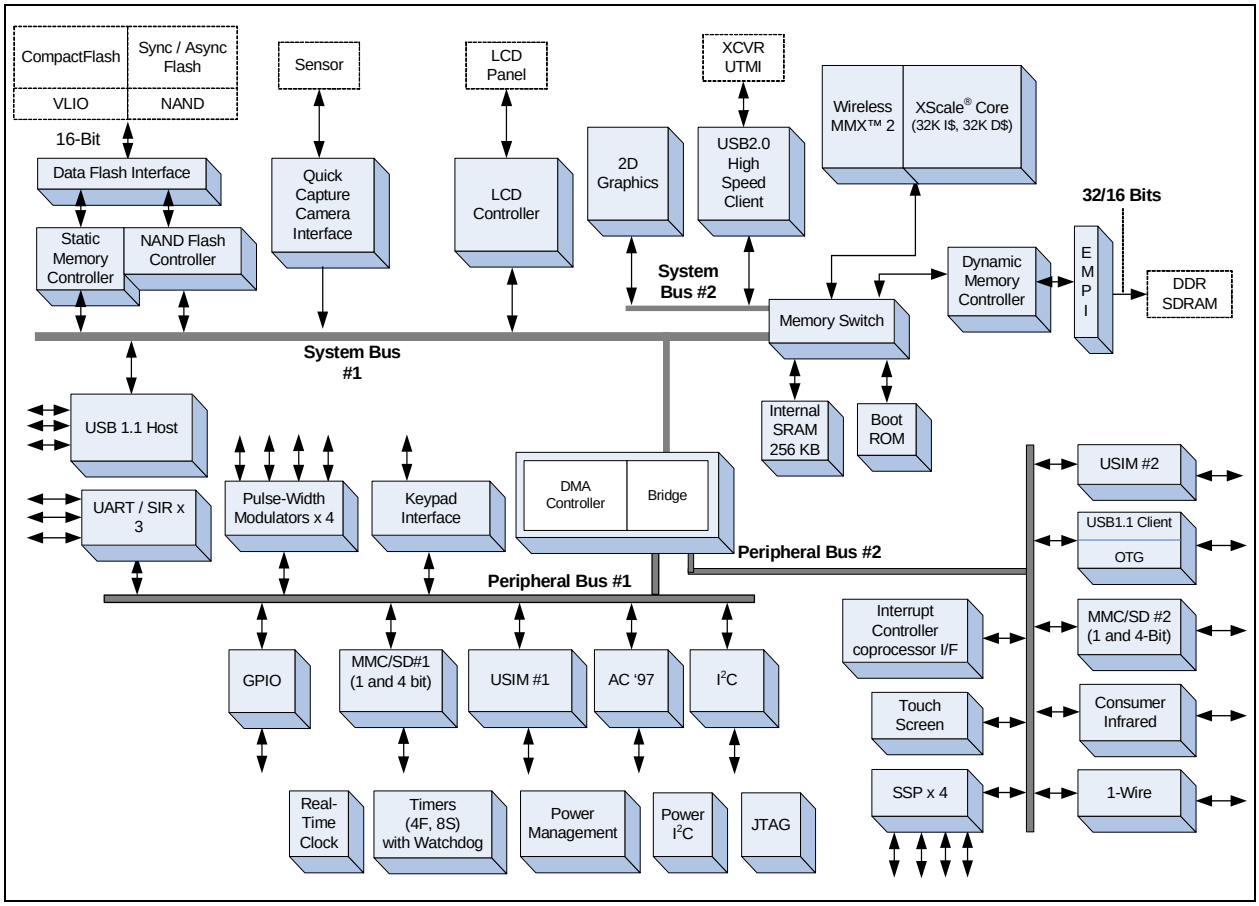


Figure 2: PXA31x Processor Block Diagram



Copyright © 2009 Marvell

Figure 3: PXA32x Processor Block Diagram



1.2.1 Application Subsystem Microarchitecture and Core

The core microarchitecture is based on a next-generation core that complies with the ARM* Architecture V5TE. The microarchitecture surrounds the core with instruction and data memory management units; instruction, and data; write, fill, pend, and branch-target buffers; power management, performance monitoring, debug, and JTAG units; coprocessor interface, MAC coprocessor; and core memory bus.

1.2.2 Application Subsystem Microarchitecture Features

- Seven-stage superpipelined RISC technology achieves high speed and ultra low power.
- Dynamic voltage management incorporates voltage and frequency on-the-fly scaling to allow applications to implement optimal performance and power.
- Media processing technology lets the multiply-accumulate coprocessor (MAC) perform two simultaneous 16-bit single-instruction multiple-data (SIMD) multiplies with 40-bit accumulation for efficient media processing.
- Power management control provides power savings via device and system low-power modes.
- 128-entry branch-target buffer keeps the pipeline filled with statistically correct branch choices.
- 32-Kbyte instruction cache keeps local copies of instructions to enable high performance and low power.
- 32-Kbyte data cache keeps local copy of data to enable high performance and low power.

- 32-entry instruction memory management unit enables logical-to-physical address translation, access permissions, and instruction cache attributes.
- 32-entry data memory management unit enables logical-to-physical address translation, access permissions, data cache attributes.
- Four-entry Fill-and-Pend buffers promote core efficiency by allowing “hit-under-miss” operation with data caches.
- Performance monitoring unit furnishes two 32-bit event counters and one 32-bit cycle counter for analysis of hit rates.
- Debug unit uses hardware breakpoints and 256-entry Trace History buffer (for flow change messages) to debug programs.
- 32-bit coprocessor interface provides a high-performance interface between core and coprocessors.
- 64-bit core memory bus with simultaneous 32-bit input path and 32-bit output path provides up to 3.2 Gbps @ 403 MHz bandwidth for internal accesses.
- Eight-entry Write buffer allows the core to continue execution while data is written to memory.

Refer to the *Intel XScale® Core Developers Manual* for additional information.

1.3 XScale® Microarchitecture Compatibility

The application subsystem core complies with the ARM* Architecture V5TE. The processor implements the integer instruction set of the ARM* Architecture V5TE. Backward compatibility for user-mode applications is maintained with the first generation of StrongARM* and previous core products. Operating systems require modifications to match the specific core hardware features and to take advantage of the performance enhancements added to this core.

Memory map and register locations are backward-compatible with the previous core handheld products (see *Intel XScale® Core Developers Manual* for exceptions). The Intel® Wireless MMX™ 2 instruction set is compatible with the standard ARM* coprocessor instruction format.

The following subsections provide a high-level overview. Refer to the respective chapters for details.

1.3.1 Multimedia Coprocessor

The core integrates a multimedia coprocessor to accelerate multimedia applications and 2-D graphics operations. This coprocessor provides a 64-bit single-instruction multiple-data (SIMD) architecture and compatibility with the integer functionality of the Intel® Wireless MMX™ 2 technology and streaming SIMD extensions (SSE) instruction sets. Key features of this coprocessor include:

- 14 new media processing instructions
- 64-bit architecture including SIMD (up to eight simultaneous eight-bit operations)
- 16 × 64-bit register file
- SIMD PSR flags with group conditional execution support
- SIMD instruction support for sum-of-absolute-differences (SAD) and multiply-accumulate (MAC) operations
- Instruction support for alignment and video operations
- Intel® Wireless MMX™ 2 and SSE integer instruction compatibility
- Superset of existing core media processing instructions

1.3.2 Internal Memory Architecture

The memory architecture features a memory switch that allows multiple simultaneous memory transactions between different sources and targets. For example, the processor architecture allows memory traffic between the core and DDR SDRAM to move in parallel with DMA generated traffic between the LCD controller and internal SRAM. In an architecture with a single shared system bus,

Copyright © 2009 Marvell

these transactions block each other. In applications with a VGA or higher display resolution, this non-blocking capability provides significantly higher overall system performance.

1.3.3 General-Purpose I/O (GPIO)

One-hundred-twenty-eight of the peripheral pins on the processor also provide software-controlled general-purpose I/O (GPIO) pin functionality. The key features of the GPIO controller are:

- As inputs, GPIO pins can be sampled or programmed to generate an interrupt from either a rising or falling edge.
- As outputs, GPIO pins can be cleared individually or set, and can be preprogrammed to either state when entering sleep mode.
- The GPIO unit does not control alternate function selection for individual pins as was the case in the Marvell® PXA27x family.

1.3.4 Power Management

The processor provides a rich set of flexible power-management controls for a wide range of usage models while enabling very low-power operation. The key features include the following:

- Five reset sources: power-on, hardware, watchdog, general-purpose I/O (GPIO), and reset upon exit from S2/D3/C4 and S3/D4/C4 modes (S3/D4/C4 is only used upon initial power up)
- Wakeup from low-power operation is supported by the available peripherals
- Functional clock gating
- Supported speeds are: 806 MHz, 624 MHz, 416 MHz, 208 MHz and 104 MHz
- Programmable frequency-change capability, with turbo settings without requiring the PLL to re-lock
- Seven power modes to control power consumption: S0/D0/C0 (normal using either run and turbo clock configurations), S0/D0CS/C0 (run operation with ring oscillator operation at 60 MHz), S0/D0/C1 (core idle), S0/D1/C2 (standby with LCD refresh), S0/D2/C2 (standby), S2/D3/C4 (sleep), and S3/D4/C4 (deep sleep)
- Dedicated programmable I^2C-based external regulator interface to power management ICs

1.3.5 One-Wire Controller

The 1-Wire bus master circuit is designed to receive and transmit 1-Wire bus data and provides complete control of the 1-Wire bus through eight-bit commands. The key features include the following:

- Reads and Writes 1-Wire bus data
- All timing and control of the 1-Wire bus are generated within the controller

1.3.6 DMA Controller

The Descriptors for the Descriptor-based DMA controller are stored in memory and are fetched upon receipt of a DMA request from a particular unit. The Descriptors support looping and branching constructs. The DMA controller provides the following key features:

- Memory-to-memory, peripheral-to-memory, and memory-to-peripheral transfers in Flowthrough mode.
- Flowthrough mode for transfers between flash and DDR SDRAM.
- 32 channels, and 92 peripheral-device requests, with the capability of pre-programming any request to any channel.
- Employs a priority mechanism to process active channels (four channels with outstanding DMA requests at any given time).

Copyright © 2009 Marvell

- Each of the 32 channels can be operated in either Descriptor-Fetch mode or No-Descriptor-Fetch mode.
- Special Descriptor-Comparison and Descriptor-Branching modes.
- Retrieves trailing bytes from the Receive peripheral-device buffers.
- Programmable data-burst sizes (8, 16, or 32 bytes) and programmable peripheral-device data widths (byte, half-word, or word).
- Up to (8 Kbytes–1 byte) of data transfer per Descriptor; larger transfers can be performed by software chaining multiple Descriptors.
- Flow-control bits to process peripheral-device requests. Requests are processed only if the flow-control bit is set.

1.3.7 Interrupt Controller

The interrupt controller, which masks and prioritizes all internal interrupts, is accessed either through memory-mapped or coprocessor registers. Key features include:

- Peripheral interrupt sources can be mapped to normal (IRQ) or fast (FIQ) interrupt request
- Each interrupt source can be independently enabled
- Priority mechanism to indicate highest-to-lowest priority interrupts
- Accessible via the coprocessor interface for fast access
- Accessible as a memory-mapped peripheral for backward compatibility

1.3.8 Real-Time Clock (RTC)

The real-time clock is a 32-bit counter with trim control that runs off the 32.768 kHz crystal oscillator. The general features of the RTC are:

- Timer
  - User-programmable free-running counter
  - User-programmable alarm register
  - Resolution of one second
- Wristwatch
  - User-programmable free-running counter displaying time of the day in units of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - User-programmable alarm registers to generate alarms in units of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - Resolution of one second
- Stopwatch
  - User-programmable Counter register that displays the time elapsed between two events in units of hours, minutes, seconds and one-hundredth of a second
  - Two user-programmable Alarm registers to generate alarms in units of hours, minutes, seconds, and one-hundredth of a second
  - Resolution of one-hundredth of a second
- Periodic interrupts
  - User-programmable Alarm register to generate periodic interrupts at regular intervals
  - Resolution of one millisecond
- Trimmers
  - User-programmable Trimmer register to generate a precise 1-Hz clock for the timer and the wristwatch

Copyright © 2009 Marvell

1.3.9 Operating System Timers

The operating-system timer provides:

- A single-counter operating at 3.25 MHz
- Four match registers
- Watchdog function

The processor also has an additional timer set that provides:

- Eight independent channels, each consisting of:
  - Counter
  - Match register
  - Control register
- Independent clock for each counter, selectable by software:
  - 32.768 kHz clock for low power
  - 13 MHz clock for high accuracy
  - Externally supplied clock for network synchronization
- Counter resolutions of 1/32768th of a second, one millisecond, and one microsecond
- Periodic and one-shot timers
- Two external synchronization events (EXT_SYNC<1:0>)
- Two periodic outputs (CHOUT<1:0>)
- Operation during reduced-power modes (S0/D1/C2, S0/D2/C2 and S2/D3/C4)

1.3.10 Performance Monitor

The performance-monitoring functionality is provided using core performance counters. It allows additional functions to be monitored within the core. Features include the following:

- Four 32-bit performance counters allowing four unique events to be monitored simultaneously
- One 32-bit counter that counts core clock cycles
- Core can monitor over 70 events
- Eight ASSP-level events are fed to the core that can monitor four of the eight ASSP-level events
- Interrupt indicating that an overflow on one of the counters has occurred, allowing the software routine to read and accumulate the register contents
- Counters are accessible using Coprocessor 14 on the core

1.3.11 Test

The boundary-scan interface has the following features:

- JTAG interface
- Conforms to the IEEE Std. 1149.1 – 1990 and IEEE Std. 1149.1a-1993, *Standard Test Access Port and Boundary-Scan Architecture*
- Test-access port with dedicated pins: TDI, TMS, TCK, nTRST, and TDO

1.3.12 Power I^{2}C Controller

The PWR I^{2}C controller allows the PWR I^{2}C unit to interface to compatible Power I^{2}C devices attached to the Power I^{2}C bus. This controller is software programmable and outputs I^{2}C commands as defined in the “PWR_I2C Bus Interface Unit” chapter. The key features include the following:

- I^{2}C compliant
- Automatically outputs I^{2}C commands when required for power mode or voltage changes
- Multi-master and arbitration support

- Standard-mode operation of 100 Kbps
- Fast-mode operation of up to 400 Kbps
- Start-of-day operation with 32.768 KHz operation of 100 bps

1.3.13 External Memory Interfaces

Two memory buses connect external memory devices:

- Data Flash Interface (DFI). A 16-bit address/data multiplexed bus that supports NAND flash and static memory devices. The 16-bit data flash/static DFI bus provides 16-bits of multiplexed address and data with various control signals. It supports 26-bit addressing by latching the address on the address/data lines in two phases.
- External Memory Pin Interface (EMPI). Up to 32-bit high-speed dedicated bus for DDR SDRAM, devices with DDR-like interfaces. The 32/16-bit EMPI bus provides a sophisticated mechanism for software to specify bus slew rates and pullup/pulldown strengths. It provides 16 data lines, 16 address lines, and several control signals.

1.3.13.1 Dynamic Memory Controller

The dynamic memory controller interfaces to DDR SDRAMs. It connects only to the EMPI. Key features include:

- One or two partitions of DDR SDRAM running at a maximum of 130 MHz clock rate; multiple partitions of DDR SDRAM must use identical devices
- Up to 32-bit data bus width
- Up to 32-bit wide DDR SDRAM devices
- Programmable impedance control
- Programmable calibration of strobe signals to and from DDR SDRAM
- DDR SDRAMs automatically placed into Self-refresh mode before entering Sleep mode
- Programmable Powerdown mode for power savings
- 1.8 V low-power DDR SDRAM

Refer to the *PXA3xx Processor Family Vol II: Memory Configuration Developers Manual for* additional information.

1.3.13.2 Static Memory Controller

The static memory controller interfaces to SRAM-like variable latency IO memories. Key features include:

- Support for NOR flash devices
- Supports up to four partitions of 16-bit static memory or VLIO
- Support for NAND flash devices with NOR flash interfaces
- Programmable output clock
- Programmable Powerdown mode for power savings
- Supports 1.8 V and 3.0 V devices

Refer to the *PXA3xx Processor Family Vol II: Memory Configuration Developers Manual* for additional information.

1.3.13.3 NAND Flash Controller

The NAND flash controller manages external data, flash memory typically used to hold the operating system image and functions as a non-volatile mass storage hard disk drive. Key features include:

- Support for third-party NAND flash devices
- Hardware ECC

Copyright © 2009 Marvell

- A/D-multiplexed interface to data flash devices
- Connection to the DFI bus
- Two chip selects
- Both 8- and 16-bit wide flash devices; two 8-bit-wide identical flash devices can be used in parallel to support 16-bit-wide operation
- System DMA for data-flash data transfers
- Computes ECC and corrects single-bit errors and detects 2 bit errors per page. Also capable of handling single-cell failures in Multi-Level-Cell (MLC) NAND flash.
- Programmable read/program/erase timings
- Interrupts to indicate page and command completion, flash-ready status, bad blocks, bit errors, and command and data write/read requests
- Supports 1.8 V, 3.0 V and 3.3V devices.

Refer to the *PXA3xx Processor Family Vol II: Memory Configuration Developers Manual for* additional information.

1.3.13.4 Internal SRAM Memory

The internal SRAM can be used in a variety of ways to provide higher system performance and lower power by reducing off-chip memory accesses. A typical use of this SRAM is as an LCD frame buffer with display resolutions up to QVGA. Key features of the internal memory module include:

- 256-Kbytes of on-chip static RAM arranged as two banks of 128-Kbytes (PXA30x and PXA31x processors only)
- 768-Kbytes of on-chip static RAM arranged as two banks of 128-Kbytes (PXA32x processor only)
- Bank-by-bank power management for reduced power consumption
- Support for byte Writes

Refer to the *PXA3xx Processor Family Vol II: Memory Configuration Developers Manual for* additional information.

1.3.13.5 Multimedia Card, SD Memory Card, and SDIO Card

The processor provides two Multimedia Card (MMC)/SD-Card/SDIO interfaces. Each controller offers the following features:

- MMC version 3.3.1, SD version 1.1, and SDIO version 1.0 protocols
- Single- and four-bit data transfers in MMC, SD and SDIO modes
- Data transfer clock of up to 26 MHz
- Two operating modes: MMC/SD/SDIO mode and SPI mode; MMC/SD/SDIO mode supports MMC, SD, and SDIO communication protocols; SPI mode supports the SPI communications protocol
- Controller turns clock on and off, based on status of FIFOs, to prevent overflows and underruns
- Dual Transmit FIFOs and dual Receive FIFOs for each interface
- All valid MMC and SD/SDIO protocol Data Transfer modes are supported
- Interrupt-based interface to application to control software interaction
- For stream-write protocol, only data sizes of 10 bytes or more are allowed
- Using the MMC communications protocol, multiple MMC cards are supported
- Using the SD or SDIO communications protocol, one SD or SDIO card is supported
- Using the SPI communications protocol, up to two MMC or SD/SDIO cards are supported; Mixed card types are supported only for the SPI communications protocol.
- 1.8 V or 3.3 V card support provided with a dedicated supply pin

Refer to *PXA3xx Processor Family Vol II: Memory Configuration Developers Manual for* additional information.

1.3.14 Graphics and Input Controllers

The PXA3xx Processor Family supports a wide range of LCD, multimedia, and input controllers. The following sections briefly describe those controllers.

1.3.14.1 LCD Panel Controller

Features of the LCD controller are as follows:

- Active or passive single-panel displays of 8, 16, or 18 bpp
- LCD panels with internal frame buffer; up to 24 bpp is supported
- The following display sizes (all in either landscape or portrait orientation)
  - 176 × 208
  - 176 × 220
  - 240 × 240
  - 320 × 240 (QVGA)
  - 320 × 320
  - 320 × 480
  - 480 × 480
  - 640 × 480 (VGA)
- Hardware color-space conversion from YCbCr to RGB for video streams
  - Upscaling for YCbCr 4:2:2 to YCbCr 4:4:4
  - Color space conversion CCIR 601 - YCbCr 4:4:4 to RGB 8:8:8
  - Conversion from true color (RGB 8:8:8) to high color (RGB 5:5:5) and RGBT
- Three 256-entry by 25-bit internal color-palette RAMs (one for each overlay and base), programmable to be automatically loaded at the beginning of each frame
- Command data RAM (16 × 9 bits) to hold command data
- One base layer plus two overlays; maximum size of each overlay can equal the display size
- Integrated seven-channel DMA: one channel for base plane, one channel for Overlay 1, three channels for Overlay 2 and one channel for the hardware cursor
- Hardware cursor
- Programmable pixel clock from 203 kHz to 104 MHz (104 MHz/512 to 208 MHz/2)
- Little-endian ordering of pixels in frame buffer
- Programmable wait-state insertion at beginning and end of each line
- Programmable polarity for output enable, frame clock, and line clock
- Programmable interrupts for Input and Output FIFOs (underruns)

Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual for* additional information.

1.3.14.2 Mini-LCD Panel Controller

The mini-LCD controller provides an interface between the processor and a flat-panel display module in low-power modes of operation for low-power (S0/D1/C2) operation. The mini-LCD controller supports active (TFT) panels only. The LCD controller has these key features:

- RGB 1:5:5:5 pixel format only
- 16-bit active panels only

Copyright © 2009 Marvell

- The following display sizes:
  - 176 × 208
  - 176 × 220
  - 240 × 240
  - 320 × 240
  - 320 × 320
  - 320 × 480
  - 480 × 480
  - 640 × 480
- Programmable wait-state insertion at beginning and end of each line
- Programmable polarity for output enable, frame clock, and line clock
- Operates at 39 MHz frequency

Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual for* additional information.

1.3.14.3 Camera Image Capture Interface

The camera image-capture interface has the following features:

- Vertical and horizontal resolutions of:
  - 176 × 144
  - 352 × 288
  - 320 × 240
  - 640 × 480
  - 1280 × 1024
  - 1600 × 1200
  - 2048 × 1536
  - 2048 × 2048
  - 2560 × 2048
- Programmable sensor clock output from 187 kHz to 52 MHz
- Pixel clock received from 187 kHz to 52 MHz
- Programmable interrupts for FIFO overflow, end of line, and end of frame
- 8- and 10-bit RAW (RGGB, CMYG) Capture modes
- Master mode operation
- Programmable interface timing signals for external synchronization signaling
- Preprocessed YCbCr 4:2:2 Planar-Capture mode
- RAW (RGGB, CMYG) Capture modes:
  - Packing of 8- and 10-bit RAW pixel data up to 2560 × 2048
  - Pixel processing preview chain up to 1280 × 1024 resolution (SXGA)
- Three programmable 64-element look-up-tables (LUT)
  - Three independent mapping functions, ($f_R(x)$, $f_G(x)$, and $f_B(x)$)
  - Companding from 10-bit to 8-bit RAW data
  - Programmable black-level clamp (BLC) offset
- Histogram unit generates statistics for image data
  - Performs statistics on 8- or 10-bit data
  - Incrementer saturates to avoid rollovers

- Up to 64 K pixels, (2^{16}) per data value
- Can perform statistics on 8-bit or 10-bit data stream with a 32-bit result

- Dead-pixel substitution unit supports sensor resolutions up to 2560 x 2048
  - Up to 128 pixels of any color can be substituted
- Scaling support for 2:1 /4:1 image resizing for RAW RGGB or YCbCr 4:2:2 image data
  - Preprocessed YCbCr 2:1 and 4:1 scaling provided up to 704 x 576 resolution
  - RAW RGGB 2:1 up to 704 x 576 resolution and 4:1 scaling provided up to 1280 x 1024 resolution
- Color-management support for RAW digital viewfinder and video-clip capture
  - Programmable coefficients for 3×3 matrix multiplication for color and tone correction
  - Programmable coefficients for color space conversion from RGB to YCbCr 4:2:2

Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual for* additional information.

1.3.14.4 Graphics Controller

Features of the graphics controller are as follows:

- Graphics instruction list parser
- Two Source buffers
- Three Destination buffers (including internal and external display buffers (front/back buffers))
- Support for multiple pixel formats
- 8-bit to 64-bit color
- 1-bit transparency or 16-bit alpha
- 2-D graphics acceleration
  - Alpha blend BLT
  - Scale BLT
  - Bias BLT
  - Chroma key BLT
  - Stretch BLT
  - Decimate BLT
  - Line draw
  - Anti-aliased line draw
  - Chroma key stretch BLT
  - Rotate BLT
  - Raster OP BLT (serves as source copy BLT)
  - Pattern copy BLT
  - Color fill
- Local buffer to reduce bus bandwidth for 2-D operations
- Pixel ALU
- Eight 64-bit Storage registers
- Three 16-bit channels for pixel data operations
- Pixel format conversion support
- Pixel multiplier/accumulator processing architecture
- Support for alpha and transparency operations
- Full raster operation support

Copyright © 2009 Marvell

Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual* for additional information.

1.3.14.5 Keypad Controller

The keypad controller has the following features:

- Three configurations:
  - Eight-by-eight matrix keys and eight direct keys
  - Eight-by-eight matrix keys, six direct keys, and one rotary encoder (two pins for the rotary encoder)
  - Eight-by-eight matrix keys, four direct keys, and two rotary encoder (four pins for the rotary encoders) (PXA30x processor and PXA31x processor only)
- Matrix-key interface with manual and automatic scan
- Interrupt or Low-power mode Wakeup event generated on key press; separate matrix- and direct-key interrupt enables
- Polling
- Key-debounce logic to check for key debounce for both matrix and direct keypads

Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual for* additional information.

1.3.14.6 Hardware Video Accelerator (PXA31x Processor Only)

The hardware video accelerator unit provides hardware assistance for processing still images and video streams. It offers high-performance, low-power video acceleration on the PXA30x processor or PXA31x processor but not on the PXA30x processor. The unit supports the following standards with varying amounts of hardware assistance:

- Video coding standards such as H.264, MPEG-2, and MPEG-4
- Proprietary standards such as Microsoft* Windows Media Viewer (WMV)
- Image compression standards such as JPEG.

Actual performance depends on the amount of hardware assist and system-level constraints, including but not limited to DDR memory bandwidth, memory latency, and the number of concurrent processes running.

Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual for* additional information.

1.3.14.7 ADC and Touchscreen Interface (PXA32x Processor Only)

The PXA32x processor has a touchscreen interface. Together, these modules create a touchscreen controller system (ADC_TS). This chapter describes guidelines when using the ADC_TS.

The ADC_TS interface has the following features:

- Bias generator with internal reference
- Analog-to-digital converter (ADC) with an input mux
- X-Y sequencing logic
- Touchscreen interface (TSI) with pad-interface logic
- Refer to *PXA3xx Processor Family Vol. III: Graphics and Input Controller Configuration Developers Manual for* additional information.

1.3.15 Serial Ports

The PXA3xx Processor Family provides a rich set of serial controllers for general-system use. All ports can be accessed through programmed I/O or through Descriptor-based DMA transfers. Pins on ports not being used can be configured as GPIOs. The following sections describe these ports.

1.3.15.1 USB 1.1 Device Controller (PXA32x and PXA30x Processors Only)

The USB client controller has these key features:

- USB Revision 1.1 compliant — 12 Mbps, half duplex
- 23 programmable endpoints (Endpoint 0 excluded):
  - Endpoint type: bulk, isochronous, or interrupt
  - Endpoint direction: in or out
  - Endpoint maximum packet size
  - Programmable configuration, interface, and alternate interface setting numbers
- Endpoint 0 for control in and out
- Four configurations:
  - Three programmable configurations with up to seven interfaces with seven alternate interface settings
  - Default Configuration 0 with one interface and Control Endpoint 0
- Configurable 4-Kbyte memory for endpoint data storage

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.2 USB 1.1 Host Controller

The USB host controller has the following key features:

- USB Rev. 1.1 compatible
- Up to three host ports
- Both low-speed and full-speed USB devices
- Open host controller interface (OHCI) Rev 1.0a compatible
- Root hub supports three chained downstream ports
- Built-in DMA

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.3 USB 2.0 High Speed Device Controller

The USB 2.0 high-speed device controller has the following key features:

- USB Rev. 2.0 high-speed/full-speed compliant
- Up to 15 concurrent programmable endpoints:
  - Endpoint type: bulk, isochronous, or interrupt
  - Endpoint direction: in or out
  - Endpoint maximum packet size
  - Programmable configuration, interface and alternate interface setting numbers
- Endpoint 0 for control in and out
- 16 Configurations:
  - 15 programmable configurations with up to 15 interfaces with 15 alternate interface settings each
  - Default Configuration 0 with one interface and Control Endpoint 0

Copyright © 2009 Marvell

- Configurable 8 Kbyte memory for endpoint data storage
- Interfaces to industry-standard UTMI bus (PXA32x and PXA30x processors only)
- Interfaces to industry-standard ULPI bus (PXA31x processor only)

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.4 Synchronous Serial Ports (SSP)

The SSP controllers support these protocols:

- Programmable serial protocol (PSP) with programmable frame sync and programmable start and stop delays
- Texas Instruments Synchronous Serial Protocol* (SSP)
- Motorola Serial Peripheral Interface* (SPI) protocol
- Inter-IC Sound (I^2S) protocol (emulated using the PSP protocol). See “Monahans Processor I^2S Emulation Using SSP/PSP Application Note.”
- Four SSPs
- Up to 26 Mbps transfer rate with internal clock generation
- Packed mode to allow double-depth FIFOs if sample less than 16 bits wide
- Sample data formats from 8, 16, 18, and 32 bits of serial data
- Network mode for operation on a time-slotted bus
- Master or slave operation for both clock- and frame-sync signals
- Receive-without-transmit operation
- Flexible clock source selection from the 13-MHz master clock, the network clock input, or the dedicated SSP external clock input
- Audio clock control to provide a 4x or 8x output clock to support most standard audio frequencies

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.5 AC’97 CODEC Interface

The AC’97 CODEC interface supports these key features:

- Independent channels for stereo pulse-code modulated (PCM) In, stereo PCM Out, surround PCM out, center/LFE PCM out, MODEM Out, MODEM-In and mono Mic-in; these channels support 16-bit samples only
- Multiple-sample-rate AC’97 2.3 codecs (48 kHz and below). The AC’97 controller depends on the codec to control the varying rate
- Read/write access to AC’97 registers
- Secondary CODEC support
- Optional AC97_SYSCLK output (for CODECs without oscillators or crystals)

The AC’97 controller does not have the following optional AC’97 Rev 2.3 features:

- Optional double-rate sampling (n+1 sample for PCM L, R)
- 18- and 20-bit sample lengths

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.6 UARTS

Three UARTs are provided. UART 1 supports the full set of modem control signals. All UARTs have these features:

- Functionally compatible with the 16550A and 16750
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data
- Independently controlled Transmit, Receive, Line-status, and Data-set interrupts
- Programmable serial-interface characteristics:
  - 7- or 8-bit characters
  - Even, odd, or no parity detection
  - 1 stop-bit generation
  - Baud-rate generation of the following frequencies: 9600, 19.2K, 38.4K, 57.6K, 115.2 K, 230 K, 460 K, 921 K, 1.8M and 3.6M baud
  - False start-bit detection
- Complete status-reporting capability
- Break generation and detection
- Internal diagnostic capabilities include:
  - Loop-back controls for communications-link fault isolation
  - Break, parity, overrun, and framing error simulation
- All UARTs can operate in Slow-infrared (SIR) IRDA mode
- All UARTs have hardware flow-control support:
  - nRTS (output) controlled by UART receiver FIFO
  - nCTS (input) from modem controls UART transmitter

UART 1 has these additional modem control functions:

- nDSR
- nDTR
- nRI
- nDCD

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.7 Consumer Infrared Controller

The consumer infrared unit (CIR) enables the processor remotely to control consumer devices such as televisions and VCRs. There are several existing standards in the market (RC-5, RC-5 extended, RC-6, etc.). The flexible, generic CIR module design makes it compatible with existing standards.

The CIR interface has these features:

- Supports existing standards—RC-5, RC-5 extended, RC-6
- Programmable symbol length (symbols represent normal “1” or “0” and short “1” or “0”)
- Four different symbol types
- Programmable modulated frequency
- Programmable duty cycle for the modulation wave (for power saving)
- Double buffer for input stream
- Manchester coding
- One system-level interrupt that indicates empty buffer or end of transmission

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

Copyright © 2009 Marvell

1.3.15.8 Pulse-Width Modulation Unit (PWM)

The PWM unit consists of four independent channels. Data can be provided either by DMA or CPU programmed I/O.

Key features are:

- Four pulse-width-modulated output channels
- Enhanced period control through 6-bit clock divider and 10-bit period counter
- 10-bit pulse control

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.9 Universal Subscriber ID Controller

The processor provides two Universal Subscriber Identity Module (USIM) interfaces. Each controller has the following features:

- Compatible with any USIM SmartCard that is compliant with standard ISO 7816-3 and 3G TS 31.101
- Control lines for two-level voltage supply (1.8 V and 3 V)
- USIM SmartCard reset-pin control (using reset-pin control and power-supply control, warm/cold reset can be software initiated)
- T=0 and T=1 protocols
- Programmable SmartCard clock frequency
- Any combination of these clock-rate conversion Factor F and bit-rate adjustment Factor D:
  - F = {372, 512, 558}
  - D = {1,2,4,8,16, 32, 12, 20}
- Auto-error signal in T=0 Receive mode
- Auto-character repeat in T=0 Transmit mode
- Transforms inverted format to regular format, and *vice versa*
- Programmable time periods and interrupt requests:
  - Block-guard time period
  - Extra-guard time period
  - Character-waiting time period
  - Block-waiting time period
  - Time-out period
  - CPU interrupt request on an error-signal detection
  - CPU interrupt request when a SmartCard is connected

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

1.3.15.10 Two Wire Serial Interface (TWSI) Serial Bus Port

The TWSI has these features:

- I^2C compliant
- Multi-master and arbitration support
- Standard-mode operation at 100 kbps
- Fast-mode operation up to 400 kbps
- Start-of-day operation with 32.768 kHz operation of 100 bits/sec

Refer to *PXA3xx Processor Family Vol. IV: Serial Controller Configuration Developers Manual for* additional information.

Copyright © 2009 Marvell

2 System Architecture Overview

This chapter outlines the core implementation, types of module resets, and signal description information for the processor. These processors are implementations of the XScale® microarchitecture, which are called the *application subsystem* throughout this document. The characteristics of this particular implementation include:

- Several coprocessor registers
- Little-endian operation
- Semaphores and interrupts for processor control
- Multiple reset mechanisms
- Sophisticated power management
- Highly multiplexed pin usage

2.1 Microarchitecture Implementation Options

The core implementation in the processor includes the options outlined in this chapter. Most of these options are specified within the coprocessor register space, as described in Section 2.16.1, Application Subsystem Microarchitecture Coprocessor Register Summary, on page 53. Access to coprocessors other than CP14 and CP15 is controlled by the Coprocessor Access Register (CPAR; see Section 2.16.6.4). To accommodate the additional functionality of the application subsystem core, registers in CP14 and CP15 are added or augmented. To accommodate the Intel* Wireless MMX multimedia extensions, the registers in CP0 and CP1 are added or augmented. For details, refer to the *Marvell PXA27x Processor Family Developers Manual.*

The following subsections of this chapter describe the coprocessor registers:

- Section 2.16.2, Interrupt Controller Registers
- Section 2.16.3, Performance Monitoring Registers
- Section 2.16.4, Clock Configuration and Power Management Registers
- Section 2.16.5, Coprocessor Software Debug Registers
- Section 2.16.6, Coprocessor 15

2.2 Endianness

Endianness is the convention that describes the order in which bits in a word are stored in memory. The processor operates only in little-endian mode, so the least-significant byte (LSB) of a value is stored in memory at a lower address than the most significant byte (MSB). For example, the value 0x8765_4321 at address 0x0 in a little-endian system appears as shown in Table 2.

Table 2: Little-Endian Value Encoding

Address	0	1	2	3
Byte Value	0x21	0x43	0x65	0x87

2.3 Memory Switch Versus System Bus

There are two internal high-speed system buses: System Bus 1 and System Bus 2. Both buses connect to the memory switch, which in turn connects directly to the core and to the internal and external memories.

2.4 I/O Ordering

The processor queues accept memory requests from the many internal masters such as the DMA controller, USB host, LCD controller, camera interface, 2D graphics controller, etc. Operations issued by each master complete in the order they are received. However, operations from one master can be interrupted by operations from another master. The processor does not provide a software method to control the order of operations from different masters.

Loads and Stores to internal addresses are generally completed more quickly than those to external addresses. The difference in completion time allows for the possibility that one operation is received before a second operation, but the second operation completes before the first.

In the following sequence, the store to the address in r4 completes before the store to the address in r2 because the first store waits for memory in the queue while the second is not delayed.

```
str r1, [r2]; store to external memory address [r2]
str r3, [r4]; store to internal (on-chip) memory address [r4]
```

If the two Stores are control operations that must be completed in order, the recommended sequence is to insert a Load to an unbuffered, uncached memory page followed by an operation that depends on data from the Load:

```
str r1, [r2]; first store issued
ldr r5, [r6]; load from external unbuffered, uncached address ([r2] if possible)
mov r5, r5; nop stalls until r5 is loaded
str r3, [r4]; second store completes in program order
```

2.5 Accessing Peripherals on Internal Peripheral Bus

There are two peripheral buses: Peripheral Bus 1 and Peripheral Bus 2. The peripheral bus modules connect to System Bus 1 through the DMA/bridge unit. Peripherals on the peripheral bus can be accessed with either programmed I/O using the bridge or a DMA transfer. The peripherals and their operation are described in separate volumes.

2.5.1 Programmed I/O Operations Using the Bridge

The processor can read and write the peripheral registers and FIFOs on the peripheral bus using the bridge (see Figure 1, Block Diagram, on page 5). All internal registers of the peripherals must be accessed using word access loads and stores. Internal register and FIFO space must be mapped as non-cacheable. Byte and half-word accesses to internal registers are not permitted and yield unpredictable results. The FIFOs of some of the peripherals on the peripheral bus can be accessed using byte, half-word, or word access loads and stores. Refer to individual peripheral chapters for details.

Some peripherals on the peripheral bus cannot receive or transmit data through the DMA controller. These devices use programmed I/O for all data transfers. Refer to individual peripheral chapters for details.

Copyright © 2009 Marvell

2.5.2 Data Transfer Using DMA

Some peripherals on the peripheral bus receive or transmit data through the DMA controller. The DMA controller can be programmed to transfer from the peripheral to the memory (Receives) or from the memory to the peripheral (Transmits).

For *Transmits*, if the DMA Descriptor has a byte count that is not an integer multiple of the transfer size, then at the end of the Descriptor, the DMA controller performs a transfer that is shorter than the transfer size. This is the trailing-bytes case for Transmits. For every peripheral request on Transmits, the DMA controller transfers the number of bytes equal to the size setting or number of bytes in the length setting in the DMA Command register of the channel, whichever is smaller. The DMA controller can perform a transfer shorter than its transfer size at the beginning of the Descriptor if the source address is not aligned to an 8-byte boundary.

Most peripherals can transmit the trailing bytes without any processor intervention. Refer to the individual peripheral chapters for trailing-byte programming requirements.

For *Receives*, the peripheral can receive data at the end of the data transfer and cannot receive any more data for a predefined time (time out), or it can receive an end-of-packet indication from the external CODEC/peripheral. If the amount of data received is below the peripheral threshold setting, then a trailing-bytes situation exists. For every peripheral request on Receives, the DMA controller transfers one of the following (whichever is smaller):

- Number of bytes equal to the size setting
- Number of bytes in the length setting in the DMA Command register of the channel
- Number of trailing bytes in the peripheral FIFO

The DMA controller can perform a transfer that is shorter than its transfer size at the beginning of the Descriptor if the target address is not aligned to an 8-byte boundary.

For Receives, the peripheral/DMA controller can be programmed to handle the trailing-byte situation without processor intervention. If the DMA controller has data in its Descriptor, it can completely receive all the bytes in the peripheral and jump to the next Descriptor, continue with the current Descriptor, or stop. If the DMA does not have data in its Descriptor and if the channel has stopped, the DMA controller cannot receive any more data from the peripheral. In this instance, the peripheral informs the processor through an interrupt. The processor then uses the programmed I/O mode through the bridge to read the remaining bytes from the FIFO. Refer to the individual peripheral chapters for more information.

2.6 Peripheral Access on Internal System Buses

Peripherals on system bus, such as the LCD controller, camera interface, graphics controller, mostly use their own internal DMA controller to access the system. The exception is the data flash controller, which uses the system DMA controller to transfer data. The core can read/write the internal registers within these peripherals, but typically the core cannot read or write the FIFOs within these peripherals. The DMA controller within the peripheral handles all Reads and Writes to these FIFOs. Refer to the individual peripheral chapters for more information.

2.7 DMA/Peripheral Split Transactions

Processor accesses to the peripheral bus are by default split (posted) operations. For a Read operation, there are two system-bus transactions: an initial transaction to send the request, and a separate data transfer to return the read data. During the gap between these two operations, the system bus is relinquished and can be used by other devices.

For a Write operation, the system-bus transaction completes when the Write transfers to the DMA/bridge rather than waiting until it reaches the actual peripheral.

In both cases, operations complete on the peripheral bus in strict order of issue on the system bus. However, for Writes in particular, completion of the Write from the perspective of the processor does

Copyright © 2009 Marvell

not mean that the Write takes effect. To guarantee that a Write takes effect, a Read to any peripheral bus location is required. After the Read completes, all earlier Writes take effect.

2.8 System Bus Arbiters

The two internal high-speed system buses support multiple clients — the bridge (with DMA controller), the LCD controller, the USB host controllers, camera interface, data flash controller and SGPR on System Bus 1, and 2-D graphics and USB2.0 client on System Bus 2. Each system bus is implemented as a multiplexer (versus a three-state approach) and the clients can request the bus without any limitations. An arbiter on each bus arbitrates for bus access. Bus arbitration is programmable through the ARB_CNTRL register. The arbiters have the following features:

- Programmable client weights
- Software selectable bus parking
- Bus locking

2.9 System Access Latencies

There are multiple masters in the system—for example, DMA controller, camera interface, and LCD controller. All accesses to external memory from any of these masters flow through the switch and memory controllers. The memory controllers have limited internal buffers that function as FIFOs. All requests are executed in the order received.

In performing an access to an external memory, a request from the master must first arbitrate for the respective system bus and then transfer on that system bus. Software can program the respective system-bus arbitration priority to enable one of the masters to access the system bus with higher priority than other masters.The request is then queued in the respective system bus to a switch bridge. The access then transfers across the switch to either the dynamic memory controller or static memory controller where it is arbitrated with the other switch masters to determine which access is performed (arbitration within the memory controllers is not programmable).

After the access, the data is returned directly to the switch bridge for the respective system bus, where the system bus is again requested and, when the system bus is granted, data is returned to the original master.

To compute the worst-case latency for an external memory transfer in a very busy system in which all internal bus masters are continuously trying to access a system bus, Marvell assumes that the master has highest priority programmed in the arbiter.

The following transfers can occur ahead of the current master accessing the external bus and must be accounted for:

- One current external bus transfer
- Several pending transfers in the switch queue
- Multiple requests queued in the memory controllers (worst case is three requests from this master plus up to eight requests from all other masters)
- Any other transfers pending within this master
- An SDRAM refresh
- Potentially a dynamic memory controller recalibration cycle (for details, see the chapter on the dynamic memory controller).

To compute the worst-case latency for an internal-memory transfer in a very busy system, Marvell assumes that the master has highest priority programmed in the arbiter and that the internal memory bank is not in Standby or Sleep mode and the queue is empty.

The following transfers can occur ahead of the current master acquiring the system bus and must be accounted for:

- One current external bus transfer

Copyright © 2009 Marvell

- Several pending transfers in the switch queue
- Multiple requests queued in the memory controllers (worst case is three requests from this master plus up to eight requests from all other masters)
- Any other transfers pending within this master.

Because any of the internal memory accesses are generally lower in latency, place into internal memory any accesses with a critical latency requirement, such as USB host Isochronous buffer data.

2.10 Achieving Optimum Performance

Software must comply to the following guidelines to achieve optimum performance in most use cases:

- Each peripheral must be programmed to initiate transfers with maximum supported transfer size for that peripheral. For example, the DMA controller Descriptor must be programmed to have a transfer size of 32 bytes.
- Whenever possible, program the peripheral to generate naturally aligned transfers. Table 3 summarizes aligned-transfers. For example, the starting address of the LCD controller Source Address Register must be 32-byte aligned.

Table 3: Naturally Aligned Transfers

Byte Count	Address
1	Any address (only used by static memory controller interface)
2	A[0] = 1’b0
4	A[1:0] = 2’b00
8	A[2:0] = 3’b000
16	A[3:0] = 4’b0000
32	A[4:0] = 5’b00000

- DDR controller is tuned for transfers greater than 8 bytes, so whenever possible, avoid transfers smaller than or equal to 8 bytes.
- Whenever possible, perform all the Reads to DDR controller from one initiator in a contiguous manner and avoid mixing these with Writes to eliminate bus turnaround-related inefficiencies.

  Consider the example where a block of memory is being copied from the DDR and moved to another location within DDR using the core. Assume all other units including LCD controller to be disabled. This operation can be achieved by performing one load from the source location followed by corresponding a Store to the target.

  To perform this operation in an optimum manner, first preload the block into caches (Reads), and then store (Write) this to the target.
- Avoid polling of registers if an interrupt can be generated to provide register status change.

2.11 Semaphores

The swap (SWP) and swap-byte (SWPB) instructions, as described in the ARM* V5TE architecture reference, can be used for semaphore manipulation. No internal master or process can access a memory location between the load and store portion of a SWP or SWPB to the same location.

Copyright © 2009 Marvell

2.12 Interrupts

The interrupt controller is described in detail in Chapter 9, Interrupt Controller. All internal interrupts are enabled, masked, and routed to the core FIQ or IRQ. Each peripheral-unit interrupt is enabled or disabled at the source through an interrupt-enable bit. Generally, all interrupt bits in a peripheral unit are OR-ed together and present a single value to the interrupt controller. Each interrupt goes through the Interrupt Controller Mask Register. Then the Interrupt Controller Level Register directs the interrupt into either the IRQ or FIQ. If an interrupt is taken, the Interrupt Controller Pending Register can be read to identify the source. Finally, the software services the interrupt and clears it in the source unit before exiting the service routine.

Note

There is a delay between a Write to a status bit to clear an interrupt and the actual clearing of the interrupt. Therefore, clear the interrupt early in the interrupt-service routine to allow the status bit time to clear before returning from the routine.

2.13 Reset

The processor can be reset in five ways. Table 4 summarizes the effects of each kind of reset. See the chapter on clock controllers and power management for more information.

- Power-on reset, equivalent to hardware reset, occurs at initial power-on when the power supply is detected on VCC_BBATT.
- Hardware reset results from asserting nRESET, which forces all units into reset state.
- Watchdog reset results from a timeout in the OS timer and can be used to recover control from runaway code by resetting the processor and peripherals. Watchdog reset is disabled by default and software must enable it. For more information, see the *Monahans Processor System and Timer Configuration Internal Architecture Specification*.
- GPIO reset is a soft reset, which preserves the state of some registers, real-time clocks, and the external and internal memories.
- Sleep-exit reset provides a reset to modules that have been powered down in Sleep or Deep Sleep so that they can recover properly when powered up to resume normal operation. Waking from Sleep or Deep-Sleep mode causes a Sleep-exit reset.

Each type of reset, except Sleep exit, affects the reset states of the processor pins. For details on how resets affect pin states, refer to Section 4.2 of the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification*.

The Reset Controller Status register indicates which reset has occurred.

Table 4: Effect of Each Type of Reset on Internal Register State

Unit	Sleep-exit Reset	GPIO Reset	Watchdog Reset	Hardware Reset†
Core	Reset	Reset	Reset	Reset
Memory Controller	Reset	Reset all registers except configuration registers (memory refresh maintained)	Reset	Reset

Copyright © 2009 Marvell

Table 4: Effect of Each Type of Reset on Internal Register State (Continued)

Unit	Sleep-exit Reset	GPIO Reset	Watchdog Reset	Hardware Reset†
Other internal functions	Reset	Reset	Reset	Reset
RTC	Preserved	Preserved	All reset except RTTR	Reset
Keypad Interface	All Keypad register states are reset but can act as wake-up events	Reset	Reset	Reset
MF pins	Multi function pins retain state and all pins can act as wakeups	Reset	Reset	Reset

† The power-on reset state is the same as the hardware-reset state.

2.14 Selecting Peripherals Versus General-Purpose I/O

Most peripherals connect to the external pins through alternate function multiplexers. To use such a peripheral, first configure the multiplexer so that the preferred functions are selected on the pins. All pins function as inputs by default. To allocate the peripheral function onto the pin, select the proper alternate function for the pin. Most pins have multiple alternate functions. After a function is selected for a pin, all other functions are excluded until another function is selected for the pin. For this reason, some peripherals are mapped to multiple pins, as shown in the pin function descriptions.

The GPIO function (that is, the ability to set and read the value on a pin) is a choice among the other alternate functions for a given pin.

Note

Multiple mappings does not mean multiple instances of a peripheral — only that the peripheral can be connected to the pins in several ways.

2.15 Power-On Reset and Boot Operation

Before the devices that use the processor module/services block are powered on, the external system must assert nRESET and nTRST. To allow the internal clocks to stabilize, all power supplies must be stable for a specified period before nRESET and nTRST are de-asserted (refer to Chapter 6 of the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification* for timing information). When nRESET is asserted, nRESET_OUT is asserted and can be used to reset other devices in the external system.

When the external system de-asserts nRESET, the processor de-asserts nRESET_OUT after a specified time, and the processor then attempts to boot from physical address location 0x0000_0000, which always points to the internal Boot ROM. Code in the Boot ROM is always without exception the first code that executes.

Various internal-configuration values steer the flow of operation of the Boot ROM, allowing the processor to boot from the data flash, the USB client, or a UART. Refer to the *PXA3xx Processors Boot ROM Reference Manual* for details.

In addition to the mapping at address 0x0000_0000, the Boot ROM is also mapped at address 0x5C00_0000. After the boot operation starts, the software jumps to the 0x5C00_0000 address

range and continues to execute. Mapping to address 0x0000_0000 in the Boot ROM is then disabled, allowing other memory devices to be visible at address 0x0000_0000. Mapping to address 0x0000_0000 in the Boot ROM is re-enabled by any reset operation.

2.16 Memory Map and Registers

This section provides an overview of the processor physical address map for memory and memory-mapped registers. It also summarizes the registers within the independent coprocessors in the processor core. Refer to individual unit chapters of the peripherals volume for the mapping of individual registers.

All internal control and configuration registers are mapped into physical memory space on 32-bit address boundaries. Use 32-bit word access loads and stores to access internal registers. Internal register space must be mapped as non-cacheable. In general, many buffers and FIFOs can be accessed in byte, half-word, and word sizes. Byte and half-word accesses to internal registers are not permitted and yield unpredictable results. Refer to the individual peripheral chapters for details.

Register space with no register mapping is defined as *reserved space*. Reading or writing reserved space causes unpredictable results.

The processor does not use all register bit locations. The unused bit locations are marked as reserved and are allocated for future use. Write reserved bit locations with 0b0. Ignore the values of these bits during Read operations because they are unpredictable.

The physical memory map includes external memory, internal memory, and the memory-mapped internal registers of the peripheral controllers. Services-unit registers are accessed indirectly through the peripheral applications subsystem. Thus, the peripheral-register address map includes all services-unit registers for power management, clock management, the PWR_I^{2}C interface (which controls external supply regulators), and the real-time clock. Software can use the memory management unit in the core to map portions of this physical address map to the virtual address map.

Note

Accessing reserved portions of the memory map results in a data-abort exception. Accessing reserved portions of a particular peripheral address space does not cause a data-abort exception, but the data returned is undefined.

Figure 2 shows the address bit regions used for decoding memory blocks, applications subsystem units, and applications subsystem sub-units from the physical address.

Figure 4: Physical Address Map Decode Regions

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

Block Decode (64 Blocks, 64 Mbytes each)	Unit Decode (64 Units, 1 Mbyte each) For the processor: bits 24 and 25 are always zero	Sub-Unit Decode (1 Mbyte); for the processor bits 19–11 are always zero

Copyright © 2009 Marvell

2.16.1 Application Subsystem Microarchitecture Coprocessor Register Summary

Table 5 summarizes the registers within the independent coprocessors in the processor core. These registers are accessed through coprocessor instructions. As noted in Table 5, some registers are also accessible through memory-map addressing. Refer to the *Intel XScale® Core Developers Manual* for more information on the coprocessor registers.

Table 5: Coprocessor Register Summary

CRn	CRm	Opcode1	Opcode2	Register Symbol	Register Description
Coprocessor 6 — Interrupt Controller					
0	0	0	0	ICIP†	Interrupt Controller IRQ Pending
1	0	0	0	ICMR†	Interrupt Controller Mask
2	0	0	0	ICLR†	Interrupt Controller Level
3	0	0	0	ICFR†	Interrupt Controller FIQ Pending
4	0	0	0	ICPR†	Interrupt Pending
5	0	0	0	ICHP†	Interrupt Highest Priority
Coprocessor 14 — Performance Monitoring					
0	1	0	0	PMNC	Performance Monitoring Control
1	1	0	0	CCNT	Performance Monitoring Clock Counter
4	1	0	0	INTEN	Interrupt Enable
5	1	0	0	FLAG	Overflow Flag
8	1	0	0	EVTSEL	Event Selection
0	2	0	0	PMN0	Performance Monitoring Event Counter 0
1	2	0	0	PMN1	Performance Monitoring Event Counter 1
2	2	0	0	PMN2	Performance Monitoring Event Counter 2
3	2	0	0	PMN3	Performance Monitoring Event Counter 3
Coprocessor 14 — Clock and Power Management					
6	0	0	0	CCLKCFG	Core Clock Configuration Register
7	0	0	0	PWRMODE	Power Mode
Coprocessor 14 — Software Debug					
8	0	0	0	TX	Transmit Debug Register
9	0	0	0	RX	Receive Debug
A	0	0	0	DCSR	Debug Control and Status

Table 5: Coprocessor Register Summary (Continued)

CRn	CRm	Opcode1	Opcode2	Register Symbol	Register Description
B	0	0	0	TBREG	Trace Buffer
C	0	0	0	CHKPT0	Checkpoint 0
D	0	0	0	CHKPT1	Checkpoint 1
E	0	0	0	TXRXCTRL	Transmit and Receive Control
Coprocessor 15 — XScale® Microprocessor System Control					
	ID and Cache Type Registers				
0	0	0	0	ID	Identification
0	0	0	1		L1Cache Type
0	0	1	1		L2 Cache Type
Control and Auxiliary Registers					
1	0	0	0		ARM* Control
1	0	0	1		Auxiliary Control
2	0	0	0		Translation Table Base
3	0	0	0		Domain Access Control
4	—	—	—	—	Reserved
5	0	0	0		Fault Status
6	0	0	0		Fault Address
	Cache Operations				
7	7	0	0		Invalidate I&D cache and BTB
7	5	0	0		Invalidate I cache and BTB
7	5	0	1		Invalidate I cache Line
7	6	0	0		Invalidate D cache
7	6	0	1		Invalidate D cache Line
7	10	0	1		Clean D cache Line
7	10	0	4		Drain Write (& Fill) Buffer
7	5	0	6		Invalidate Branch Target Buffer
7	2	0	5		Allocate Line in the Data Cache
8	7	0	0		Invalidate I & D TLB
8	5	0	0		Invalidate I TLB

Copyright © 2009 Marvell

Table 5: Coprocessor Register Summary (Continued)

CRn	CRm	Opcode1	Opcode2	Register Symbol	Register Description
8	5	0	1		Invalidate I TLB Entry
8	6	0	0		Invalidate D TLB
8	6	0	1		Invalidate D TLB Entry
	Cache Lock Down				
9	1	0	0		Fetch and Lock I Cache Line
	TLB Operations				
9	1	0	1		Unlock I-Cache
9	2	0	0		Read Data Cache Lock Register
9	2	0	0		Write Data Cache Lock Register
9	2	0	1		Unlock Data Cache
	TLB Lock Down				
10	4	0	0		Translate and Lock Instruction TLB Entry
10	8	0	0		Translate and Lock Data TLB Entry
10	4	0	1		Unlock Instruction TLB
10	8	0	1		Unlock Data TLB
11	—	—	—	—	Reserved
12	—	—	—	—	Reserved
13	0	0	0	PID	Processor ID
	Breakpoint Registers				
14	0	0	0	DBCR0	Data Breakpoint Register 0
14	3	0	0	DBCR1	Data Breakpoint Register 1
14	4	0	0	DBCON	Data Breakpoint Control Register
14	8	0	0	IBCR0	Instruction Breakpoint Register 0
14	9	0	0	IBCR1	Instruction Breakpoint Register 1
15	1	0	0	CPAR	Coprocessor Access
15	1	0	0	CPAR	Coprocessor Access

† These registers are also accessible through memory-map addressing.

2.16.2 Interrupt Controller Registers

The interrupt controller registers, which reside in Coprocessor 6, can be accessed in either Memory-Mapped Register Access mode or Coprocessor-Register Access mode. Coprocessor-Register Access mode significantly reduces interrupt latencies. Accesses to the interrupt controller registers in this mode must be performed in Supervisor mode.

The MRC and MRC2 coprocessor operations are treated identically and access the same registers within the coprocessor. Similarly, the MCR and MCR2 coprocessor partitions are treated identically and access the same registers within the coprocessor.

Access to interrupt control registers through the CP is limited depending whether the core is in User or Supervisor mode. An undefined instruction exception is generated if an access is made in User mode.

2.16.3 Performance Monitoring Registers

The performance-monitoring registers include four 32-bit performance counters, so that four separate events can be monitored simultaneously. In addition, a 32-bit clock counter counts the number of core clock cycles. For details, refer to Chapter 12, Performance Monitoring and Debug which defines ASSP-level monitors.

Table 6: Performance Monitoring Registers, Access: Coprocessor 14

Name	Description	CRm	CRn	Instruction
PMNC	Performance Monitor Control Register	0	1	Read: MRC p14, 0, Rd, c0, c1, 0 Write: MCR p14, 0, Rd, c0, c1, 0
CCNT	Clock Counter Register	1	1	Read: MRC p14, 0, Rd, c1, c1, 0 Write: MCR p14, 0, Rd, c1, c1, 0
INTEN	Interrupt Enable Register	4	1	Read: MRC p14, 0, Rd, c4, c1, 0 Write: MCR p14, 0, Rd, c4, c1, 0
FLAG	Overflow Flag Register	5	1	Read: MRC p14, 0, Rd, c5, c1, 0 Write: MCR p14, 0, Rd, c5, c1, 0
EVTSEL	Event Selection Register	8	1	Read: MRC p14, 0, Rd, c8, c1, 0 Write: MCR p14, 0, Rd, c8, c1, 0
PMN0	Performance Count Register 0	0	2	Read: MRC p14, 0, Rd, c0, c2, 0 Write: MCR p14, 0, Rd, c0, c2, 0
PMN1	Performance Count Register 1	1	2	Read: MRC p14, 0, Rd, c1, c2, 0 Write: MCR p14, 0, Rd, c1, c2, 0
PMN2	Performance Count Register 2	2	2	Read: MRC p14, 0, Rd, c2, c2, 0 Write: MCR p14, 0, Rd, c2, c2, 0
PMN3	Performance Count Register 3	3	2	Read: MRC p14, 0, Rd, c3, c2, 0 Write: MCR p14, 0, Rd, c3, c2, 0

2.16.4 Clock Configuration and Power Management Registers

The clock configuration and power management registers are Registers 6 and 7 in Coprocessor 14. CCLKCFG (register 6) and PWRMODE (register 7) allow software to modify the clock and power-management modes. The valid operations are described in the Clocks and Power Management chapter.

Copyright © 2009 Marvell

Ring Oscillator mode is a low-power operating mode in which the whole system runs from an integrated ring oscillator. Running from the oscillator produces a set of frequencies different from the ones derived from the core and system PLL. Because the base frequency standard is accurate only to +/– 10%, there are inaccuracies on the delivered frequencies. In the Ring Oscillator mode, only a subset of the devices can run and with some restrictions.

Table 7: Devices Operating in S0/D0CS/C0 Mode

Module	Function
Internal memory	Normal operation
DMEMC	Recalibration of delay lines is required on entry or exit
SMEMC	Operation is at low frequency
DMA	Normal operation
GCU	Normal operation
LCD	Refresh rate is controlled through the PCD register
Data Flash	Normal operation
Camera interface	Very limited function. Only limited resolutions supported
UART	Baud rates are restricted and require auto baud to get the correct baud rate
MMC/SD	Normal operation
SSP	Normal operation except for Audio mode, which requires an external clock source
AC97	Operation only using external clock source
CIR	Operates but data rates need to be adjusted
PWM	Normal operation
OST	Only 32K/1kHz/100Hz operation is supported
RTC	Normal operation
Keypad	Normal operation
I^2C	Normal operation
USIM	Normal operation

2.16.5 Coprocessor Software Debug Registers

The coprocessor software debug registers include the following coprocessors and registers:

- Coprocessor 14, registers 8–14
- Coprocessor 15, register 14

2.16.6 Coprocessor 15

The following subsections describe the registers in Coprocessor 15:

- Section 2.16.6.1, Processor ID Register
- Section 2.16.6.2, Processor Cache Type Register
- Section 2.16.6.3, Auxiliary Control Register (P-Bit)
- Section 2.16.6.4, Coprocessor Access Register (CPAR)
- Section 2.16.6.5, Additions to Coprocessor 15 Functionality

2.16.6.1 Processor ID Register

Table 8 shows the format and values presented in the Processor ID Register. This read-only register is accessible only in supervisor mode. It conforms with the values provided in the *ARM® Architecture Reference Manual*.

Table 8: Processor ID Register

Coprocessor 15 Register 0 opcode_2 = 0 | **Processor ID Register** | **Processor ID**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Vendor								Arch Version								Core G			Core R			Prod ID						Prod R			
Reset	0	1	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1	1	†	†	†	†	†	†	†	†	†	†	†	†	†

Bits	Name	Access	Description
31:24	Vendor	Read-Only	Vendor Vendor = Intel (0x69 = “i” = Intel Corporation)
23:16	Arch Version	Read-Only	Architecture Version ARM* Architecture Version 5TE = 0b0000_0101
15:13	Core G	Read-Only	Core Generation Core Generation core = 0b011
12:10	Core R	Read-Only	Core Revision Core Revision = 0b010 This field reflects revisions of core generations. Differences may include errata that dictate different operating conditions, software work-arounds, and so on.
9:4	Prod ID	Read-Only	Product ID† 0b00_0010 = PXA32x processor 0b00_1000 = PXA30x processor 0b00_1001 = PXA31x processor
3:0	Prod R	Read-Only	Processor Revision Processor stepping: 0b0001 = A1 0b0010 = A2 0b0011 = B0 0b0101 = B1 0b0110 = B2 0b0111 = C0

† These values reflect the actual product identification and revision numbers embedded in the processor.

Copyright © 2009 Marvell

Table 9: Coprocessor: New CPU ID and JTAG ID Values

Stepping	CPU ID	JTAG ID
PXA30x-A0	0x6905_6880	0x0E64_8013
PXA-30x-A1	0x6905_6881	0x1E64_8013
PXA31x-A0	0x6905_6890	0x0E64_9013
PXA31x-A1	0x6905_6891	0x1E64_9013
PXA31x-A2	0x6905_6892	0x2E64_9013
PXA31x-B0	0x6905_6893	0x3E64_9013
PXA32x-B1	0x6905_6825	0x5E64_2013
PXA32x-B2	0x6905_6826	0x6E64_2013
PXA32x-C0	0x6905_6827	0x7E64_2013

2.16.6.2 Processor Cache Type Register

For the Processor Cache Type Register, the access is as follows: Coprocessor 15, register 0, opcode_2 = 1. This register describes the cache configuration of the core.

2.16.6.3 Auxiliary Control Register (P-Bit)

For the Auxiliary Control Register, the access is as follows: Coprocessor 15, Register 1, opcode_2 = 1.

Bit 1 of this register is defined as the page table memory attribute (P-bit). It is not implemented in the processor and must be written with 0b0. Similarly, the P-bit in the memory-management unit (MMU) page table Descriptor is not implemented and must be written with 0b0.

2.16.6.4 Coprocessor Access Register (CPAR)

CPAR, defined in Table 10, controls access to all coprocessors other than CP14 and CP15. This register is accessible in supervisor mode only. Example 1 demonstrates setting the CPAR in supervisor mode.

Table 10: Processor CPAR Register

	Coprocessor 15 Register 15 opcode_2 = 0 CRm=1																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																		CP13	CP12	CP11	CP10	CP9	CP8	CP7	CP6	CP5	CP4	CP3	CP2	CP1	CP0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Access	Description
31:14	Reserved	—	Reserved
13:0	CPn	Read/Write	**Coprocessor Access Rights**: 0 = Access denied. Any attempt to access the corresponding coprocessor generates an *undefined* exception, even in supervisor mode. 1 = Access allowed, including read and write access to the coprocessor.

Example 1. Enabling Access to CP0, CP1, and CP6

```
;; The following code sets bits 0, 1, and 6 of the CPAR.
;; This enables access to Intel Wireless MMX media enhancements.
;; This enables access to Interrupt Controller Coprocessor registers.

LDR R0, =0x0043                     ; Set bits 0,1, and 6.
MCR P15, 0, R0, C15, C1, 0          ; Move to CPAR.
CPWAIT                              ; Wait for effect (Section 2.16.6.5).
```

2.16.6.5 Additions to Coprocessor 15 Functionality

Sometimes it is necessary to know exactly when a CP15 update takes effect. For example, when memory address translation is enabled (turning on the MMU), it is vital to know when the MMU is actually guaranteed to be in operation. To address this need, a processor-specific code sequence is defined for the application subsystem core. Example 2 describes this sequence, CPWAIT.

Example 2. CPWAIT: Canonical Method to Wait for CP15 Update

```
;; The following macro should be used when software needs to be
;; assured that a CP15 update has taken effect.
;; It may only be used while in a privileged mode, because it
;; accesses CP15.

    MACRO CPWAIT
    MRC P15, 0, R0, C2, C0, 0     ; arbitrary read of CP15
    MOV R0, R0                    ; wait for it
    SUB PC, PC, #4                ; branch to next instruction
                                  ; At this point, any previous CP15 writes are
                                  ; guaranteed to have taken effect.

ENDM
```


Copyright © 2009 Marvell

When multiple CP15 registers are set, it is acceptable to execute CPWAIT only once, after the sequence of MCR instructions. The CPWAIT sequence guarantees that CP15 updates complete by the time the CPWAIT completes. A CP15 side effect can occur *before* CPWAIT completes or is issued. Use the technique shown in Example 2 to ensure that this does not affect the correctness of the code.

Copyright © 2009 Marvell

3 Memory Switch

The internal switch bus (memory switch) provides a dedicated connection to/from the initiators to the completers for the data transfers. Agents that can initiate new Read or Write transfers are *initiators*. Here, the initiators are the core subsystem, System Bus 1, AXI system bus, and System Bus 2. The agents that complete the transfers are referred to as *completers*. Here, the completers are the external static-memory controller, external dynamic-memory controller, internal SRAM memory controller System Bus 1, and System Bus 2. The completer interface and initiator interface are the logic that connects the agents to the memory switch bus.The term *agent* refers to either initiators or completers.

By duplicating the data paths within the processor, the memory switch promotes very high internal data bandwidth among various controllers. The memory switch also promotes lower latencies because fewer controllers compete for a single bus.

Figure 5 depicts the memory switch bus, which separates the completer from the completer interface and the initiator from the initiator interface. The core subsystem comprises the XScale® microarchitecture (core), the Intel® Wireless MMX™ 2 coprocessor, and the internal cache.
For simplicity, these subsystem components are treated as a single unit in most cases. There are no differences between the memory switch controllers of the PXA3xx processor family.

3.1 Features

- All bus clients connect in a crossbar structure to maximize bandwidth and minimize latency
- Per agent, there is 32-bit address, 64-bit write data, and 64-bit read data
- Three initiators (core subsystem, System Bus 1, and System Bus 2)
- Completers are the external static-memory controller, external dynamic-memory controller, internal SRAM memory controller, internal flash-memory controller, System Bus 1, and System Bus 2
- Concurrent accesses to all completers from any initiator
- Programmable priority mechanism in each completer
- Atomic operations from the application subsystem core microarchitecture
- Completers execute transactions in order for each initiator
- Manages critical word first (CWF) address re-ordering

3.2 I/O Pins

The memory switch bus has no external I/O interface.

3.3 Interface Modules

The memory switch bus consists of the following interface modules:

- Application core subsystem interface: core subsystem interface between the core subsystem application core and the bus
- System Bus 1 interface between System Bus 1 and the bus
- System Bus 2 interface between System Bus 2 and the bus
- Dynamic memory controller (DMC) interface between the DMC and the bus
- Static memory controller (SMC) interface: Interface between the SMC and the bus
- Internal SRAM interface between internal SRAM controller (or on-chip ROM arrays) and the bus

Copyright © 2009 Marvell

Figure 5 shows the interconnections between the memory switch interface modules.

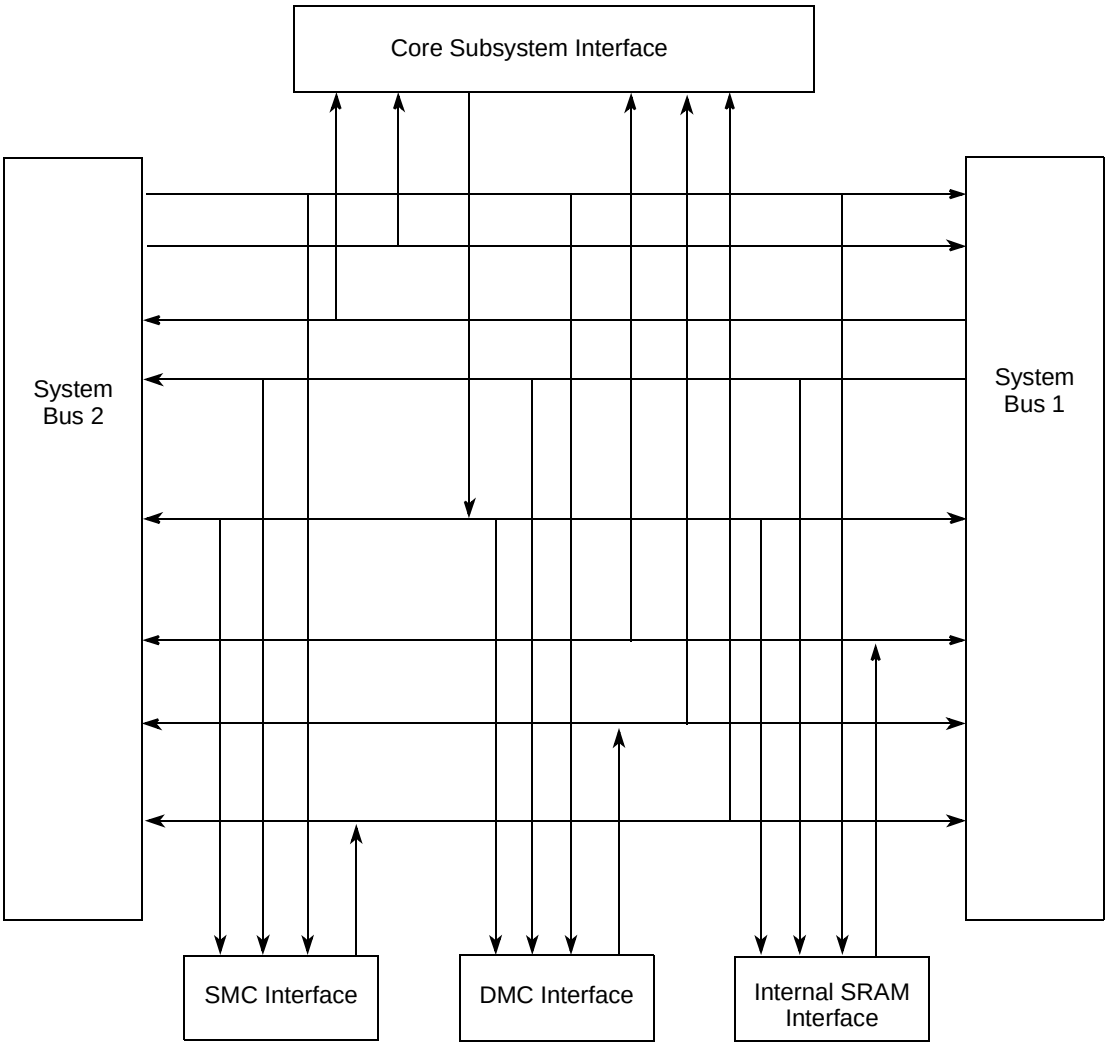
3.3.1 Priority Control

Each *completer interface* (internal SRAM, external dynamic memory controller, external static memory controller) receives transfer requests from initiators (core subsystem, System Bus 1 and System Bus 2). Within each completer interface, a transaction age-based priority algorithm assures flow control and fairness. This algorithm can reorder transfers between the three initiator sources despite the age of the transfer, but it does not reorder transfers within each initiator queue.

Note

Program the Arbiter Control register to achieve the preferred priority on System Bus 1 and System Bus 2. Refer to the ARB_CNTRL_1 and ARB_CNTRL_2 registers in the system bus arbiters chapter for details on how to prioritize transfers.

Figure 5: Memory Switch Block Diagram

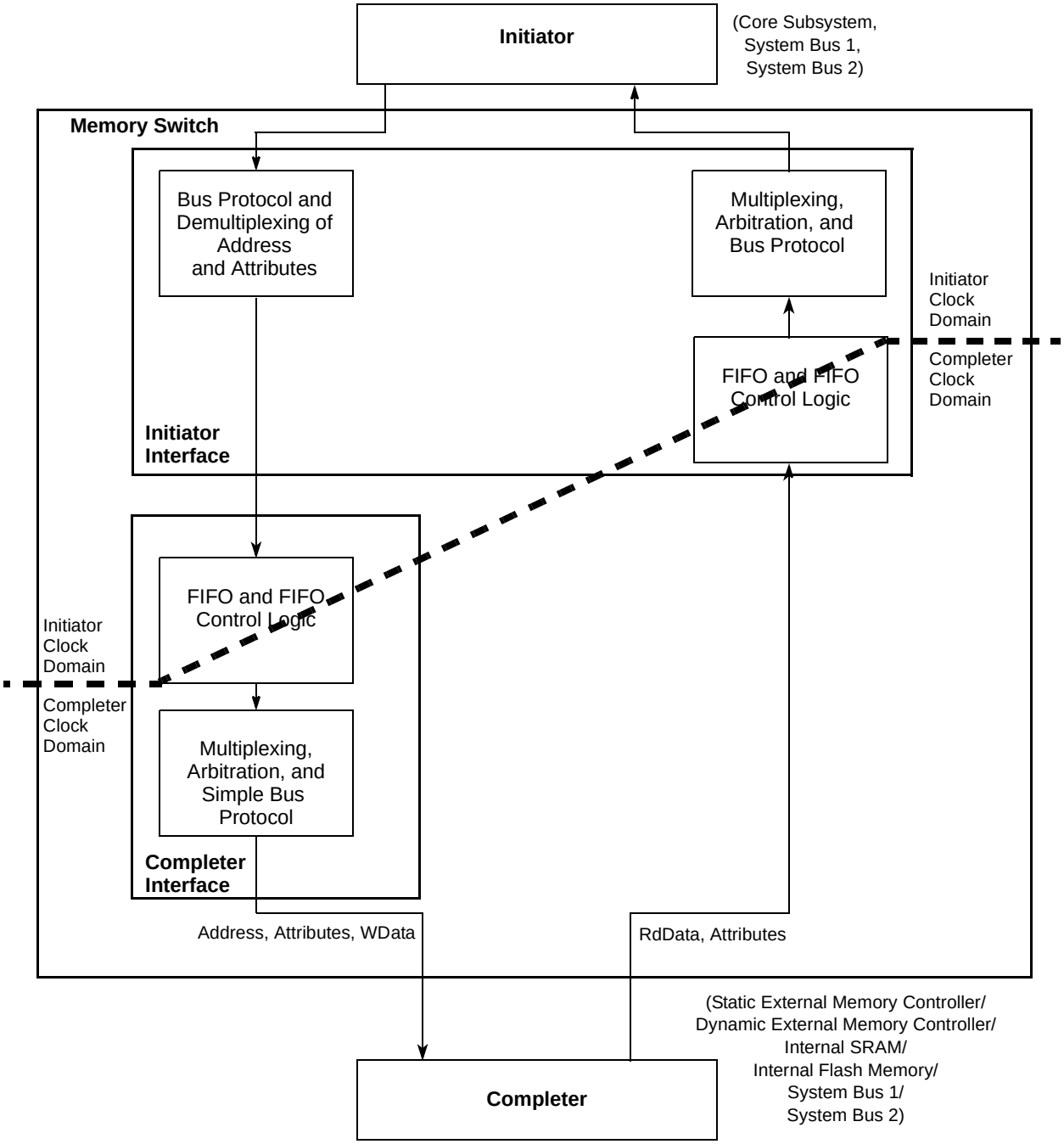


Copyright © 2009 Marvell

3.3.2 The Memory Switch Concept

The memory switch bus manages the read and write address, data, and related attributes from the initiators to the completers. Figure 6 shows only the path from one initiator to one completer, though many such paths exist within the memory switch bus. The initiators interfacing with the memory switch bus initiate new Read or Write transfers.

Figure 6: Memory Switch Concept



Copyright © 2009 Marvell

4 Pin Descriptions and Control

This chapter describes both the PXA3xx processor logical signals and their mapping to physical package pins.

4.1 Overview

The PXA3xx processor feature both single-function (dedicated) and multi-function pins. The pin-control unit manages the configuration of these multi-function pins, including the selection of alternate peripheral functions, and controls the pin state during reset and low-power modes for every pin.

The pin-control unit contains registers that allow the reset and low-power mode configuration of each multi-function pin to hold its last driven output value or instead be forced into one of five states: (1) output driven high, (2) output driven low, (3) output high impedance, (4) input pulled high, and (5) input pulled low. This register defaults to an appropriate state at reset or power up but is software configurable thereafter. The pin unit also contains registers to configure the output drive strength of individual pins when configured as outputs. All multi-function pins support configuration as a software-managed GPIO channel, which can be programmed as an output or an input that can serve as an interrupt source. Many pins can also generate wake-up events to bring the processor out of the S0/D1/C2, S0/D2/C2, S2/D3/C4 and S3/D4/C4 low-power modes.

4.2 Features

This section lists the general features of the pin-control unit.

- Controls the state of pins during reset and low-power modes
- Supports holding last-driven state or register-defined state for all outputs during reset and low-power modes
- Manages selection of GPIO and other alternate peripheral functions
- Supports software configuration of output drive strength
- Supports external interrupt generation and wakeup-event detection.

Copyright © 2009 Marvell

4.3 PXA32x Processor Pin List with Alternate Functions

Table 11: PXA32x processor Alternate Function Table

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO0	VCC_MEM	GPIO<0> (wake GENERIC[12])		DREQ					
GPIO1	VCC_MEM	GPIO<1> (wake GENERIC[12])							
GPIO2	VCC_DF	GPIO<2> (wake GENERIC[12])	RDY						
GPIO3	VCC_DF	GPIO<3> (wake GENERIC[12])	nCS<2>						
GPIO4	VCC_DF	GPIO<4> (wake GENERIC[12])	nCS<3>						
GPIO5	VCC_DF	GPIO<5>		MM2_DATA<0> (wake GENERIC[11])	nPIOR	MM1_DATA<0> (Wake GENERIC[10])		SC_IO (wake ADxER[20])	
GPIO6	VCC_DF	GPIO<6>		MM2_DATA<1> (wake GENERIC[11])	nPIOW	MM1_DATA<1> (Wake GENERIC[10])		SC_DETECT (wake ADxER[20])	
GPIO7	VCC_DF	GPIO<7>		MM2_DATA<2>/ MM2_CS0 (wake GENERIC[11])	nIOIS16	MM1_DATA<2>/ MM1_CS0 (Wake GENERIC[10])		SC_CLK	
GPIO8	VCC_DF	GPIO<8>		MM2_DATA<3>/ MM2_CS1 (wake GENERIC[11])	nPWAIT	MM1_DATA<3> (Wake GENERIC[10])		SC_nRST	
GPIO9	VCC_IO3	GPIO<9> (wake GENERIC[12])	SC_VS0	L_CS					
GPIO10	VCC_IO4	GPIO<10>	UTM_CLK (wake ADxER[26])						
GPIO11	VCC_IO3	GPIO<11>	PWM<0>					EXT_SYNC0 (wake GENERIC[13])	
GPIO12	VCC_IO3	GPIO<12>	PWM<1>					EXT_SYNC1 (wake GENERIC[13])	
GPIO13	VCC_IO3	GPIO<13>	PWM<2>		KP_MKOUT<4>			CHOUT0	
GPIO14	VCC_IO3	GPIO<14>	PWM<3>		KP_MKOUT<5>	HZ_CLK	ONE_WIRE	CHOUT1	
GPIO15	VCC_IO3	GPIO<15>	U_VS0					KP_DKIN<6> (wake ADxER[21])	
GPIO16	VCC_IO3	GPIO<16>	U_nVS1					KP_DKIN<5> (wake ADxER[21])	

Copyright © 2009 Marvell

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO17	VCC_IO3	GPIO<17>	U_nVS2					KP_DKIN<4> (wake ADxER[21])	
GPIO18	VCC_CARD1	GPIO<18>	U_IO (wake ADxER[19])			MM1_DATA<0> (Wake GENERIC[10])			
GPIO19	VCC_CARD1	GPIO<19>	U_DETECT (wake ADxER[19])			MM1_DATA<1> (Wake GENERIC[10])			
GPIO20	VCC_CARD1	GPIO<20>	U_CLK			MM1_DATA<2> (Wake GENERIC[10])			
GPIO21	VCC_CARD1	GPIO<21>	U_nRST			MM1_DATA<3> (Wake GENERIC[10])			
GPIO22	VCC_CARD1	GPIO<22>				MM1_CLK			
GPIO23	VCC_CARD1	GPIO<23>				MM1_CMD (Wake GENERIC[10])			
GPIO24	VCC_CARD2	GPIO<24>	SC_IO (wake ADxER[20])			MM2_DATA<0> (wake GENERIC[11])			
GPIO25	VCC_CARD2	GPIO<25>	SC_DETECT (wake ADxER[20])			MM2_DATA<1> (wake GENERIC[11])			
GPIO26	VCC_CARD2	GPIO<26>	SC_CLK			MM2_DATA<2> (wake GENERIC[11])			
GPIO27	VCC_CARD2	GPIO<27>	SC_nRST			MM2_DATA<3> (wake GENERIC[11])			
GPIO28	VCC_CARD2	GPIO<28>				MM2_CLK			
GPIO29	VCC_CARD2	GPIO<29>				MM2_CMD (Wake GENERIC[11])			
GPIO30	VCC_IO4	GPIO<30>		UART3_RXD (wake GENERIC[5])	MM2_CLK	MM1_CLK	UTM_RXVALID (Wake ADxER[26])	UART3_TXD	
GPIO31	VCC_IO4	GPIO<31>		UART3_TXD	MM2_CMD (Wake GENERIC[11])	MM1_CMD (Wake GENERIC[10])	CIR_OUT	UART3_RXD (wake GENERIC[5])	KP_DKIN<3> (wake ADxER[21])

Copyright © 2009 Marvell

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO32	VCC_IO4	GPIO<32>	SCL (wake GENERIC[13])						
GPIO33	VCC_IO4	GPIO<33>	SDA (wake GENERIC[13])						
GPIO34	VCC_IO4	GPIO<34>	AC97_SYSCLK		UTM_RXVALID (Wake ADxER[26])				
GPIO35	VCC_IO4	GPIO<35>	AC97_SDATA_IN_0 (Wake GENERIC[0])	SSPSCLK2 (wake GENERIC[2])	UTM_RXACTIVE (Wake ADxER[26])				
GPIO36	VCC_IO4	GPIO<36>	AC97_SDATA_IN_1 (Wake GENERIC[0])	SSPSFRM2 (wake GENERIC[2])	UTM_RXERROR				
GPIO37	VCC_IO4	GPIO<37>	AC97_SDATA_OUT	SSPTXD2		UTM_OPMODE0	SSPRXD2 (wake GENERIC[2])		
GPIO38	VCC_IO4	GPIO<38>	AC97_SYNC	SSPRXD2 (wake GENERIC[2])		UTM_OPMODE1	SSPTXD2		
GPIO39	VCC_IO4	GPIO<39>	AC97_BITCLK (Wake GENERIC[0])	SSPEXTCLK2/ SSPCLK2EN (wake GENERIC[2])		UTM_TXVALID			
GPIO40	VCC_IO4	GPIO<40>	AC97_nACRESET	SSPSYSCLK2					
GPIO41	VCC_IO4	GPIO<41>		UART1_RXD (wake GENERIC[3])	UTM_DATA<0>	UART1_TXD			
GPIO42	VCC_IO4	GPIO<42>		UART1_TXD	UTM_DATA<1>	UART1_RXD (wake GENERIC[3])			
GPIO43	VCC_IO4	GPIO<43>		UART1_CTS (wake GENERIC[3])	UTM_DATA<2>	UART1_RTS			
GPIO44	VCC_IO4	GPIO<44>		UART1_DCD (wake GENERIC[3])	UTM_DATA<3>				
GPIO45	VCC_IO4	GPIO<45>		UART1_DSR (wake GENERIC[3])	UTM_DATA<4>	UART1_DTR			
GPIO46	VCC_IO4	GPIO<46>		UART1_RI (wake GENERIC[3])	UTM_DATA<5>				
GPIO47	VCC_IO4	GPIO<47>		UART1_DTR	UTM_DATA<6>	UART1_DSR (wake GENERIC[3])			
GPIO48	VCC_IO4	GPIO<48>		UART1_RTS	UTM_DATA<7>	UART1_CTS (wake GENERIC[3])			
GPIO49	VCC_CI	GPIO<49>	CIF_DD<0>		UTM_DATA<0>				
GPIO50	VCC_CI	GPIO<50>	CIF_DD<1>		UTM_DATA<1>				

Copyright © 2009 Marvell

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO51	VCC_CI	GPIO<51>	CIF_DD<2>		UTM_DATA<2>				
GPIO52	VCC_CI	GPIO<52>	CIF_DD<3>		UTM_DATA<3>				
GPIO53	VCC_CI	GPIO<53>	CIF_DD<4>		UTM_DATA<4>				
GPIO54	VCC_CI	GPIO<54>	CIF_DD<5>		UTM_DATA<5>				
GPIO55	VCC_CI	GPIO<55>	CIF_DD<6>		UTM_DATA<6>				
GPIO56	VCC_CI	CIF_DD<7>			UTM_DATA<7>				
GPIO57	VCC_CI	GPIO<57>	CIF_DD<8>						
GPIO58	VCC_CI	GPIO<58>	CIF_DD<9>	UTM_RXVALID (Wake ADxER[26])					
GPIO59	VCC_CI	CIF_MCLK	UTM_RXACTIVE (Wake ADxER[26])	CLK_48M					
GPIO60	VCC_CI	CIF_PCLK	UTM_RXERROR		UTM_LINESTATE<1> (Wake ADxER[26])				
GPIO61	VCC_CI	CIF_HSYNC		UTM_OPMODE0					
GPIO62	VCC_CI	CIF_VSYNC		UTM_OPMODE1					
GPIO63	VCC_LCD	GPIO<63>	L_DD<8>	L_CS					
GPIO64	VCC_LCD	GPIO<64>	L_DD<9>	L_VSYNC					
GPIO65	VCC_LCD	GPIO<65>	L_DD<10>				UTM_XCVR_SELECT		
GPIO66	VCC_LCD	GPIO<66>	L_DD<11>				UTM_TERM_SELECT		
GPIO67	VCC_LCD	GPIO<67>	L_DD<12>		UTM_SUSPENDM_X				
GPIO68	VCC_LCD	GPIO<68>	L_DD<13>		UTM_LINESTATE<0> (Wake ADxER[26])				
GPIO69	VCC_LCD	GPIO<69>	L_DD<14>	SSPSCLK3 (wake GENERIC[7])					
GPIO70	VCC_LCD	GPIO<70>	L_DD<15>	SSPSFRM3 (wake GENERIC[7])			UTM_TXVALID		
GPIO71	VCC_LCD	GPIO<71>	L_DD<16>	SSPTXD3			SSPRXD3 (wake GENERIC[7])		
GPIO72	VCC_LCD	GPIO<72>	L_DD<17>	SSPRXD3 (wake GENERIC[7])			SSPTXD3		
GPIO73	VCC_LCD	GPIO<73>	UTM_TXREADY						
GPIO74	VCC_LCD	GPIO<74>	UTM_RESET	L_VSYNC					

Copyright © 2009 Marvell

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO75	VCC_MSL	GPIO<75>	UART1_RXD (wake GENERIC[3])	USB_P3_1 (wake ADxER[23])		MM2_DATA<0> (wake GENERIC[11])			
GPIO76	VCC_MSL	GPIO<76>	UART1_TXD	USB_P3_2	UART1_RXD (wake GENERIC[3])	MM2_DATA<1> (wake GENERIC[11])			
GPIO77	VCC_MSL	GPIO<77>	UART1_CTS (wake GENERIC[3])	USB_P3_3 (wake ADxER[23])		MM2_DATA<2>/ MM2_CS0 (wake GENERIC[11])			
GPIO78	VCC_MSL	GPIO<78>	UART1_DCD (wake GENERIC[3])	USB_P3_4 (wake ADxER[23])		MM2_DATA<3>/ MM2_CS1 (wake GENERIC[11])			
GPIO79	VCC_MSL	GPIO<79>	UART1_DSR (wake GENERIC[3])	USB_P3_5 (wake ADxER[23])		MM2_CLK			
GPIO80	VCC_MSL	GPIO<80>	UART1_RI (wake GENERIC[3])	USB_P3_6 (wake ADxER[23])		MM2_CMD (Wake GENERIC[11])			
GPIO81	VCC_MSL	GPIO<81>	UART1_DTR		UART1_DSR (wake GENERIC[3])		KP_DKIN<2> (wake ADxER[21])		
GPIO82	VCC_MSL	GPIO<82>	UART1_RTS		UART1_CTS (wake GENERIC[3])		KP_DKIN<1> (wake ADxER[21])		
GPIO83	VCC_IO6	GPIO<83>	SSPSCLK (wake GENERIC[1])	KP_MKOUT<0>	KP_DKIN<0> (wake ADxER[21])			L_DD<18>	UTM_TXVALID
GPIO84	VCC_IO6	GPIO<84>	SSPSFRM (wake GENERIC[1])	KP_MKOUT<1>	KP_DKIN<1> (wake ADxER[21])				
GPIO85	VCC_IO6	GPIO<85>	SSPTXD	KP_MKOUT<2>	KP_DKIN<2> (wake ADxER[21])		UTM_RXVALID (Wake ADxER[26])	SSPRXD (wake GENERIC[1])	
GPIO86	VCC_IO6	GPIO<86>	SSPRXD (wake GENERIC[1])	KP_MKOUT<3>	KP_DKIN<3> (wake ADxER[21])		UTM_RXACTIVE (Wake ADxER[26])		
GPIO87	VCC_IO6	GPIO<87>	SSPEXTCLK/SSPCLKEN (wake GENERIC[1])	SC_nVS1			UTM_RXERROR		
GPIO88	VCC_IO6	GPIO<88>	SSPSYSCLK	SC_nVS2				L_DD<19>	UTM_OPMODE0
GPIO89	VCC_IO6	GPIO<89>	SSPSCLK3 (wake GENERIC[7])	UART3_CTS (wake GENERIC[5])		UART3_RTS			

Copyright © 2009 Marvell

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO90	VCC_IO6	GPIO<90>	SSPSFRM3 (wake GENERIC[7])	UART3_RTS	UTM_LINESTATE<0> (Wake ADxER[26])	UART3_CTS (wake GENERIC[5])			
GPIO91	VCC_IO6	GPIO<91>	SSPTXD3	UART3_TXD	UTM_LINESTATE<1> (Wake ADxER[26])	UART3_RXD (wake GENERIC[5])	SSPRXD3 (wake GENERIC[7])		
GPIO92	VCC_IO6	GPIO<92>	SSPRXD3 (wake GENERIC[7])	UART3_RXD (wake GENERIC[5])		UART3_TXD	SSPTXD3		
GPIO93	VCC_IO6	GPIO<93>	SSPSCLK4 (wake GENERIC[8])	UTM_RESET					
GPIO94	VCC_IO6	GPIO<94>	SSPSFRM4 (wake GENERIC[8])		UTM_XCVR_SELECT				
GPIO95	VCC_IO6	GPIO<95>	SSPTXD4		UTM_TERM_SELECT		SSPRXD4 (wake GENERIC[8])		
GPIO96	VCC_IO6	GPIO<96>	SSPRXD4 (wake GENERIC[8])	UTM_SUSPENDM_X			SSPTXD4		
GPIO97	VCC_IO6	GPIO<97>	UART1_RXD (wake GENERIC[3])	USB_P2_2 (wake ADxER[22])	USB_P2_5 (wake ADxER[22])	USB_P2_6		UART1_TXD	
GPIO98	VCC_IO6	GPIO<98>	UART1_TXD	USB_P2_6	UTM_RESET	USB_P2_2 (wake ADxER[22])	USB_P2_5 (wake ADxER[22])	UART1_RXD (wake GENERIC[3])	
GPIO99	VCC_IO6	GPIO<99>	UART1_CTS (wake GENERIC[3])	USB_P2_1 (wake ADxER[22])			UTM_XCVR_SELECT	UART1_RTS	
GPIO100	VCC_IO6	GPIO<100>	UART1_DCD (wake GENERIC[3])	USB_P2_4 (wake ADxER[22])		UART1_TXD	UTM_TERM_SELECT	UART1_RXD (wake GENERIC[3])	
GPIO101	VCC_IO6	GPIO<101>	UART1_DSR (wake GENERIC[3])	USB_P2_8	UTM_SUSPENDM_X			UART1_DTR	
GPIO102	VCC_IO6	GPIO<102>	UART1_RI (wake GENERIC[3])	USB_P2_3 (wake ADxER[22])	UTM_LINESTATE<0> (Wake ADxER[26])	UART1_RXD (wake GENERIC[3])			
GPIO103	VCC_IO6	GPIO<103>	UART1_DTR	USB_P2_5 (wake ADxER[22])	UTM_LINESTATE<1> (Wake ADxER[26])			UART1_DSR (wake GENERIC[3])	
GPIO104	VCC_IO6	GPIO<104>	UART1_RTS	USB_P2_7 (wake ADxER[22]		UTM_OPMODE1		UART1_CTS (wake GENERIC[3])	
GPIO105	VCC_IO6	GPIO<105>	UART3_CTS (wake GENERIC[5])	KP_DKIN<0> (wake ADxER[21])	UART3_RTS			L_DD<20>	
GPIO106	VCC_IO6	GPIO<106>	UART3_RTS	KP_DKIN<1> (wake ADxER[21])	UART3_CTS (wake GENERIC[5])				L_DD<21>

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO107	VCC_IO6	GPIO<107>	UART3_TXD	KP_DKIN<2> (wake ADxER[21])	UART3_RXD (wake GENERIC[5])	UTM_LINESTATE<0> (Wake ADxER[26])		L_DD<22>	
GPIO108	VCC_IO6	GPIO<108>	UART3_RXD (wake GENERIC[5])	KP_DKIN<3> (wake ADxER[21])	UART3_TXD		UTM_OPMODE 1	L_DD<23>	
GPIO109	VCC_IO6	GPIO<109>	UART2_RTS	KP_DKIN<4> (wake ADxER[21])	UART2_CTS (wake GENERIC[4])				
GPIO110	VCC_IO6	GPIO<110>	UART2_RXD (wake GENERIC[4])	KP_DKIN<5> (wake ADxER[21])	UART2_TXD				
GPIO111	VCC_IO6	GPIO<111>	UART2_TXD	KP_DKIN<6> (wake ADxER[21])	UART2_RXD (wake GENERIC[4])				
GPIO112	VCC_IO6	GPIO<112>	UART2_CTS (wake GENERIC[4])	KP_DKIN<7> (wake ADxER[21])	UART2_RTS				
GPIO113	VCC_IO1	GPIO<113>	KP_MKIN<0> (wake GENERIC[6])	KP_DKIN<0> (wake ADxER[21])					
GPIO114	VCC_IO1	GPIO<114>	KP_MKIN<1> (wake GENERIC[6])	KP_DKIN<1> (wake ADxER[21])					
GPIO115	VCC_IO1	GPIO<115>	KP_MKIN<2> (wake GENERIC[6])	KP_DKIN<2> (wake ADxER[21])					
GPIO116	VCC_IO1	GPIO<116>	KP_MKIN<3> (wake GENERIC[6])	KP_DKIN<3> (wake ADxER[21])					
GPIO117	VCC_IO1	GPIO<117>	KP_MKIN<4> (wake GENERIC[6])	KP_DKIN<4> (wake ADxER[21])					
GPIO118	VCC_IO1	GPIO<118>	KP_MKIN<5> (wake GENERIC[6])	KP_DKIN<5> (wake ADxER[21])					
GPIO119	VCC_IO1	GPIO<119>	KP_MKIN<6> (wake GENERIC[6])	KP_DKIN<6> (wake ADxER[21])					
GPIO120	VCC_IO1	GPIO<120>	KP_MKIN<7> (wake GENERIC[6])	KP_DKIN<7> (wake ADxER[21])					
GPIO121	VCC_IO1	GPIO<121>	KP_MKOUT<0>	KP_DKIN<6> (wake ADxER[21])					
GPIO122	VCC_IO1	GPIO<122>	KP_MKOUT<1>	KP_DKIN<5> (wake ADxER[21])					
GPIO123	VCC_IO1	GPIO<123>	KP_MKOUT<2>	KP_DKIN<4> (wake ADxER[21])					
GPIO124	VCC_IO1	GPIO<124>	KP_MKOUT<3>	KP_DKIN<3> (wake ADxER[21])					

Copyright © 2009 Marvell

Table 11: PXA32x processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO125	VCC_IO1	GPIO<125>	KP_MKOUT<4>	KP_DKIN<2> (wake ADxER[21])					
GPIO126	VCC_IO1	GPIO<126>	KP_MKOUT<5>	KP_DKIN<1> (wake ADxER[21])					
GPIO127	VCC_IO1	GPIO<127>	KP_MKOUT<6>	KP_DKIN<0> (wake ADxER[21])					
GPIO0_2	VCC_IO1	GPIO<0>	HZ_CLK	ONE_WIRE	CLK_EXT (wake GENERIC[13])				
GPIO1_2	VCC_IO1	GPIO<1>					KP_DKIN<0> (wake ADxER[21])		
GPIO2_2	VCC_IO1	GPIO<2>	USBHPEN	KP_DKIN<0> (wake ADxER[21])					
GPIO3_2	VCC_IO1	GPIO<3>	USBHPWR	KP_DKIN<1> (wake ADxER[21])					
GPIO4_2	VCC_IO1	GPIO<4>	KP_DKIN<7> (wake ADxER[21])						
GPIO5_2	VCC_IO1	GPIO<5>	KP_MKOUT<7>						
GPIO6_2	VCC_LCD	GPIO<6>	L_DD<0>						
GPIO7_2	VCC_LCD	GPIO<7>	L_DD<1>						
GPIO8_2	VCC_LCD	GPIO<8>	L_DD<2>						
GPIO9_2	VCC_LCD	GPIO<9>	L_DD<3>						
GPIO10_2	VCC_LCD	GPIO<10>	L_DD<4>						
GPIO11_2	VCC_LCD	GPIO<11>	L_DD<5>						
GPIO12_2	VCC_LCD	GPIO<12>	L_DD<6>						
GPIO13_2	VCC_LCD	GPIO<13>	L_DD<7>						
GPIO14_2	VCC_LCD	GPIO<14>	L_FCLK_RD						
GPIO15_2	VCC_LCD	GPIO<15>	L_LCLK_A0						
GPIO16_2	VCC_LCD	GPIO<16>	L_PCLK_WR						
GPIO17_2	VCC_LCD	GPIO<17>	L_BIAS						

Copyright © 2009 Marvell

4.4 PXA31x Processor Pin List with Alternate Functions

Table 12: PXA31x Processor Alternate Function Table

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO0	VCC_DF	GPIO_0 (wake GENERIC[12])	RDY					DREQ1	
GPIO1	VCC_DF	GPIO_1 (wake GENERIC[12])	nCS<2>					DREQ2	
GPIO2	VCC_DF	GPIO_2 (wake GENERIC[12])	nCS<3>	nXCVREN				DREQ3	
GPIO3	VCC_CARD1	GPIO_3	U_IO (wake ADxER[19])	KP_DKIN<6> (wake ADxER[21])		MM1_DATA<0> (Wake GENERIC[10])			
GPIO4	VCC_CARD1	GPIO_4	U_DETECT (wake ADxER[19])	KP_DKIN<7> (wake ADxER[21])		MM1_DATA<1> (Wake GENERIC[10])			
GPIO5	VCC_CARD1	GPIO_5	U_CLK	KP_MKIN<0> (wake GENERIC[6])		MM1_DATA<2> (Wake GENERIC[10])			
GPIO6	VCC_CARD1	GPIO_6	U_nRST	KP_MKIN<1> (wake GENERIC[6])		MM1_DATA<3> (Wake GENERIC[10])			
GPIO7	VCC_CARD1	GPIO_7	KP_MKOUT<5>	UART3_RXD (wake GENERIC[5])		MM1_CLK		UART3_TXD	
GPIO8	VCC_CARD1	GPIO_8		UART3_TXD		MM1_CMD (Wake GENERIC[10])	CIR_OUT	UART3_RXD (wake GENERIC[5])	
GPIO9	VCC_CARD2	GPIO_9	SC_IO (wake ADxER[20])		KP_MKIN<6> (wake GENERIC[6])	MM2_DATA<0> (wake GENERIC[11])			
GPIO10	VCC_CARD2	GPIO_10	SC_DETECT (wake ADxER[20])		KP_MKIN<7> (wake GENERIC[6])	MM2_DATA<1> (wake GENERIC[11])			
GPIO11	VCC_CARD2	GPIO_11	SC_CLK		KP_MKOUT<5>	MM2_DATA<2> (wake GENERIC[11])			
GPIO12	VCC_CARD2	GPIO_12	SC_nRST		KP_MKOUT<6>	MM2_DATA<3> (wake GENERIC[11])			
GPIO13	VCC_CARD2	GPIO_13			KP_MKOUT<7>	MM2_CLK			
GPIO14	VCC_CARD2	GPIO_14				MM2_CMD (Wake GENERIC[11])	MM1_CMD (Wake GENERIC[10])		
GPIO15	VCC_CARD2	GPIO_15	SC_VS0	L_CS	UART2_CTS (wake GENERIC[4])	UART2_RTS	MM1_CMD (Wake GENERIC[10])	SSPSCLK (wake GENERIC[1])	
GPIO16	VCC_CARD2	GPIO_16	U_VS0	SSPSFRM (wake GENERIC[1])	CIR_OUT	UART2_RTS	UART2_CTS (wake GENERIC[4])	KP_DKIN<6> (wake ADxER[21])	
GPIO17	VCC_IO3	GPIO_17	PWM<0>	SSPSFRM2 (wake GENERIC[2])				EXT_SYNC0 (wake GENERIC[12])	
GPIO18	VCC_IO3	GPIO_18	PWM<1>	SSPRXD (wake GENERIC[1])	AC97_SDATA_IN_3 (wake GENERIC[0])	UART2_TXD	UART2_RXD (wake GENERIC[4])	EXT_SYNC1 (wake GENERIC[12])	SSPTXD

Copyright © 2009 Marvell

Table 12: PXA31x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO19	VCC_IO3	GPIO_19	PWM<2>	SSPTXD2	KP_MKOUT<4>	UART2_RXD (wake GENERIC[4])	UART2_TXD	CHOUT0	SSPRXD2 (wake GENERIC[2])
GPIO20	VCC_IO3	GPIO_20	PWM<3>	SSPTXD	KP_MKOUT<5>	HZ_CLK	ONE_WIRE	CHOUT1	SSPRXD (wake GENERIC[1])
GPIO21	VCC_IO3	GPIO_21	SCL (wake GENERIC[9])						
GPIO22	VCC_IO3	GPIO_22	SDA (wake GENERIC[9])	AC97_SDATA_IN_3 (wake GENERIC[0])					
GPIO23	VCC_IO3	GPIO_23	AC97_nACRESET	SSPSYSCLK2	MMC_CLK				
GPIO24	VCC_IO3	GPIO_24	AC97_SYSCLK		MMC1_CMD (Wake ADxER[26])	SSPRXD2 (wake GENERIC[2])	SSPTXD2		
GPIO25	VCC_IO3	GPIO_25	AC97_SDATA_IN_0 (Wake GENERIC[0])	SSPSCLK2 (wake GENERIC[2])					
GPIO26	VCC_IO3	GPIO_26	AC97_SDATA_IN_1 (wake GENERIC[0])	SSPSFRM2 (wake GENERIC[2])					
GPIO27	VCC_IO3	GPIO_27	AC97_SDATA_OUT	SSPTXD2			SSPRXD2 (wake GENERIC[2])		
GPIO28	VCC_IO3	GPIO_28	AC97_SYNC	SSPRXD2 (wake GENERIC[2])			SSPTXD2		
GPIO29	VCC_IO3	GPIO_29	AC97_BITCLK (Wake GENERIC[0])	SSPEXTCLK2/ SSPCLK2EN (wake GENERIC[2])	MM1_DATA<0> (Wake GENERIC[10])				
GPIO30	VCC_ULPI	GPIO_30	USB_P2_2	UART1_RXD (wake GENERIC[3])	ULPI_DATA<0> (wake ADxER[26])	UART1_TXD			
GPIO31	VCC_ULPI	GPIO_31	USB_P2_6 (wake ADxER[22])	UART1_TXD	ULPI_DATA<1> (wake ADxER[26])	UART1_RXD (wake GENERIC[3])			
GPIO32	VCC_ULPI	GPIO_32	USB_P2_4 (wake ADxER[22])	UART1_CTS (wake GENERIC[3])	ULPI_DATA<2>	UART1_RTS			
GPIO33	VCC_ULPI	GPIO_33	ULPI_OTG_INTR (wake ADxER[22])	UART1_DCD (wake GENERIC[3])	ULPI_DATA<3> (wake ADxER[26])	ULPI_OTG_INTR (wake GENERIC[3])	SSPSCLK (wake GENERIC[1])	SSPSCLK2 (wake GENERIC[2])	
GPIO34	VCC_ULPI	GPIO_34	USB_P2_5 (wake ADxER[22])	UART1_DSR (wake GENERIC[3])	ULPI_DATA<4>	UART1_DTR	SSPSFRM (wake GENERIC[1])	SSPSFRM2 (wake GENERIC[2])	
GPIO35	VCC_ULPI	GPIO_35	USB_P2_3 (wake ADxER[22])	UART1_RI (wake GENERIC[3])	ULPI_DATA<5>	SSPRXD (wake GENERIC[1])	SSPTXD	SSPRXD2 (wake GENERIC[2])	SSPTXD2
GPIO36	VCC_ULPI	GPIO_36	USB_P2_1 (wake ADxER[22])	UART1_DTR	ULPI_DATA<6>	UART1_DSR (wake GENERIC[3])	SSPTXD	SSPRXD (wake GENERIC[1])	SSPTXD2
GPIO37	VCC_ULPI	GPIO_37		UART1_RTS	ULPI_DATA<7>	UART1_CTS (wake GENERIC[3])			

Copyright © 2009 Marvell

Table 12: PXA31x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO38	VCC_ULPI	GPIO_38	UPLI_CLK (wake ADxER[26])				KP_MKOUT<5>		
GPIO39	VCC_CI	GPIO_39	CIF_DD<0>						
GPIO40	VCC_CI	GPIO_40	CIF_DD<1>						
GPIO41	VCC_CI	GPIO_41	CIF_DD<2>						
GPIO42	VCC_CI	GPIO_42	CIF_DD<3>						
GPIO43	VCC_CI	GPIO_43	CIF_DD<4>						
GPIO44	VCC_CI	GPIO_44	CIF_DD<5>						
GPIO45	VCC_CI	GPIO_45	CIF_DD<6>						
GPIO46	VCC_CI	CIF_DD<7>	GPIO<46>						
GPIO47	VCC_CI	GPIO_47	CIF_DD<8>						
GPIO48	VCC_CI	GPIO_48	CIF_DD<9>						
GPIO49	VCC_CI	CIF_MCLK		CLK_48M	GPIO<49>				
GPIO50	VCC_CI	CIF_PCLK		GPIO<50>					
GPIO51	VCC_CI	CIF_HSYNC			GPIO<51>				
GPIO52	VCC_CI	CIF_VSYNC			GPIO<52>				
GPIO53	VCC_LCD	GPIO_53 (wake GENERIC[12])					KP_MKOUT6		L_DD<18>
GPIO54	VCC_LCD	GPIO_54	L_DD<0>			SMEM_FADDR 4			ML_DD<0>
GPIO55	VCC_LCD	GPIO_55	L_DD<1>			SMEM_FADDR 5			ML_DD<1>
GPIO56	VCC_LCD	GPIO_56	L_DD<2>			SMEM_FADDR 6			ML_DD<2>
GPIO57	VCC_LCD	GPIO_57	L_DD<3>			SMEM_FADDR 7			ML_DD<3>
GPIO58	VCC_LCD	GPIO_58	L_DD<4>			SMEM_FADDR 8			ML_DD<4>
GPIO59	VCC_LCD	GPIO_59	L_DD<5>			SMEM_FADDR 9			ML_DD<5>
GPIO60	VCC_LCD	GPIO_60	L_DD<6>			SMEM_FADDR 10			ML_DD<6>
GPIO61	VCC_LCD	GPIO_61	L_DD<7>			SMEM_FADDR 11			ML_DD<7>
GPIO62	VCC_LCD	GPIO_62	L_DD<8>	L_CS		SMEM_FADDR 12			ML_DD<8>
GPIO63	VCC_LCD	GPIO_63	L_DD<9>	L_VSYNC		SMEM_FADDR 13			ML_DD<9>
GPIO64	VCC_LCD	GPIO_64	L_DD<10>	SSPSCLK2 (wake GENERIC[2])		SMEM_FADDR 14			ML_DD<10>
GPIO65	VCC_LCD	GPIO_65	L_DD<11>	SSPSFRM2 (wake GENERIC[2])		SMEM_FADDR 15			ML_DD<11>

Copyright © 2009 Marvell

Table 12: PXA31x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO66	VCC_LCD	GPIO_66	L_DD<12>	SSPRXD2 (wake GENERIC[2])		SSPTXD2	SMEM_FADDR 16		ML_DD<12>
GPIO67	VCC_LCD	GPIO_67	L_DD<13>	SSPTXD2		SSPRXD2 (wake GENERIC[2])	SMEM_FADDR 17		ML_DD<13>
GPIO68	VCC_LCD	GPIO_68	L_DD<14>	SSPSCLK3 (wake GENERIC[7])		SMEM_FADDR 18			ML_DD<14>
GPIO69	VCC_LCD	GPIO_69	L_DD<15>	SSPSFRM3 (wake GENERIC[7])		SMEM_FADDR 19			ML_DD<15>
GPIO70	VCC_LCD	GPIO_70	L_DD<16>	SSPTXD3	KP_MKIN<6> (wake GENERIC[6])	SMEM_FADDR 20	SSPRXD3 (wake GENERIC[7])		
GPIO71	VCC_LCD	GPIO_71	L_DD<17>	SSPRXD3 (wake GENERIC[7])	KP_MKIN<7> (wake GENERIC[6])	SMEM_FADDR 21	SSPTXD3		
GPIO72	VCC_LCD	GPIO_72	L_FCLK_RD			SMEM_FADDR 22			ML_FCLK
GPIO73	VCC_LCD	GPIO_73	L_LCLK_A0			SMEM_FADDR 23			ML_LCLK
GPIO74	VCC_LCD	GPIO_74	L_PCLK_WR			SMEM_FADDR 24			ML_PCLK
GPIO75	VCC_LCD	GPIO_75	L_BIAS			SMEM_FADDR 25			ML_BIAS
GPIO76	VCC_LCD	GPIO_76 (wake GENERIC[12])		L_VSYNC					
GPIO77	VCC_MSL	GPIO_77	UART1_RXD (wake GENERIC[3])	USB_P3_1 (wake ADxER[23])	UART1_TXD	MM2_DATA<0> (wake GENERIC[11])			UART3_RXD (wake GENERIC[5])
GPIO78	VCC_MSL	GPIO_78	UART1_TXD	USB_P3_2 (wake ADxER[23])	UART1_RXD (wake GENERIC[3])	MM2_DATA<1> (wake GENERIC[11])	KP_MKOUT<7>		KP_MKOUT<6>
GPIO79	VCC_MSL	GPIO_79	UART1_CTS (wake GENERIC[3])	USB_P3_3 (wake ADxER[23])	UART1_RTS	MM2_DATA<2> /MM2_CS0 (wake GENERIC[11])			
GPIO80	VCC_MSL	GPIO_80	UART1_DCD (wake GENERIC[3])	USB_P3_4 (wake ADxER[23])		MM2_DATA<3> /MM2_CS1 (wake GENERIC[11])			
GPIO81	VCC_MSL	GPIO_81	UART1_DSR (wake GENERIC[3])	USB_P3_5 (wake ADxER[23])	UART1_DTR	MM2_CLK			UART3_CTS (wake GENERIC[5])
GPIO82	VCC_MSL	GPIO_82	UART1_RI (wake GENERIC[3])	USB_P3_6 (wake ADxER[23])		MM2_CMD (Wake GENERIC[11])			
GPIO83	VCC_MSL	GPIO_83 (wake GENERIC[12])	UART1_DTR		UART1_DSR (wake GENERIC[3])		KP_DKIN<2> (wake ADxER[21])		
GPIO84	VCC_MSL	GPIO_84	UART1_RTS	MM1_DATA<1> (Wake GENERIC[10])	UART1_CTS (wake GENERIC[3])		KP_DKIN<1> (wake ADxER[21])		

Copyright © 2009 Marvell

Table 12: PXA31x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO85	VCC_MSL	GPIO_85	SSPSCLK (wake GENERIC[1])	KP_MKOUT<0>	KP_DKIN<0> (wake ADxER[21])				
GPIO86	VCC_MSL	GPIO_86	SSPSFRM (wake GENERIC[1])	KP_MKOUT<1>	KP_DKIN<1> (wake ADxER[21])				
GPIO87	VCC_MSL	GPIO_87	SSPTXD	KP_MKOUT<2>	KP_DKIN<2> (wake ADxER[21])			SSPRXD (wake GENERIC[1])	
GPIO88	VCC_MSL	GPIO_88	SSPRXD (wake GENERIC[1])	KP_MKOUT<3>	KP_DKIN<3> (wake ADxER[21])			SSPTXD	
GPIO89	VCC_MSL	GPIO_89	SSPEXTCLK/SSPCLKEN (wake GENERIC[1])	SC_nVS1	MM1_DATA<2> (Wake GENERIC[10])				
GPIO90	VCC_MSL	GPIO_90	SSPSYSCLK	SC_nVS2			MM1_DATA<3> (Wake GENERIC[10])		
GPIO91	VCC_IO1	GPIO_91	SSPSCLK3 (wake GENERIC[7])	UART3_CTS (wake GENERIC[5])		UART3_RTS			
GPIO92	VCC_IO1	GPIO_92	SSPSFRM3 (wake GENERIC[7])	UART3_RTS		UART3_CTS (wake GENERIC[5])			
GPIO93	VCC_IO1	GPIO_93	SSPTXD3	UART3_TXD		UART3_RXD (wake GENERIC[5])	SSPRXD3 (wake GENERIC[7])		
GPIO94	VCC_IO1	GPIO_94	SSPRXD3 (wake GENERIC[7])	UART3_RXD (wake GENERIC[5])		UART3_TXD	SSPTXD3		
GPIO95	VCC_IO1	GPIO_95	SSPSCLK4 (wake GENERIC[8])						
GPIO96	VCC_IO1	GPIO_96	SSPSFRM4 (wake GENERIC[8])						
GPIO97	VCC_IO1	GPIO_97	SSPTXD4				SSPRXD4 (wake GENERIC[8])		
GPIO98	VCC_IO1	GPIO_98	SSPRXD4 (wake GENERIC[8])				SSPTXD4		
GPIO99	VCC_IO1	GPIO_99	UART1_RXD (wake GENERIC[3])					UART1_TXD	
GPIO100	VCC_IO1	GPIO_100	UART1_TXD					UART1_RXD (wake GENERIC[3])	KP_MKIN<6> (wake GENERIC[6])
GPIO101	VCC_IO1	GPIO_101	UART1_CTS (wake GENERIC[3])					UART1_RTS	KP_MKIN<7> (wake GENERIC[6])
GPIO102	VCC_IO1	GPIO_102	UART1_DCD (wake GENERIC[3])			UART1_TXD		UART1_RXD (wake GENERIC[3])	

Copyright © 2009 Marvell

Table 12: PXA31x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO103	VCC_IO1	GPIO_103	UART1_DSR (wake GENERIC[3])	MM3_CLK				UART1_DTR	
GPIO104	VCC_IO1	GPIO_104	UART1_RI (wake GENERIC[3])			UART1_RXD (wake GENERIC[3])	KP_MKOUT<6>		L_DD<19>
GPIO105	VCC_IO1	GPIO_105	UART1_DTR	MM3_CMD (wake GENERIC[12])			KP_MKOUT<7>	UART1_DSR (wake GENERIC[3])	
GPIO106	VCC_IO1	GPIO_106	UART1_RTS					UART1_CTS (wake GENERIC[3])	L_DD<20>
GPIO107	VCC_IO1	GPIO_107	UART3_CTS (wake GENERIC[5])	KP_DKIN<0> (wake ADxER[21])	UART3_RTS				L_DD<21>
GPIO108	VCC_IO1	GPIO_108	UART3_RTS	KP_DKIN<1> (wake ADxER[21])	UART3_CTS (wake GENERIC[5])				L_DD<22>
GPIO109	VCC_IO1	GPIO_109	UART3_TXD	KP_DKIN<2> (wake ADxER[21])	UART3_RXD (wake GENERIC[5])				
GPIO110	VCC_IO1	GPIO_110	UART3_RXD (wake GENERIC[5])	KP_DKIN<3> (wake ADxER[21])	UART3_TXD				
GPIO111	VCC_IO1	GPIO_111	UART2_RTS	KP_DKIN<4> (wake ADxER[21])	UART2_CTS (wake GENERIC[4])				
GPIO112	VCC_IO1	GPIO_112	UART2_RXD (wake GENERIC[4])	KP_DKIN<5> (wake ADxER[21])	UART2_TXD	KP_MKIN<6> (wake GENERIC[6])			
GPIO113	VCC_IO1	GPIO_113	UART2_TXD	KP_DKIN<6> (wake ADxER[21])	UART2_RXD (wake GENERIC[4])	KP_MKIN<7> (wake GENERIC[6])			
GPIO114	VCC_IO1	GPIO_114	UART2_CTS (wake GENERIC[4])	KP_DKIN<7> (wake ADxER[21])	UART2_RTS				
GPIO115	VCC_IO1	GPIO_115	KP_MKIN<0> (wake GENERIC[6])	KP_DKIN<0> (wake ADxER[21])					
GPIO116	VCC_IO1	GPIO_116	KP_MKIN<1> (wake GENERIC[6])	KP_DKIN<1> (wake ADxER[21])					
GPIO117	VCC_IO1	GPIO_117	KP_MKIN<2> (wake GENERIC[6])	KP_DKIN<2> (wake ADxER[21])					
GPIO118	VCC_IO1	GPIO_118	KP_MKIN<3> (wake GENERIC[6])	KP_DKIN<3> (wake ADxER[21])					
GPIO119	VCC_IO1	GPIO_119	KP_MKIN<4> (wake GENERIC[6])	KP_DKIN<4> (wake ADxER[21])					
GPIO120	VCC_IO1	GPIO_120	KP_MKIN<5> (wake GENERIC[6])	KP_DKIN<5> (wake ADxER[21])					
GPIO121	VCC_IO1	GPIO_121	KP_MKOUT<0>	KP_DKIN<6> (wake ADxER[21])					

Table 12: PXA31x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO122	VCC_IO1	GPIO_122	KP_MKOUT<1>	KP_DKIN<5> (wake ADxER[21])					
GPIO123	VCC_IO1	GPIO_123	KP_MKOUT<2>	KP_DKIN<4> (wake ADxER[21])					
GPIO124	VCC_IO1	GPIO_124	KP_MKOUT<3>	KP_DKIN<3> (wake ADxER[21])					
GPIO125	VCC_IO1	GPIO_125	KP_MKOUT<4>	KP_DKIN<2> (wake ADxER[21])					
GPIO126	VCC_IO1	GPIO_126	HZ_CLK	ONE_WIRE	CLK_EXT (wake GENERIC[12])		KP_MKOUT<7>		
GPIO127	VCC_IO1	GPIO_127 (wake GENERIC[12])	L_CS	CLK_48M			KP_DKIN<0> (wake ADxER[21])	L_DD<23>	
GPIO0_2	VCC_IO1	GPIO0_2 (wake GENERIC[12])	USBHPEN	KP_DKIN<0> (wake ADxER[21])					
GPIO1_2	VCC_IO1	GPIO1_2 (wake GENERIC[12])	USBHPWR	KP_DKIN<1> (wake ADxER[21])					
GPIO2_2	VCC_IO3	GPIO2_2	KP_MKIN<6> (wake GENERIC[6])	KP_DKIN<6> (wake ADxER[21])	SSPSFRM (wake GENERIC[1])				
GPIO3_2	VCC_IO3	GPIO3_2	KP_MKIN<7> (wake GENERIC[6])	KP_DKIN<7> (wake ADxER[21])	SSPSFRM (wake GENERIC[1])				
GPIO4_2	VCC_IO3	GPIO4_2	KP_MKOUT[5]	KP_DKIN[1] (wake ADxER[21])					
GPIO5_2	VCC_IO3	GPIO5_2	KP_MKOUT[6]	KP_DKIN[0] (wake ADxER[21])					
GPIO6_2	VCC_IO3	GPIO6_2	KP_MKOUT[7]		SSPSFRM (wake GENERIC[1])				
GPIO7_2	VCC_IO1	GPIO<7> _2	MM3_DATA<0> (wake GENERIC[12])						
GPIO8_2	VCC_IO1	GPIO<8> _2	MM3_DATA<1> (wake GENERIC[12])						
GPIO9_2	VCC_IO1	GPIO<9> _2	MM3_DATA<2> (wake GENERIC[12])						
GPIO10_2	VCC_IO1	GPIO<10> _2	MM3_DATA<3> (wake GENERIC[12])						

4.5 PXA30x Pin List with Alternate Functions

Table 13 defines the alternate functions for the PXA30x processor.

Copyright © 2009 Marvell

Table 13: PXA30x Processor Alternate Function Table

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO0	VCC_DF	GPIO_0 (wake GENERIC[12])	RDY						
GPIO1	VCC_DF	GPIO_1 (wake GENERIC[12])	nCS<2>						
GPIO2	VCC_DF	GPIO_2(wake GENERIC[12])	nCS<3>	nXCVREN					
GPIO3	VCC_CARD1	GPIO_3	U_IO (wake ADxER[19])	KP_DKIN<6> (wake ADxER[21])		MM1_DATA<0> (Wake GENERIC[10])			
GPIO4	VCC_CARD1	GPIO_4	U_DETECT (wake ADxER[19])	KP_DKIN<7> (wake ADxER[21])		MM1_DATA<1> (Wake GENERIC[10])			
GPIO5	VCC_CARD1	GPIO_5	U_CLK	KP_MKIN<0> (wake GENERIC[6])		MM1_DATA<2> (Wake GENERIC[10])			
GPIO6	VCC_CARD1	GPIO_6	U_nRST	KP_MKIN<1> (wake GENERIC[6])		MM1_DATA<3> (Wake GENERIC[10])			
GPIO7	VCC_CARD1	GPIO_7	KP_MKOUT<5>	UART3_RXD (wake GENERIC[5])		MM1_CLK		UART3_TXD	
GPIO8	VCC_CARD1	GPIO_8		UART3_TXD		MM1_CMD (Wake GENERIC[10])	CIR_OUT	UART3_RXD (wake GENERIC[5])	
GPIO9	VCC_CARD2	GPIO_9	SC_IO (wake ADxER[20])		KP_MKIN<6> (wake GENERIC[6])	MM2_DATA<0> (wake GENERIC[11])			
GPIO10	VCC_CARD2	GPIO_10	SC_DETECT (wake ADxER[20])		KP_MKIN<7> (wake GENERIC[6])	MM2_DATA<1> (wake GENERIC[11])			
GPIO11	VCC_CARD2	GPIO_11	SC_CLK		KP_MKOUT<5>	MM2_DATA<2> (wake GENERIC[11])			
GPIO12	VCC_CARD2	GPIO_12	SC_nRST		KP_MKOUT<6>	MM2_DATA<3> (wake GENERIC[11])			
GPIO13	VCC_CARD2	GPIO_13			KP_MKOUT<7>	MM2_CLK			
GPIO14	VCC_CARD2	GPIO_14				MM2_CMD (Wake GENERIC[11])	MM1_CMD (Wake GENERIC[11])		
GPIO15	VCC_CARD2	GPIO_15	SC_VS0	L_CS	UART2_CTS (wake GENERIC[4])	UART2_RTS	MM1_CMD (Wake GENERIC[11])	SSPSCLK (wake GENERIC[1])	
GPIO16	VCC_CARD2	GPIO_16	U_VS0	SSPSFRM (wake GENERIC[1])	CIR_OUT	UART2_RTS	UART2_CTS (Wake GENERIC[[4])	KP_DKIN<6> (wake ADxER[21])	
GPIO17	VCC_IO3	GPIO_17	PWM<0>	SSPSFRM2 (wake GENERIC[2])				EXT_SYNC0 (wake GENERIC[12])	
GPIO18	VCC_IO3	GPIO_18	PWM<1>	SSPRXD (wake GENERIC[1])		UART2_TXD	UART2_RXD (wake GENERIC[4])	EXT_SYNC1 (wake GENERIC[12])	SSPTXD1
GPIO19	VCC_IO3	GPIO_19	PWM<2>	SSPTXD2	KP_MKOUT<4>	UART2_RXD (wake GENERIC[4])	UART2_TXD	CHOUT0	SSPRXD2 (wake GENERIC[2])

Table 13: PXA30x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO20	VCC_IO3	GPIO_20	PWM<3>	SSPTXD	KP_MKOUT<5>	HZ_CLK	ONE_WIRE	CHOUT1	SSPRXD (wake GENERIC[1])
GPIO21	VCC_IO3	GPIO_21	SCL (wake GENERIC[9])						
GPIO22	VCC_IO3	GPIO_22	SDA (wake GENERIC[9])						
GPIO23	VCC_IO3	GPIO_23	AC97_nACRESET	SSPSYSCLK2					
GPIO24	VCC_IO3	GPIO_24	AC97_SYSCLK		UTM_RXVALID	SSPRXD2 (wake GENERIC[2])	SSPTXD2		
GPIO25	VCC_IO3	GPIO_25	AC97_SDATA_IN_0 (Wake GENERIC[0])	SSPSCLK2 (wake GENERIC[2])	UTM_RXACTIVE				
GPIO26	VCC_IO3	GPIO_26	AC97_SDATA_IN_1 (wake GENERIC[0])	SSPSFRM2 (wake GENERIC[2])	UTM_RXERROR				
GPIO27	VCC_IO3	GPIO_27	AC97_SDATA_OUT	SSPTXD2		UTM_OPMODE0	SSPRXD2 (wake GENERIC[2])		
GPIO28	VCC_IO3	GPIO_28	AC97_SYNC	SSPRXD2 (wake GENERIC[2])		UTM_OPMODE1	SSPTXD2		
GPIO29	VCC_IO3	GPIO_29	AC97_BITCLK (Wake GENERIC[0])	SSPEXTCLK2/ SSPCLK2EN (wake GENERIC[2])		UTM_TXVALID			
GPIO30	VCC_IO3	GPIO_30		UART1_RXD (wake GENERIC[3])	UTM_DATA<0>	UART1_TXD			
GPIO31	VCC_IO3	GPIO_31		UART1_TXD	UTM_DATA<1>	UART1_RXD (wake GENERIC[3])			
GPIO32	VCC_IO3	GPIO_32		UART1_CTS (wake GENERIC[3])	UTM_DATA<2>	UART1_RTS			
GPIO33	VCC_IO3	GPIO_33		UART1_DCD (wake GENERIC[3])	UTM_DATA<3>		SSPSCLK (wake GENERIC[1])	SSPSCLK2 (wake GENERIC[2])	
GPIO34	VCC_IO3	GPIO_34		UART1_DSR (wake GENERIC[3])	UTM_DATA<4>	UART1_DTR	SSPSFRM (wake GENERIC[1])	SSPSFRM2 (wake GENERIC[2])	
GPIO35	VCC_IO3	GPIO_35		UART1_RI (wake GENERIC[3])	UTM_DATA<5>	SSPRXD (wake GENERIC[1])	SSPTXD	SSPRXD2 (wake GENERIC[2])	SSPTXD2
GPIO36	VCC_IO3	GPIO_36		UART1_DTR	UTM_DATA<6>	UART1_DSR (wake GENERIC[3])	SSPTXD	SSPRXD (wake GENERIC[1])	SSPTXD2
GPIO37	VCC_IO3	GPIO_37		UART1_RTS	UTM_DATA<7>	UART1_CTS (wake GENERIC[3])			
GPIO38	VCC_IO3	GPIO_38	UTM_CLK				KP_MKOUT<5>		
GPIO39	VCC_CI	GPIO_39	CIF_DD<0>		UTM_DATA<0>				
GPIO40	VCC_CI	GPIO_40	CIF_DD<1>		UTM_DATA<1>				

Copyright © 2009 Marvell

Table 13: PXA30x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO41	VCC_CI	GPIO_41	CIF_DD<2>		UTM_DATA<2>				
GPIO42	VCC_CI	GPIO_42	CIF_DD<3>		UTM_DATA<3>				
GPIO43	VCC_CI	GPIO_43	CIF_DD<4>		UTM_DATA<4>				
GPIO44	VCC_CI	GPIO_44	CIF_DD<5>		UTM_DATA<5>				
GPIO45	VCC_CI	GPIO_45	CIF_DD<6>		UTM_DATA<6>				
GPIO46	VCC_CI	CIF_DD<7>	GPIO<46>		UTM_DATA<7>				
GPIO47	VCC_CI	GPIO_47	CIF_DD<8>	UTM_RXACTIVE					
GPIO48	VCC_CI	GPIO_48	CIF_DD<9>	UTM_RXVALID					
GPIO49	VCC_CI	CIF_MCLK	UTM_RXACTIVE	CLK_48M	GPIO<49>				
GPIO50	VCC_CI	CIF_PCLK	UTM_ERROR	GPIO<50>					
GPIO51	VCC_CI	CIF_HSYNC		UTM_OPMODE0	GPIO<51>				
GPIO52	VCC_CI	CIF_VSYNC		UTM_OPMODE1	GPIO<52>		UTM_TXVALID		
GPIO53	VCC_LCD	GPIO_53 (wake GENERIC[12])	UTM_TXREADY				KP_MKOUT6		
GPIO54	VCC_LCD	GPIO_54	L_DD<0>			SMEM_FADDR4			ML_DD<0>
GPIO55	VCC_LCD	GPIO_55	L_DD<1>			SMEM_FADDR5			ML_DD<1>
GPIO56	VCC_LCD	GPIO_56	L_DD<2>			SMEM_FADDR6			ML_DD<2>
GPIO57	VCC_LCD	GPIO_57	L_DD<3>			SMEM_FADDR7			ML_DD<3>
GPIO58	VCC_LCD	GPIO_58	L_DD<4>			SMEM_FADDR8			ML_DD<4>
GPIO59	VCC_LCD	GPIO_59	L_DD<5>			SMEM_FADDR9			ML_DD<5>
GPIO60	VCC_LCD	GPIO_60	L_DD<6>			SMEM_FADDR10			ML_DD<6>
GPIO61	VCC_LCD	GPIO_61	L_DD<7>			SMEM_FADDR11			ML_DD<7>
GPIO62	VCC_LCD	GPIO_62	L_DD<8>	L_CS		SMEM_FADDR12			ML_DD<8>
GPIO63	VCC_LCD	GPIO_63	L_DD<9>	L_VSYNC		SMEM_FADDR13			ML_DD<9>
GPIO64	VCC_LCD	GPIO_64	L_DD<10>	SSPSCLK2 (wake GENERIC[2])		SMEM_FADDR14	UTM_XCVR_SELECT		ML_DD<10>
GPIO65	VCC_LCD	GPIO_65	L_DD<11>	SSPSFRM2 (wake GENERIC[2])		SMEM_FADDR15	UTM_TERM_SELECT		ML_DD<11>
GPIO66	VCC_LCD	GPIO_66	L_DD<12>	SSPRXD2 (wake GENERIC[2])	UTM_SUSPEND	SSPTXD2	SMEM_FADDR16		ML_DD<12>

Table 13: PXA30x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO67	VCC_LCD	GPIO_67	L_DD<13>	SSPTXD2	UTM_LINESTATE0	SSPRXD2 (wake GENERIC[2])	SMEM_FADDR17		ML_DD<13>
GPIO68	VCC_LCD	GPIO_68	L_DD<14>	SSPSCLK3 (wake GENERIC[7])	UTM_LINESTATE1	SMEM_FADDR18			ML_DD<14>
GPIO69	VCC_LCD	GPIO_69	L_DD<15>	SSPSFRM3 (wake GENERIC[7])		SMEM_FADDR19	UTM_TXVALID		ML_DD<15>
GPIO70	VCC_LCD	GPIO_70	L_DD<16>	SSPTXD3	KP_MKIN<6> (wake GENERIC[6])	SMEM_FADDR20	SSPRXD3 (wake GENERIC[7])		
GPIO71	VCC_LCD	GPIO_71	L_DD<17>	SSPRXD3 (wake GENERIC[7])	KP_MKIN<7> (wake GENERIC[6])	SMEM_FADDR21	SSPTXD3		
GPIO72	VCC_LCD	GPIO_72	L_FCLK_RD			SMEM_FADDR22			ML_FCLK
GPIO73	VCC_LCD	GPIO_73	L_LCLK_A0			SMEM_FADDR23			ML_LCLK
GPIO74	VCC_LCD	GPIO_74	L_PCLK_WR			SMEM_FADDR24			ML_PCLK
GPIO75	VCC_LCD	GPIO_75	L_BIAS			SMEM_FADDR25			ML_BIAS
GPIO76	VCC_LCD	GPIO_76 (wake GENERIC[12])	UTM_RESET	L_VSYNC					
GPIO77	VCC_MSL	GPIO_77	UART1_RXD (wake GENERIC[3])	USB_P3_1 (wake ADxER[23])	UART1_TXD	MM2_DATA<0> (wake GENERIC[11])			
GPIO78	VCC_MSL	GPIO_78	UART1_TXD	USB_P3_2	UART1_RXD (wake GENERIC[3])	MM2_DATA<1> (wake GENERIC[11])	KP_MKOUT<7>		KP_MKOUT<6>
GPIO79	VCC_MSL	GPIO_79	UART1_CTS (wake GENERIC[3])	USB_P3_3	UART1_RTS	MM2_DATA<2> /MM2_CS0 (wake GENERIC[11])			
GPIO80	VCC_MSL	GPIO_80	UART1_DCD (wake GENERIC[3])	USB_P3_4 (wake ADxER[23])		MM2_DATA<3> /MM2_CS1 (wake GENERIC[11])			
GPIO81	VCC_MSL	GPIO_81	UART1_DSR (wake GENERIC[3])	USB_P3_5 (wake ADxER[23])	UART1_DTR	MM2_CLK			
GPIO82	VCC_MSL	GPIO_82	UART1_RI (wake GENERIC[3])	USB_P3_6 (wake ADxER[23])		MM2_CMD (Wake GENERIC[11])			
GPIO83	VCC_MSL	GPIO_83 (wake GENERIC[12])	UART1_DTR		UART1_DSR (wake GENERIC[3])		KP_DKIN<2> (wake ADxER[21])		
GPIO84	VCC_MSL	GPIO_84	UART1_RTS		UART1_CTS (wake GENERIC[3])		KP_DKIN<1> (wake ADxER[21])		
GPIO85	VCC_MSL	GPIO_85	SSPSCLK (wake GENERIC[1])	KP_MKOUT<0>	KP_DKIN<0> (wake ADxER[21])				UTM_TXVALID

Copyright © 2009 Marvell

Table 13: PXA30x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO86	VCC_MSL	GPIO_86	SSPSFRM (wake GENERIC[1])	KP_MKOUT<1>	KP_DKIN<1> (wake ADxER[21])				
GPIO87	VCC_MSL	GPIO_87	SSPTXD	KP_MKOUT<2>	KP_DKIN<2> (wake ADxER[21])		UTM_RXVALID	SSPRXD (wake GENERIC[1])	
GPIO88	VCC_MSL	GPIO_88	SSPRXD (wake GENERIC[1])	KP_MKOUT<3>	KP_DKIN<3> (wake ADxER[21])		UTM_RXACTIVE	SSPTXD	
GPIO89	VCC_MSL	GPIO_89	SSPEXTCLK/SSPCLKEN (wake GENERIC[1])	SC_nVS1			UTM_RXERROR		
GPIO90	VCC_MSL	GPIO_90	SSPSYSCLK	SC_nVS2					UTM_OPMODE0
GPIO91	VCC_IO1	GPIO_91	SSPSCLK3 (wake GENERIC[7])	UART3_CTS (wake GENERIC[5])		UART3_RTS			
GPIO92	VCC_IO1	GPIO_92	SSPSFRM3 (wake GENERIC[7])	UART3_RTS	UTM_LINESTATE0	UART3_CTS (wake GENERIC[5])			
GPIO93	VCC_IO1	GPIO_93	SSPTXD3	UART3_TXD	UTM_LINESTATE1	UART3_RXD (wake GENERIC[5])	SSPRXD3 (wake GENERIC[7])		
GPIO94	VCC_IO1	GPIO_94	SSPRXD3 (wake GENERIC[7])	UART3_RXD (wake GENERIC[5])		UART3_TXD	SSPTXD3		
GPIO95	VCC_IO1	GPIO_95	SSPSCLK4 (wake GENERIC[8])	UTM_RESET					
GPIO96	VCC_IO1	GPIO_96	SSPSFRM4 (wake GENERIC[8])		UTM_XCVR_SELECT				
GPIO97	VCC_IO1	GPIO_97	SSPTXD4		UTM_TERMSELECT		SSPRXD4 (wake GENERIC[8])		
GPIO98	VCC_IO1	GPIO_98	SSPRXD4 (wake GENERIC[8])	UTM_SUSPENDM			SSPTXD4		
GPIO99	VCC_IO1	GPIO_99	UART1_RXD (wake GENERIC[3])	USB_P2_2 (wake ADxxR[22])	USB_P2_5 (wake ADxxR[22])	USB_P2_6	UTM_TERM_SELECT	UART1_TXD	
GPIO100	VCC_IO1	GPIO_100	UART1_TXD	USB_P2_6	UTM_RESET	USB_P2_2 (wake ADxxR[22])	USB_P2_5 (wake ADxxR[22])	UART1_RXD (wake GENERIC[3])	KP_MKIN<6> (wake GENERIC[6])
GPIO101	VCC_IO1	GPIO_101	UART1_CTS (wake GENERIC[3])	USB_P2_1 (wake ADxxR[22])			UTM_XCVR_SELECT	UART1_RTS	KP_MKIN<7> (wake GENERIC[6])
GPIO102	VCC_IO1	GPIO_102	UART1_DCD (wake GENERIC[3])	USB_P2_4 (wake ADxxR[22])		UART1_TXD	UTM_TERM_SELECT	UART1_RXD (wake GENERIC[3])	
GPIO103	VCC_IO1	GPIO_103	UART1_DSR (wake GENERIC[3])	USB_P2_8	UTM_SUSPENDM			UART1_DTR	
GPIO104	VCC_IO1	GPIO_104	UART1_RI (wake GENERIC[3])	USB_P2_3 (wake ADxxR[22])	UTM_LINESTATE0	UART1_RXD (wake GENERIC[3])	KP_MKOUT<6>		

Table 13: PXA30x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO105	VCC_IO1	GPIO_105	UART1_DTR	USB_P2_5 (wake ADxxR[22])	UTM_LINESTATE1		KP_MKOUT<7>	UART1_DSR (wake GENERIC[3])	
GPIO106	VCC_IO1	GPIO_106	UART1_RTS	USB_P2_7 (wake ADxxR[22])		UTM_OPMODE1		UART1_CTS (wake GENERIC[3])	
GPIO107	VCC_IO1	GPIO_107	UART3_CTS (wake GENERIC[5])	KP_DKIN<0> (wake ADxER[21])	UART3_RTS				
GPIO108	VCC_IO1	GPIO_108	UART3_RTS	KP_DKIN<1> (wake ADxER[21])	UART3_CTS (wake GENERIC[5])				
GPIO109	VCC_IO1	GPIO_109	UART3_TXD	KP_DKIN<2> (wake ADxER[21])	UART3_RXD (wake GENERIC[5])	UTM_LINESTATE0			
GPIO110	VCC_IO1	GPIO_110	UART3_RXD (wake GENERIC[5])	KP_DKIN<3> (wake ADxER[21])	UART3_TXD		UTM_OPMODE1		
GPIO111	VCC_IO1	GPIO_111	UART2_RTS	KP_DKIN<4> (wake ADxER[21])	UART2_CTS (wake GENERIC[4])				
GPIO112	VCC_IO1	GPIO_112	UART2_RXD (wake GENERIC[4])	KP_DKIN<5> (wake ADxER[21])	UART2_TXD	KP_MKIN<6> (wake GENERIC[6])			
GPIO113	VCC_IO1	GPIO_113	UART2_TXD	KP_DKIN<6> (wake ADxER[21])	UART2_RXD (wake GENERIC[4])	KP_MKIN<7> (wake GENERIC[6])			
GPIO114	VCC_IO1	GPIO_114	UART2_CTS (wake GENERIC[4])	KP_DKIN<7> (wake ADxER[21])	UART2_RTS				
GPIO115	VCC_IO1	GPIO_115	KP_MKIN<0> (wake GENERIC[6])	KP_DKIN<0> (wake ADxER[21])					
GPIO116	VCC_IO1	GPIO_116	KP_MKIN<1> (wake GENERIC[6])	KP_DKIN<1> (wake ADxER[21])					
GPIO117	VCC_IO1	GPIO_117	KP_MKIN<2> (wake GENERIC[6])	KP_DKIN<2> (wake ADxER[21])					
GPIO118	VCC_IO1	GPIO_118	KP_MKIN<3> (wake GENERIC[6])	KP_DKIN<3> (wake ADxER[21])					
GPIO119	VCC_IO1	GPIO_119	KP_MKIN<4> (wake GENERIC[6])	KP_DKIN<4> (wake ADxER[21])					
GPIO120	VCC_IO1	GPIO_120	KP_MKIN<5> (wake GENERIC[6])	KP_DKIN<5> (wake ADxER[21])					
GPIO121	VCC_IO1	GPIO_121	KP_MKOUT<0>	KP_DKIN<6> (wake ADxER[21])					
GPIO122	VCC_IO1	GPIO_122	KP_MKOUT<1>	KP_DKIN<5> (wake ADxER[21])					
GPIO123	VCC_IO1	GPIO_123	KP_MKOUT<2>	KP_DKIN<4> (wake ADxER[21])					

Copyright © 2009 Marvell

Table 13: PXA30x Processor Alternate Function Table (Continued)

Pin Name	Power Supply	Primary Function at Reset (Alt FN 0)	Alt. FN 1	Alt. FN 2	Alt. FN 3	Alt. FN 4	Alt. FN 5	Alt. FN 6	Alt. FN 7
GPIO124	VCC_IO1	GPIO_124	KP_MKOUT<3>	KP_DKIN<3> (wake ADxER[21])					
GPIO125	VCC_IO1	GPIO_125	KP_MKOUT<4>	KP_DKIN<2> (wake ADxER[21])					
GPIO126	VCC_IO1	GPIO_126	HZ_CLK	ONE_WIRE	CLK_EXT (wake GENERIC[12])		KP_MKOUT<7>		
GPIO127	VCC_IO1	GPIO_127 (wake GENERIC[12])	L_CS				KP_DKIN<0> (wake ADxER[21])		
GPIO0_2	VCC_IO1	GPIO0_2 (wake GENERIC[12])	USBHPEN	KP_DKIN<0> (wake ADxER[21])					
GPIO1_2	VCC_IO1	GPIO1_2 (wake GENERIC[12])	USBHPWR	KP_DKIN<1> (wake ADxER[21])					
GPIO2_2	VCC_IO3	GPIO2_2	KP_MKIN<6> (wake GENERIC[6])	KP_DKIN<6> (wake ADxER[21])					
GPIO3_2	VCC_IO3	GPIO3_2	KP_MKIN<7> (wake GENERIC[6])	KP_DKIN<7> (wake ADxER[21])					
GPIO4_2	VCC_IO3	GPIO4_2	KP_MKOUT<5>	KP_DKIN<1> (wake ADxER[21])					
GPIO5_2	VCC_IO3	GPIO5_2	KP_MKOUT<6>	KP_DKIN<0> (wake ADxER[21])					
GPIO6_2	VCC_IO3	GPIO6_2	KP_MKOUT<7>						

4.6 Signal Descriptions

Table 14 describes the PXA3xx processor signals used by each interface. Most of the processor pins are multiplexed so that they can be configured for one of the up to eight available functions using the Multi-Function Pin Registers (MFPR). Some signals can be configured to appear on one of several different pins.

Note

The "Pin Usage" section in the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification (EMTS)* has more details about the processor pins and signals, including reset states.

Table 14: PXA3xx Processor Signal Descriptions

Signal Name	Type	Signal Descriptions	Notes
External Memory Pin Interface (EMPI) Signals			
MA<15:11>; MA<9:0>	Output	**Dynamic Memory Controller (DMC) Address Bus**— used for DDR SDRAM addressing.	
SDMA10	Output	**Address Bus Bit 10—** This signal drives address bit 10 to allow DDR SDRAM refresh cycles.	
MD<31:16>	Bidirectional	**Data Bus**— provides data to the DDR SDRAM device.	PXA32x Only
MD<15:0>	Bidirectional	**Data Bus**— provides data to the DDR SDRAM device.	
nSDWE	Output	**SDRAM Write Enable—**Connects to the write enables of DDR SDRAM memory devices.	
DQM<3:2>	Output	**SDRAM DQM Data Byte Mask Control—**Connects to the data output mask enables (DQM) for DDR SDRAM. (DQM2 corresponds to MD<23:16> and DQM3 corresponds to MD<31:24>)	PXA32x Only
DQM<1:0>	Output	**SDRAM DQM Data Byte Mask Control—**Connects to the data output mask enables (DQM) for DDR SDRAM. (DQM0 corresponds to MD<7:0> and DQM1 corresponds to MD<15:8>)	
DQS<3:2>	Bidirectional	**Data Strobe for DDR SDRAM**— used by the memory controller to latch data during reads from DDR SDRAM. On writes to D DRSDRAM, DQS<1:0> are used by DDR SDRAM to latch data. Data is latched on both rising and falling edges.	PXA32x Only
DQS<1:0>	Bidirectional	**Data Strobe for DDR SDRAM**— used by the memory controller to latch data during reads from DDR SDRAM. On writes to D DRSDRAM, DQS<1:0> are used by DDR SDRAM to latch data. Data is latched on both rising and falling edges.	
nSDRAS	Output	**SDRAM RAS**—Connects to the row address strobe (RAS) pins for all banks of DDR SDRAM.	
nSDCAS	Output	**SDRAM CAS**—Connects to the column address strobe (CAS) pins for all banks of DDR SDRAM. Also functions as the active low address valid strobe for synchronous flash.	
SDCKE	Output	**SDRAM Clock Enable**—Connects to the clock-enable pins of all DDR SDRAM.	

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
SDCLK0 SDCLK1	Output	**DDR SDRAM Differential Clock**—Connect SDCLK0 to the (+) and SDCLK1 to the (-) differential clock pins of DDR SDRAM.	
RCOMP_DDR	Analog	**Resistive Compensation**—Connects to an external resistor to support EMPI impedance control.	
nSDCS1 nSDCS0	Output	**SDRAM Chip Selects**—Chip selects for DDR SDRAM memory devices, individually programmable in the memory configuration registers.	
Data Flash Interface Signals (Static Memory Controller)			
nXCVREN	Output	**Transceiver Enable—**Active low control signal for a transparent latch/transceiver between the DF_IO<15:0> bus. The latch should be transparent when nXCVREN is low. The direction of this transceiver is controlled by a RDnWR logical signal latched from the data pins earlier in the cycle with nLUA.	
DF_SCLK_E	Output	**External Clock**—Connects to the clock input of external synchronous device.	
nBE<1:0>	Output	**Byte Enable 1 and 0—**Active low byte enable. When high, this signal indicates the appropriate data byte should be masked. nBE<0> corresponds to DF_IO<7:0>, nBE<1> corresponds to DF_IO<15:8>.	
nLUA	Output	**Latch Upper Address**—Used to latch the high order address bits and transceiver control signal (RDnWR) on DF_IO<15:0>.	
nLLA	Output	**Latch Lower Address—**Used to latch the low order address bits on DF_IO<15:0>.	
nCS3 nCS2	Output	**Chip Selects**—Active-low chip selects for static memory devices.	
nCS0 nCS1	Output	**Chip Selects**—Active-low chip selects for static memory devices.	PXA30x and PXA31x processors only
DF_ADDR<3:0>	Output	**Low Order Address—**Lower address bits for the current static memory access. These signals provide the lowest four address bits during a transfer to eliminate the need for an additional address latch cycle (nLLA).	
RDY	Input	**Variable Latency I/O Ready—**An external VLIO device asserts RDY (high) when it is ready to transfer data.	
DREQ	Input	**DMA Request Signal—**An external VLIO device asserts RDY (high) when it is ready to transfer data.	PXA32x Only
DREQ<3:1>	Input	**DMA Request Signal—**An external VLIO device asserts RDY (high) when it is ready to transfer data.	PXA31x Only
Data Flash Interface Signals (NAND Flash Controller)			
DF_INT_RNB	Input	**Ready/Busy# signal —** used as an input to signal to the NAND Flash Controller when the external NAND device is ready.	
ND_nRE	Output	**Read Enable—**Enables reads from NAND flash devices	
ND_nWE	Output	**Write Enable—**Enables writes to NAND flash devices	

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
ND_nCS1 ND_nCS0	Output	**Chip Selects—**Active low chip selects for NAND flash devices.	
Data Flash Interface Signals (Shared SMC and NFC)			
DF_IO<15:0>	Bidirectional	**Data/Address Bus**—Carries data and multiplexed address information	
DF_CLE_nOE	Output	**Static Memory Controller Output Enable—**Enable reads from static memory. **NAND Flash Command Latch Enable—**Connects to the CLE input of NAND flash devices	
DF_ALE_nWE	Output	**Static Memory Controller Write Enable—**Enables writes to static memory. **NAND Flash Address Latch Enable—**Connects to the ALE input of NAND flash devices.	
LCD Controller Signals			
L_DD<23:0> L_DD<7:0>	Bidirectional	**LCD Display Data**—Transfers pixel information from the LCD controller to the external LCD panel. **Smart Panel Data:** These pins become inputs driven by the panel during a read from a panel with an integrated frame buffer.	
ML_DD<17:0>	Output	**Low-Power Mode LCD Display Data**—Transfers pixel information from the low-power (mini) LCD controller to the external LCD panel in the S0/D1/C2 low-power mode. **NOTE:** The low-power mode LCD signals are connected to the same GPIO pins as the main LCD signals (L_DD<17:0>)	PXA30x and PXA31x processors only
L_CS	Output	**Smart Panel Chip Select—**Chip select signal for LCD panels with an internal frame buffer.	
L_FCLK_RD	Output	**LCD Frame Clock—**Frame clock used by the LCD display module to signal the start of a new frame of pixels that resets the line pointers to the top of the screen. This pin is also the vertical synchronization signal for active (TFT) displays. **Smart Panel Read Enable—**This pin is the read signal during reads from a panel with an internal frame buffers.	
L_LCLK_A0	Output	**LCD Line Clock—**Indicates the start of a new line. Also referred to as HSYNC for active panels. **Smart Panel Command/Data Select—** For LCDs with an internal frame buffer, this signal indicates a command or data transaction**.**	
L_PCLK_WR	Output	**LCD Pixel Clock**—Pixel clock used by the LCD display module to clock the pixel data into the Line Shift register. **NOTE:** In passive mode, the pixel clock toggles only when valid data is available on the data pins. In active mode, the pixel clock toggles continuously, and the AC bias pin is used as an output to signal when data is valid on the LCD data pins. **Smart Panel Write Enable—**This pin also functions as a write signal for LCD panels with an internal frame buffer.	

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
ML_FCLK	Output	**Low-Power Mode LCD Frame Clock—**Vertical synchronization signal for active (TFT) displays in the S0/D1/C2 low-power mode. **NOTE:** The low-power mode LCD signals are connected to the same GPIO pins as the main LCD signals (L_FCLK_RD)	PXA30x and PXA31x processors only
ML_LCLK	Output	**Low-Power Mode LCD Line Clock—**HSYNC for active panels in the S0/D1/C2 low-power mode. **NOTE:** The low-power mode LCD signals are connected to the same GPIO pins as the main LCD signals (L_LCLK_A0)	PXA30x and PXA31x processors only
ML_PCLK	Output	**Low-Power Mode LCD Pixel Clock**—Pixel clock for active TFT displays in the S0/D1/C2 low-power mode. **NOTE:** The low-power mode LCD signals are connected to the same GPIO pins as the main LCD signals (L_PCLK_WR)	PXA30x and PXA31x processors only
L_VSYNC	Input	**Smart Panel LCD Refresh Sync**—Sync input driven by LCDs with an internal frame buffer	
L_BIAS	Output	**LCD Bias Drive—**AC bias used to signal the LCD display module to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. In active (TFT) mode, it is used as the output enable to signal when data should be latched from the data pins using the pixel clock.	
ML_BIAS	Output	**Low-Power Mode LCD Bias Drive—**Output enable for active TFT panels to signal when data should be latched from the data pins using the pixel clock in the S0/D1/C2 low-power mode. **NOTE:** The low-power mode LCD signals are connected to the same GPIO pins as the main LCD signals (L_BIAS)	PXA30x and PXA31x processors only
UART Signals			
UART1_RXD	Input	UART 1 Receive Data	
UART1_TXD	Output	UART 1 Transmit Data	
UART1_CTS	Input	UART 1 Clear-to-Send	
UART1_DCD	Input	UART 1 Data-Carrier-Detect	
UART1_DSR	Input	UART 1 Data-Set-Ready	
UART1_RI	Input	UART 1 Ring Indicator	
UART1_DTR	Output	UART 1 Data-Terminal-Ready	
UART1_RTS	Output	UART 1 Request-to-Send	
UART2_RXD	Input	UART 2 Receive Data	
UART2_TXD	Output	UART 2 Transmit Data	
UART2_CTS	Input	UART 2 Clear-to-Send	
UART2_RTS	Output	UART 2 Request-to-Send	
UART3_RXD	Input	UART 3 Receive Data	
UART3_TXD	Output	UART 3 Transmit Data	
UART3_CTS	Input	UART 3 Clear-to-Send	

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
UART3_RTS	Output	UART 3 Request-to-Send	
Consumer Infrared Controller Signals			
CIR_OUT	Output	**Consumer Infrared Output**—CIR unit output that connects to the IR diode	
MultiMediaCard/SD/SDIO Controller #1 and #2 Signals			
MM1_CLK	Output	**MultiMediaCard and SD Card Bus Clock**—MMC/SD/SDIO Controller #1	
MM1_CMD	Bidirectional	**MultiMediaCard Command**—MMC/SD/SDIO Controller #1 MMC and SD:bidirectional line for command and response tokens SPI:output for command and write data	
MM1_DAT0	Bidirectional	**MultiMediaCard Data 0**—MMC/SD/SDIO Controller #1 MMC and SD: Bidirectional line for read and write data. SPI: input for response token and read data	
MM1_DAT1	Bidirectional	**MultiMediaCard Data 1**—MMC/SD/SDIO Controller #1 MMC and SD: Bidirectional line for read and write data. SDIO:Signals an interrupt condition to the controller.	
MM1_DAT2/MM1_CS0	Bidirectional	**MultiMediaCard Data 2**—MMC/SD/SDIO Controller #1 MMC and SD: Bidirectional line for read and write data. **MultiMediaCard Chip Select 0**—MMC/SD/SDIO Controller #1 SPI:chip select 0	
MM1_DAT3/MM1_CS1	Bidirectional	**MultiMediaCard Data 3**—MMC/SD/SDIO Controller #1 MMC and SD: Bidirectional line for read and write data. **MultiMediaCard Chip Select 1**—MMC/SD/SDIO Controller #1 SPI:chip select 1	
MM2_CLK	Output	**MultiMediaCard and SD Card Bus Clock**—MMC/SD/SDIO Controller #2	
MM2_CMD	Bidirectional	**MultiMediaCard Command**—MMC/SD/SDIO Controller #2 MMC and SD:Bidirectional line for command and response tokens SPI: Output for command and write data	
MM2_DAT0	Bidirectional	**MultiMediaCard Data 0—**MMC/SD/SDIO Controller #2 MMC and SD: Bidirectional line for read and write data. SPI: Input for response token and read data	
MM2_DAT1	Bidirectional	**MultiMediaCard Data 1—**MMC/SD/SDIO Controller #2 MMC and SD: Bidirectional line for read and write data. SDIO:Signals an interrupt condition to the controller.	
MM2_DAT2/MM2_CS0	Bidirectional	**MultiMediaCard Data 2**—MMC/SD/SDIO Controller #2 MMC and SD: Bidirectional line for read and write data. **MultiMediaCard Chip Select 0**—MMC/SD/SDIO Controller #2 SPI:chip select 0	
MM2_DAT3/MM2_CS1	Bidirectional	**MultiMediaCard Data 3**—MMC/SD/SDIO Controller #2 MMC and SD: Bidirectional line for read and write data. **MultiMediaCard Chip Select 1**—MMC/SD/SDIO Controller #2 SPI:chip select 1	

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
MultiMediaCard/SD/SDIO Controller #3 Signals **NOTE:** Controller #3 Is for the PXA31x Processor Only			
MM3_CLK	Output	**MultiMediaCard and SD Card Bus Clock**—MMC/SD/SDIO Controller #3	PXA31x only
MM3_CMD	Bidirectional	**MultiMediaCard Command**—MMC/SD/SDIO Controller #3 MMC and SD:bidirectional line for command and response tokens SPI:output for command and write data	PXA31x only
MM3_DAT0	Bidirectional	**MultiMediaCard Data 0—**MMC/SD/SDIO Controller #3 MMC and SD: Bidirectional line for read and write data. SPI: Input for response token and read data	PXA31x only
MM3_DAT1	Bidirectional	**MultiMediaCard Data 1—**MMC/SD/SDIO Controller #3 MMC and SD: Bidirectional line for read and write data. SDIO:Signals an interrupt condition to the controller.	PXA31x only
MM3_DAT2/MM3_CS0	Bidirectional	**MultiMediaCard Data 2**—MMC/SD/SDIO Controller #3 MMC and SD: Bidirectional line for read and write data. **MultiMediaCard Chip Select 0**—MMC/SD/SDIO Controller #3 SPI:chip select 0	PXA31x only
MM3_DAT3/MM3_CS1	Bidirectional	**MultiMediaCard Data 3**—MMC/SD/SDIO Controller #3 MMC and SD: Bidirectional line for read and write data. **MultiMediaCard Chip Select 1**—MMC/SD/SDIO Controller #3 SPI:chip select 1	PXA31x only
Universal Subscriber ID (USIM) Signals			
U_IO	Bidirectional	**USIM I/O**—USIM data signal	
U_VS0	Output	**USIM Voltage Select 0**—this output goes high to disable the USIM card power and connect VCC_CARD1 to VSS_CARD.	
U_nVS1	Output	**USIM Voltage Select 1**—this output goes low to enable the external USIM card power supply that provides 1.8 V on VCC_CARD1.	PXA32x Only
U_nVS2	Output	**USIM Voltage Select 2—**This output goes low to enable the external USIM card power supply that provides 3.0 V on VCC_CARD1.	PXA32x Only
U_CLK	Output	**USIM Clock**—USIM card clock signal.	
U_nRST	Output	**USIM Reset**—USIM card reset signal.	
U_DETECT	Input	**USIM**—Card Detect signal for USIM.	
Smart Card Signals			
SC_IO	Bidirectional	**Smart Card I/O**—Smart Card data signal	
SC_VS0	Output	**Smart Card Voltage Select 0**—this output goes high to disable the SmartCard card power and connect VCC_CARD2 to VSS_CARD.	
SC_nVS1	Output	**Smart Card Voltage Select 1**—this output goes low to enable the external SmartCard card power supply that provides 1.8 V on VCC_CARD2.	
SC_nVS2	Output	**Smart Card Voltage Select 2—**This output goes low to enable the external SmartCard card power supply that provides 3.0 V on VCC_CARD2.	

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
SC_CLK	Output	**Smart Card Clock**—SmartCard card clock signal.	
SC_nRST	Output	**Smart Card Reset**—SmartCard card reset signal.	
SC_DETECT	Input	**Smart Card Detect—**Card Detect signal for SmartCard.	
Keypad Controller Signals			
KP_DKIN<7:0>	Input	Keypad Direct Key Inputs	
KP_MKIN<7:0>	Input	Keypad Matrix Key Inputs	
KP_MKOUT<7:0>	Output	Keypad Matrix Key Outputs	
SSP Signals			
SSPSCLK	Bidirectional	**Synchronous Serial Port Clock 1**—the serial bit clock may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPSFRM	Bidirectional	**Synchronous Serial Port Frame 1**—the serial frame sync may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPTXD	Output	**Synchronous Serial Port Transmit Data 1**—serial data driven out synchronously with the bit clock	
SSPRXD	Input	**Synchronous Serial Port Receive Data 1**—serial data latched using the bit clock	
SSPEXTCLK/ SSPCKEN	Input	**Synchronous Serial Port External Clock 1**—this input may be used to supply an external bit clock or an external enable request for the internally generated bit clock.	
SSPSYSCLK	Output	**Synchronous Serial Port 1 System Clock**—When enabled, provides a reference clock at four times the port 1-bit clock.	
SSPSCLK2	Bidirectional	**Synchronous Serial Port Clock 2**—the serial bit clock may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPSFRM2	Bidirectional	**Synchronous Serial Port Frame 2**—the serial frame sync may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPTXD2	Output	**Synchronous Serial Port Transmit Data 2**—serial data driven out synchronously with the bit clock.	
SSPRXD2	Input	**Synchronous Serial Port Receive Data 2**—serial data latched using the bit clock.	
SSPEXTCLK2/SSPCLK EN2	Input	**Synchronous Serial Port External Clock 2**—this input may be used to supply an external bit clock or an external enable request for the internally generated bit clock.	
SSPSYSCLK2	Output	**Synchronous Serial Port 2 System Clock**—When enabled, provides a reference clock at four times the Port 2 bit clock.	
SSPSCLK3	Bidirectional	**Synchronous Serial Port Clock 3**—the serial bit clock may be configured as an output (master mode operation) or an input (slave mode operation).	

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
SSPSFRM3	Bidirectional	**Synchronous Serial Port Frame 3—**the serial frame sync may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPTXD3	Output	**Synchronous Serial Port Transmit Data 3**—serial data driven out synchronously with the bit clock	
SSPRXD3	Input	**Synchronous Serial Port Receive Data 3**—serial data latched using the bit clock	
SSPSCLK4	Bidirectional	**Synchronous Serial Port Clock 4**—the serial bit clock may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPSFRM4	Bidirectional	**Synchronous Serial Port Frame 4**—the serial frame sync may be configured as an output (master mode operation) or an input (slave mode operation).	
SSPTXD4	Output	**Synchronous Serial Port Transmit Data 4**—serial data driven out synchronously with the bit clock	
SSPRXD4	Input	**Synchronous Serial Port Receive Data 4**—serial data latched using the bit clock	
USB Full Speed Single-Ended Signals - Host Port 3			
USB_P3_1	Bidirectional	**USB Full Speed Host Port 3 RCV—**receive data signal which connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control register (UP3OCR).	
USB_P3_2	Bidirectional	**USB Full Speed Host Port 3 OE**—output enable signal which connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control register (UP3OCR).	
USB_P3_3	Bidirectional	**USB Full Speed Host Port 3 RXD–** —receive data (-) signal which connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control register (UP3OCR).	
USB_P3_4	Bidirectional	**USB Full Speed Host Port 3 TXD–** —transmit data (-) signal which connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control register (UP3OCR).	
USB_P3_5	Bidirectional	**USB Full Speed Host Port 3 RXD+** —receive data (+) signal which connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control register (UP3OCR).	
USB_P3_6	Bidirectional	**USB Full Speed Host Port 3 TXD+** —transmit data (+) signal which connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control register (UP3OCR).	
USB Full Speed Single-Ended Signals - Port 2 Host/Client and OTG			

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
USB_P2_1	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 RCV/INT/SRP—**This signal is the receive data from an external USB transceiver for Port 2. When configured for an external OTG transceiver this signal is an interrupt input. When configured for an external OTG power controller and the internal transceiver, this signal is the SRP detect input.	
USB_P2_2	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 OE/Valid**—This signal connects to the OE signal of an external USB transceiver for USB Port 2. This signal connects to the session valid output of an external OTG power controller.	
USB_P2_3	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 RXD-/SV—**This signal is the receive negative data line from an external USB transceiver for Port 2. For an external OTG power controller, this signal is the session valid status input.	
USB_P2_4	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 TXD-/SE0_VM/SRP**—This signal is the transmit negative data line to an external USB transceiver for Port 2. For an external OTG transceiver, this signal is the SE0_VM bidirectional data line. For an external OTG power controller this signal is the SRP enable control output.	
USB_P2_5	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 RXD+/DAT_VP/VALID40**—This signal is the receive positive data line from an external USB transceiver for Port 2. For an external OTG transceiver, this signal is the DAT_VP bidirectional data line. For an external OTG power controller this signal is the 4.0V Vbus valid status input.	
USB_P2_6	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 TXD+ —**This signal is the positive transmit data line for an external USB Transceiver for port 2.	
USB_P2_7	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 Speed/OTGID**—Speed select signal for USB Port 2 when configured for an external USB transceiver. Provides the OTG ID configuration using the internal transceiver.	PXA30x and PXA32x Only
USB_P2_8	Bidirectional	**USB Full Speed Host/Client and OTG Port 2 Suspend**—Suspend enable for USB Port 2 when configured for an external USB OTG transceiver.	PXA30x and PXA32x Only
USB Full Speed Transceiver Differential Signals - OTG Port 2 and Host Port 1 **NOTE:** Available On PXA32x and PXA30x Processors Only			
USBC_P	Analog	**USB Full Speed Client and OTG Port 2 Positive Line**—this differential signal connects to the USB client interface. This signal is routed to the pin named USBOTG_P.	PXA30x and PXA32x Only, 1
USBC_N	Analog	**USB Full Speed Client and OTG Port 2 Negative Line**—this differential signal connects to the USB client interface. This signal is routed to the pin named USBOTG_N.	PXA30x and PXA32x Only, 2
USBH2_P	Analog	**USB Full Speed Host/Client and OTG Port 2 Positive Line**—this differential signal connects to the USB OTG interface for Port 2. This signal is routed to the pin named USBOTG_P.	PXA30x and PXA32x Only, 1

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
USBH2_N	Analog	**USB Full Speed Host/Client and OTG Port 2 Negative Line**—this differential signal connects to the USB OTG interface for Port 2. This signal is routed to the pin named USBOTG_N.	PXA30x and PXA32x Only, 2
USBH1_P	Analog	**USB Full Speed Host Positive Line**—this differential signal connects to the USB host interface for host Port 1.	PXA30x and PXA32x Only
USBH1_N	Analog	**USB Full Speed Host Negative Line**—this differential signal connects to the USB host interface for host Port 1.	PXA30x and PXA32x Only
USBHPWR	Input	**USB Full Speed Host Port 1 Power Indicator**—over-current indicator from USB host Port 1	PXA30x and PXA32x Only
USBHPEN	Output	**USB Full Speed Host Port 1 Power Control—**controls power to the USB host Port 1	PXA30x and PXA32x Only,
USBHPEN2	Output	**USB Full Speed Host Port 2 Power Control—**controls power to the USB host Port 2. This signal is muxed with USB_P2_8 but depends on the value of UP2OCR[SEOS]. Refer to The *Universal Serial Bus Client Controller* chapter of *Vol. IV: The PXA30x Processor and PXA31x Processor Developers Manual: Serial Controller Configuration* for more information.	PXA30x and PXA32x Only,
USB 2.0 High-Speed Client UTMI Transceiver Interface Signals **NOTE:** The UMTI Interface Is Available On PXA32x and PXA30x Processors Only			
UTM_CLK	Input	**UTMI Clock—**connect to the clock output of an external UTMI transceiver.	PXA30x and PXA32x Only
UTM_DATA<7:0>	Bidirectional	**UTMI Data Bus—**connect to the data bus of an external UTMI transceiver.	PXA30x and PXA32x Only
UTM_RESET	Output	**UTMI Reset—**connect to the reset input of an external UTMI transceiver.	PXA30x and PXA32x Only
UTM_XCVR_SELECT	Output	**UTMI Transceiver Select—**connect to the external UTMI transceiver input that selects high speed and full speed operating modes. The full speed transceiver should be enabled when this signal is high.	PXA30x and PXA32x Only
UTM_TERM_SELECT	Output	**UTMI Termination Select—**connect to the external UTMI transceiver input that selects high speed and full speed termination modes. The full speed termination is enabled when this signal is high.	PXA30x and PXA32x Only
UTM_SUSPENDM_X	Output	**UTMI Suspend**—connect to the external UTMI transceiver input that goes low to place the transceiver in a mode that draws minimal power from supplies while retaining the capability for suspend/resume operation.	PXA30x and PXA32x Only

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
UTM_LINESTATE<1:0>	Input	**UTMI Line State**—connect s to the single ended receiver status signals. They are asynchronous until a usable CLK is available then they are synchronized to CLK. They directly reflect the current state of the DP (LineState[0]) and DM (LineState[1]) signals: D– D+ Description 0 0 SE0 0 1 'J' State 1 0 'K' State 1 1 SE1	PXA30x and PXA32x Only
UTM_TXVALID	Output	**UTMI Transmit Valid—**Indicates that the transmit output data is valid.	PXA30x and PXA32x Only
UTM_TXREADY	Input	**UTMI Transmit Data Ready**—If this signal is asserted, the controller will have data available for clocking in to the TX Holding register on the rising edge of UTM_CLK.	PXA30x and PXA32x Only
UTM_RXVALID	Input	**UTMI Receive Data Valid**—Indicates that the data bus has valid data.	PXA30x and PXA32x Only
UTM_RXACTIVE	Input	**UTMI Receive Active**—Indicates that the receive state machine has detected SYNC and is active.	PXA30x and PXA32x Only
UTM_RXERROR	Input	**UTMI Receive Error**—connects to the transceiver error output. High indicates that a receive error has been detected.	PXA30x and PXA32x Only
UTM_OPMODE<1:0>	Output	**UTMI Operating Mode**—these signals configure the transceiver operating modes: OPMODE<1:0> Description 0 0 Normal Operation 0 1 Non-Driving 1 0 Disable Bit Stuffing and NRZI encoding 1 1 Reserved	PXA30x and PXA32x Only
USB 2.0 High-Speed Client UPLI Transceiver Interface Signals **NOTE:** The ULPI Interface Is Available On PXA31x Processor Only			
ULPI_CLK	Input	Clock. This input from the PHY is used for clocking receive and transmit parallel data	PXA31x Only
ULPI_DATA<7:0>	Bidirectional	8-bit data bus to the PHY	PXA31x Only
ULPI_STP	Output	The Link asserts ulpi_stp for one clock cycle to stop the data stream currently on the bus. If the Link is sending data to the PHY, ULPI_STP indicates the last byte of data was on the bus in the previous cycle.	PXA31x Only
ULPI_NXT	Input	The PHY asserts ULPI_NXT to throttle the data. When the Link is sending data to the PHY, ULPI_NXT indicates when the current byte has been accepted by the PHY. The Link places the next byte on the data bus in the following clock cycle.	PXA31x Only
ULPI_DIR	Input	Controls the direction of the data bus. When the PHY has data to transfer to the Link, it drives ULPI_DIR high to take ownership of the bus. When the PHY has no data to transfer, it drives ULPI_DIR low and monitors the bus for commands from the Link. The PHY will pull ULPI_DIR high whenever the interface cannot accept data from the Link, such as during PLL startup.	PXA31x Only

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
ULPI_OTG_INTR	Input	Interrupt, used for Serial Mode, Low-Power Mode, and Carkit Mode	PXA31x Only
Quick Capture Interface Signals			
CIF_MCLK	Output	Quick Capture Interface Master Clock Signal	
CIF_PCLK	Input	Quick Capture Interface Pixel Clock Signal	
CIF_DD<9:0>	Input	Quick Capture Interface Data Signals	
CIF_VSYNC	Bidirectional	**Quick Capture Interface Frame Synchronization Signal** - vertical sync signal	
CIF_HSYNC	Bidirectional	**Quick Capture Interface Line Synchronization Signal** - horizontal sync signal	
AC'97 Controller Signals			
AC97_nACRESET	Output	**AC '97 Reset**—active-low CODEC reset	
AC97_BITCLK	Input	**AC '97 Bit Clock**—bit-rate clock	
AC97_SYNC	Output	**AC '97 Sync**—frame indicator and synchronizer	
AC97_SDATA_OUT	Output	**AC '97 Serial Data Out**—serial audio data output to the CODEC for digital-to-analog conversion	
AC97_SDATA_IN_0	Input	**AC '97 Serial Data In 0**—serial audio data from the primary CODEC analog-to-digital converter	
AC97_SDATA_IN_1	Input	**AC '97 Serial Data In 1**—serial audio data from the secondary CODEC analog-to-digital converter	
AC97_SYSCLK	Output	**AC '97 System Clock**—AC97 system clock output	
I^2C Interface Signals			
SCL	Bidirectional	**I^2C Clock**—serial clock	
SDA	Bidirectional	**I^2C Data**—serial data/address bus	
PWM Signals			
PWM<3>	Output	**Pulse Width Modulation Channel 3**—pulse width modulator channel 3 output	
PWM<2>	Output	**Pulse Width Modulation Channel 2—**pulse width modulator channel 2 output	
PWM<1>	Output	**Pulse Width Modulation Channel 1**—pulse width modulator channel 1 output	
PWM<0>	Output	**Pulse Width Modulation Channel 0**—pulse width modulator channel 0 output	
Touch Screen Interface			
TSI_XP	Analog	**TSI X Positive**—This input connects to the X+ terminal of the touchscreen.	PXA32x only
TSI_XM	Analog	**TSI X Negative**—This input connects to the X- terminal of the touchscreen.	PXA32x only

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
TSI_YP	Analog	**TSI Y Positive**—This input connects to the Y+ terminal of the touchscreen.	PXA32x only
TSI_YM	Analog	**TSI Y Negative**—This input connects to the Y- terminal of the touchscreen.	PXA32x only
General-Purpose I/O Signals			
GPIO<127:0>	Bidirectional	**General-Purpose I/O signals—**these software managed logical I/O channels support interrupt generation with rising and/or falling edge detection. GPIO channels may be mapped to multiple package balls, but not all balls support a GPIO alternate function option.	
GPIO<6:0>_2	Bidirectional	**Alternate General-Purpose I/O signals—**these signals duplicate GPIO alternate function options on additional pins. If one of these functions is selected with the primary function also selected on another pin, then the GPIO channel is connected to both. When the GPIO channel is an output it is driven on both pins. When the GPIO is an input (the default at reset), then the two inputs are OR'ed and driven to the GPIO input channel.	PXA30x Only
GPIO0<10:0>_2	Bidirectional	**Alternate General-Purpose I/O signals—**these signals duplicate GPIO alternate function options on additional pins. If one of these functions is selected with the primary function also selected on another pin, then the GPIO channel is connected to both. When the GPIO channel is an output it is driven on both pins. When the GPIO is an input (the default at reset), then the two inputs are OR'ed and driven to the GPIO input channel.	PXA31x Only
GPIO0<17:0>_2	Bidirectional	**Alternate General-Purpose I/O signals—**these signals duplicate GPIO alternate function options on additional pins. If one of these functions is selected with the primary function also selected on another pin, then the GPIO channel is connected to both. When the GPIO channel is an output it is driven on both pins. When the GPIO is an input (the default at reset), then the two inputs are OR'ed and driven to the GPIO input channel.	PXA32x Only
Crystal and Clock Signals			
PXTAL_IN	Input	**Processor Crystal Input—**can be connected to an external 13 MHz crystal or to an external clock source.	
PXTAL_OUT	Analog	**Processor Crystal Output—**can be connected to an external 13 MHz crystal or to an external clock source (which must be complementary to PXTAL_IN). **NOTE:** If an external 13MHz oscillator is used, PXTAL_OUT MUST be left floating	
TXTAL_IN	Input	**Timekeeping Crystal Input—**clock input that is distributed to the timekeeping control system. **NOTE:** Can be connected to an external 32.768 kHz crystal or to an external clock source	
TXTAL_OUT	Analog	**Timekeeping Crystal Output—**can be connected to an external 32.768-kHz crystal or to an external clock source (which must be complementary to TXTAL_IN). **NOTE:** If an external 32.768 kHz clock source is used, TXTAL_OUT must be tied to GROUND	

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
HZ_CLK	Output	**Real-Time 1-Hz Clock**—real-time 1 Hz clock (after RTC trim adjustment)	
CLK_TOUT	Output	**Timekeeping Clock Output**—32.768kHz clock output - CLK_TOUT is a buffered version of the 32.768 kHz timekeeping clock. It is enabled when OSCC[TOUT_EN] bit is set. When enabled, this clock is output during S2/D3/C4, but it is always disabled in S3/D4/C4	
CLK_POUT	Output	**13-MHz Clock Output**—CLK_POUT is a buffered version of the 13 MHz processor clock. It is enabled when OSCC[POUT_EN] bit is set. When enabled, this clock is output during S0/D0/C0, S0/D0CS/C0, Start Of Day and S0/D2/C2 modes. For all other modes, CLK_POUT is disabled.	
VCTCXO_EN	Output	**13-MHz Clock Request**—VCTCXO_EN is an active high output indicating when the processor 13 MHz clock signal is required on PXTAL_IN. **NOTE:** VCTCXO_EN is normally connected to an external 13 MHz oscillator that supplies the 13 MHz clock to the processor. It is not needed if using a 13 MHz crystal connected to PXTAL_IN and PXTAL_OUT.	
CLK_EXT	Input	**External network Clock**—this input accepts an external network clock, up to 13 MHz, that may be selected as the timing reference for the SSP ports and the OS timer channels.	
CLK_48M	Output	**48-MHz Clock Output**—The CLK_48M is a 48 MHz clock out.	
Operating System Timer Signals			
EXT_SYNC1	Input	**External Sync 1**—this input provides a reset for any timer channels enabled to use it.	
EXT_SYNC0	Input	**External Sync 0**—this input provides a reset for any timer channels enabled to use it.	
CHOUT1	Output	**Timer Channel Output 1**—periodic clock output from timer channel 11	
CHOUT0	Output	**Timer Channel Output 0—**periodic clock output from timer channel 10	
Miscellaneous Signals			
PWR_EN	Output	**Power Enable for Core Power Supply**—this output, when deasserted, this signals the PMIC to turn off the low-voltage supplies (VCC_APPS, VCC_SRAM) because the system is entering S2/D3/C4 or S3/D4/C4 modes. VCC_SRAM may remain enabled during S2/D3/C4.	
SYS_EN	Output	**System Enable for System Peripheral Power Supply**—this output, when deasserted, this signals the PMIC to turn off the high-voltage supplies because the system is entering S3/D4/C4 mode.	
nBATT_FAULT	Input	**Main Battery Fault—**this input signals that the main battery is low or removed. Assertion causes the PXA3xx processor to enter S2/D3/C4 mode The PXA3xx processor does not recognize a wake-up event while this signal is asserted.	

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
nRESET	Input	**Reset**—this active-low, level-sensitive input is used to start the processor from the reset vector at address 0. Assertion causes the current instruction to terminate abnormally and causes a reset. When nRESET is driven high, the processor starts execution from address 0. nRESET must remain low until the power supply is stable and the internal 13 MHz oscillator has stabilized.	
nGPIO_RESET	Input	**GPIO Reset—**this active-low, level-sensitive input is used to start the processor from the reset vector at address 0 while preserving the contents of memory controller registers. Assertion causes the current instruction to terminate abnormally and causes a reset.	
nRESET_OUT	Output	**Reset Out—**an active-low output that signals the system that the SPMU is in any reset state (configurable for S2/D3/C4, S3/D4/C4 and for GPIO reset).	
PWR_SCL	Bidirectional	**Power Manager I^2C Clock**—the power manager I2C clock that connects to the external power management controller (PMIC)	3
PWR_SDA	Bidirectional	**Power Manager I^2C Data**—the power manager I2C data signal that connects to the external PMIC.	3
PWR_CAP<1:0>	Analog	**Power Capacitor<1:0>**—must be connected to external capacitors, which are used to achieve very low power in S2/D3/C4 mode. **NOTE:** Connect a ceramic 0.1uF capacitor between these pins	
PWR_OUT	Analog	**Internal Regulator Out—**must be decoupled with an external 0.1uF capacitor to GROUND	
EXT_WAKEUP0	Input	**External Wakeup 0—**Wake up signal	
EXT_WAKEUP1	Input	**External Wakeup 1** — Wake up signal	PXA320 only
ONE_WIRE	Bidirectional	**One Wire—**one wire smart battery interface data signal.	
JTAG and Test Signals			
nTRST	Input	**JTAG Test Reset—**IEEE 1194.1 test reset.	
TDI	Input	**JTAG Test Data Input—**data from the external JTAG controller is sent to the PXA3xx processor using this signal.	
TDO	Output	**JTAG Test Data Output**—data from the PXA3xx processor is returned to the external JTAG controller using this signal.	
TMS	Input	**JTAG Test Mode Select**—selects the test mode required from the external JTAG controller.	
TCK	Input	**JTAG Test Clock**—for all transfers on the JTAG test interface.	
TEST	Input	**Test Mode**—reserved for manufacturing test. Must be grounded for normal operation.	
TESTCLK	Input	**Test Clock**—reserved for manufacturing test. Must be grounded for normal operation.	

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
Power Supplies **NOTE:** Refer to the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification (EMTS)* for complete voltage specification details.			
VCC_BBATT	Power	**Backup Battery Supply—**connect to the backup battery supply. If a backup battery is not required, this pin may be connected to another system supply. **NOTE:** This pin is usually connected to the VCC_BBAT output of the PMIC.	4
VSS_BBATT	Power	**Ground for Backup Battery Supply**—connect to the backup battery ground reference.	4
VCC_APPS	Power	**Positive Supply for Internal Logic**—must be connected to a low voltage, adjustable system power supply.	4
VCC_SRAM	Power	**Positive Supply for Internal SRAM**—must be connected to a separate, variable supply.	4
VCC_MVT	Power	**Positive Supply for Internal Logic and I/O—**must be connected to a separate, fixed supply.	4
VCC_OSC13M	Power	**Positive Supply for 13-MHz Oscillator—**must be connected to a separate, fixed supply.	4
VSS_OSC13M	Power	**Ground Reference for 13-MHz Oscillator—**must be connected to the VSS ground reference.	4
VCC_BG	Power	**Positive Supply for Internal Bandgap Voltage Reference**—must be connected to a separate, fixed supply.	4
VSS_BG	Power	**Ground Reference for Internal Bandgap Voltage Reference—**must be connected to the ground reference for the VCC_BG supply.	4
VSS	Power	**Ground Reference for All Non-I/O Digital Supplies**	4
VCC_PLL	Power	**Positive Supply for PLLs and Oscillators**—must be connected to a separate, fixed supply.	4
VSS_PLL	Power	**Ground for PLL**—must be connected to the common ground plane on the PCB.	4
VCC_LCD	Power	**Positive Supply for LCD I/O**—must be connected to an external power supply	4
VSS_LCD	Power	**Ground for LCD I/O—**must be connected to the ground reference for the VCC_LCD supply.	4
VCC_CI	Power	**Positive Supply for Quick Capture Interface I/O—**must be connected to an external power supply	4
VSS_CI	Power	**Ground for Quick Capture Interface I/O—**must be connected to the ground reference for the VCC_CI supply.	4
VCC_USB	Power	**Positive Supply for USB Transceivers** (both host and client)—must be connected to an external power supply.	PXA30x and PXA32x only, 4
VSS_USB	Power	**Ground for USB Transceivers**—must be connected to the ground reference for the VCC_USB supply.	PXA30x and PXA32x only, 4

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes
VCC_BIAS	Power	**Positive Supply for USB Transceivers** (both host and client)—must be connected to an external power supply	PXA31x only, 4
VCC_ULPI	Power	**Positive Supply for ULPI power domain** — must be connected to an external power supply	PXA31x only, 4
VSS_ULPI	Power	**Ground for ULPI power domain** — must be connected to the ground reference for the VCC_ULPI supply	PXA31x only, 4
VCC_CARD1	Power	**Positive Supply for Card Interface #1**—must be connected to an external power supply for pins on the CARD1 power domain.	4
VSS_CARD1	Power	**Ground for Card Interface #1—**must be connected to the ground reference for the VCC_CARD1 supply.	4
VCC_CARD2	Power	**Positive Supply for Card Interface #2**—must be connected to an external power supply for pins on the CARD2 power domain.	4
VSS_CARD2	Power	**Ground for Card Interface #2**—must be connected to the ground reference for the VCC_CARD2 supply.	4
VCC_DF	Power	**Positive supply for Data Flash Interface—**must be connected to an external power supply.	4
VSS_DF	Power	**Ground for Data Flash Interface**—must be connected to the ground reference for the VCC_DF supply.	4
VCC_MSL	Power	**Positive supply for MSL voltage domain**—must be connected to an external power supply.	4
VSS_MSL	Power	**Ground for MSL**—must be connected to the ground reference for the VCC_MSL supply.	4
VCC_IO1	Power	**Digital I/O Supply—**must be connected to an external power supply.	4
VCC_IO3	Power	**Digital I/O Supply—**must be connected to an external power supply.	4
VCC_IO4	Power	**Digital I/O Supply—**must be connected to the common 1.8 V, 3.0 V or 3.3 V I/O supply.[1]	PXA32x only, 4
VCC_IO6	Power	**Digital I/O Supply—**must be connected to the common 1.8 V, 3.0 V or 3.3 V I/O supply.[1]	PXA32x only, 4
VSS_IOx	Power	**Ground for Primary Digital I/O Supply—**must be connected to the ground reference for the VCC_IO supply.	4
VCC_MEM	Power	**Positive Supply for EMPI—**must be connected to an external power supply for the Dynamic Memory Controller Interface.	4
VSS_MEM	Power	**Ground for EMPI**—must be connected to the ground reference for the regulator providing the VCC_MEM supply.	4
VCC_TSI	Power	**Positive Supply for Touchscreen Controller Interface—**must be connected to an external 3.3 V power supply.	PXA32x only, 4
VSS_TSI	Power	**Ground Reference for Touchscreen Controller Interface**—must be connected to the ground reference for the regulator providing the VCC_TSI supply.	PXA32x only, 4

Copyright © 2009 Marvell

Table 14: PXA3xx Processor Signal Descriptions (Continued)

Signal Name	Type	Signal Descriptions	Notes

NOTES:
1. The signals USBH2_P and USBC_P are muxed onto the same pin, USBOTG_P. Which signal is routed to USBOTG_P is determined by the configuration of UP2OCR[HXS]. Refer to The *Universal Serial Bus Client Controller* chapter of the *PXA3xx Processor Family Vol. IV: Developers Manual: Serial Controller Configuration* for more information.
2. The signals USBH2_N and USBC_N are muxed onto the same pin, USBOTG_N. Which signal is routed to USBOTG_N is determined by the configuration of UP2OCR[HXS]. Refer to The *Universal Serial Bus Client Controller* chapter of *PXA3xx Processor Family Vol. IV: Developers Manual: Serial Controller Configuration* for more information. This pin has an internal pullup resistor.
3. These PWR I^2C signals are intended as outputs to power controllers.
4. Positive supply for all CMOS I/O except the EMPI and data flash interfaces, USB differential transceiver pins, card interface pins, LCD pins, and Quick Capture Interface pins.

4.7 Pin Control Unit Overview

- The basic operation of each multi-function pin is controlled by a register. Each register controls a single multi-function pin.
- To program multiple pins (for example to select all the pins for a multi- pin function such as a UART), several registers need to be programmed in series.
- Each register operation can occur in a slightly unpredictable order, non-obvious re-ordering of completion being possible. A completion can be checked only by performing a Read operation. Software should not assume that pins are programmed in exactly the order of issue.
- Each register is assumed to be 32 bits wide and with word-aligned access.
- Each multi-function pin has a physical address in the overall processor-address map.
- A single multi-function pin can have up to eight alternate functions and a low-power mode (during S0/D1/C2, S0/D2/C2, and S2/D3/C4 modes) value.
- A multi-function pin always acts as an input (although the value may be ignored), which means that as long as power is applied to the multi-function pin it must have a good logic value present on the board. This output can, of course, be sourced from an external device or via the device driving itself.
- A multi-function pin can be programmed to take one of five outputs values: resistive pullup (nominal 100K), resistive pulldown (nominal 100K), driven high, driven low or high-impedance. Use three-state carefully to ensure that no floating node remains on the board at any time. These states can be used to form multiple output types (for example open collector, three-state drive etc.).

Note

The mechanism differs considerably from previous versions of this function (for example, on the Marvell PXA270 processor) where a control was provided to ignore the input of the pad (the RDH control).

4.7.1 Checking for Completion of a Multi-Function Pin Operation

A significant amount of queueing of operations is possible between software and the actual Write of the multi-function register. It is important that the software be programmed to detect when the last operation has completed and taken effect. This detection is performed by a Read operation to the address of the last operation that was performed.

4.7.2 Access to Nonexistent Registers or Pins

A Write access to a register within the multi-function pin control region that does not exist is not aborted and simply occurs with no effect. A Read access to such a register proceeds in an identical manner to an access to an existing register; however, the data returned is unpredictable.

4.7.3 Pin Control Unit Address Map

The address region for the multi-function pins is 0x40E1_0000–0x40E1_FFFF. Each location accessed is 32 bits in size and is 32-bit-aligned in memory. Accesses to other sizes and alignments in this region are not permitted and are undefined in effect. Table 17, “PXA30x Processor Pad Control Addresses”, Table 16, “PXA31x Processor Pad Control Addresses” and Table 15, “PXA32x Processor Pad Control Addresses” lists the addresses for each multi-function pin.

Table 15: PXA32x Processor Pad Control Addresses

Pin Name	Power Supply	Pad Control Address
GPIO0	VCC_MEM	0x40E1_0124
GPIO1	VCC_MEM	0x40E1_0128
GPIO2	VCC_DF	0x40E1_012C
GPIO3	VCC_DF	0x40E1_0130
GPIO4	VCC_DF	0x40E1_0134
nXCVREN	VCC_DF	0x40E1_0138
DF_CLE_NOE	VCC_DF	0x40E1_0204
DF_ALE_WE1	VCC_DF	0x40E1_0208
Reserved	VCC_DF	0x40E1_020C
DF_SCLK_E	VCC_DF	0x40E1_0210
nBE0	VCC_DF	0x40E1_0214
nBE1	VCC_DF	0x40E1_0218
DF_ALE_WE2	VCC_DF	0x40E1_021C
DF_INT_RnB	VCC_DF	0x40E1_0220
DF_nCS0	VCC_DF	0x40E1_0224
DF_nCS1	VCC_DF	0x40E1_0228
DF_nWE	VCC_DF	0x40E1_022C
DF_nRE	VCC_DF	0x40E1_0230
nLUA	VCC_DF	0x40E1_0234
nLLA	VCC_DF	0x40E1_0238
DF_ADDR0	VCC_DF	0x40E1_023C
DF_ADDR1	VCC_DF	0x40E1_0240
DF_ADDR2	VCC_DF	0x40E1_0244

Copyright © 2009 Marvell

Table 15: PXA32x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
DF_ADDR3	VCC_DF	0x40E1_0248
DF_IO0	VCC_DF	0x40E1_024C
DF_IO8	VCC_DF	0x40E1_0250
DF_IO1	VCC_DF	0x40E1_0254
DF_IO9	VCC_DF	0x40E1_0258
DF_IO2	VCC_DF	0x40E1_025C
DF_IO10	VCC_DF	0x40E1_0260
DF_IO3	VCC_DF	0x40E1_0264
DF_IO11	VCC_DF	0x40E1_0268
DF_IO4	VCC_DF	0x40E1_026C
DF_IO12	VCC_DF	0x40E1_0270
DF_IO5	VCC_DF	0x40E1_0274
DF_IO13	VCC_DF	0x40E1_0278
DF_IO6	VCC_DF	0x40E1_027C
DF_IO14	VCC_DF	0x40E1_0280
DF_IO7	VCC_DF	0x40E1_0284
DF_IO15	VCC_DF	0x40E1_0288
GPIO5	VCC_DF	0x40E1_028C
GPIO6	VCC_DF	0x40E1_0290
GPIO7	VCC_DF	0x40E1_0294
GPIO8	VCC_DF	0x40E1_0298
GPIO9	VCC_IO3	0x40E1_029C
GPIO11	VCC_IO3	0x40E1_02A0
GPIO12	VCC_IO3	0x40E1_02A4
GPIO13	VCC_IO3	0x40E1_02A8
GPIO14	VCC_IO3	0x40E1_02AC
GPIO15	VCC_IO3	0x40E1_02B0
GPIO16	VCC_IO3	0x40E1_02B4
GPIO17	VCC_IO3	0x40E1_02B8
GPIO18	VCC_CARD1	0x40E1_02BC
GPIO19	VCC_CARD1	0x40E1_02C0
GPIO20	VCC_CARD1	0x40E1_02C4

Copyright © 2009 Marvell

Table 15: PXA32x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO21	VCC_CARD1	0x40E1_02C8
GPIO22	VCC_CARD1	0x40E1_02CC
GPIO23	VCC_CARD1	0x40E1_02D0
GPIO24	VCC_CARD2	0x40E1_02D4
GPIO25	VCC_CARD2	0x40E1_02D8
GPIO26	VCC_CARD2	0x40E1_02DC
GPIO27	VCC_CARD2	0x40E1_0400
GPIO28	VCC_CARD2	0x40E1_0404
GPIO29	VCC_CARD2	0x40E1_0408
GPIO30	VCC_IO4	0x40E1_040C
GPIO31	VCC_IO4	0x40E1_0410
GPIO32	VCC_IO4	0x40E1_0414
GPIO33	VCC_IO4	0x40E1_0418
GPIO34	VCC_IO4	0x40E1_041C
GPIO35	VCC_IO4	0x40E1_0420
GPIO36	VCC_IO4	0x40E1_0424
GPIO37	VCC_IO4	0x40E1_0428
GPIO38	VCC_IO4	0x40E1_042C
GPIO39	VCC_IO4	0x40E1_0430
GPIO40	VCC_IO4	0x40E1_0434
GPIO41	VCC_IO4	0x40E1_0438
GPIO42	VCC_IO4	0x40E1_043C
GPIO43	VCC_IO4	0x40E1_0440
GPIO44	VCC_IO4	0x40E1_0444
GPIO45	VCC_IO4	0x40E1_0448
GPIO46	VCC_IO4	0x40E1_044C
GPIO47	VCC_IO4	0x40E1_0450
GPIO48	VCC_IO4	0x40E1_0454
GPIO10	VCC_IO4	0x40E1_0458
GPIO49	VCC_CI	0x40E1_045C
GPIO50	VCC_CI	0x40E1_0460
GPIO51	VCC_CI	0x40E1_0464

Copyright © 2009 Marvell

Table 15: PXA32x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO52	VCC_CI	0x40E1_0468
GPIO53	VCC_CI	0x40E1_046C
GPIO54	VCC_CI	0x40E1_0470
GPIO55	VCC_CI	0x40E1_0474
GPIO56	VCC_CI	0x40E1_0478
GPIO57	VCC_CI	0x40E1_047C
GPIO58	VCC_CI	0x40E1_0480
GPIO59	VCC_CI	0x40E1_0484
GPIO60	VCC_CI	0x40E1_0488
GPIO61	VCC_CI	0x40E1_048C
GPIO62	VCC_CI	0x40E1_0490
GPIO6_2	VCC_LCD	0x40E1_0494
GPIO7_2	VCC_LCD	0x40E1_0498
GPIO8_2	VCC_LCD	0x40E1_049C
GPIO9_2	VCC_LCD	0x40E1_04A0
GPIO10_2	VCC_LCD	0x40E1_04A4
GPIO11_2	VCC_LCD	0x40E1_04A8
GPIO12_2	VCC_LCD	0x40E1_04AC
GPIO13_2	VCC_LCD	0x40E1_04B0
GPIO63	VCC_LCD	0x40E1_04B4
GPIO64	VCC_LCD	0x40E1_04B8
GPIO65	VCC_LCD	0x40E1_04BC
GPIO66	VCC_LCD	0x40E1_04C0
GPIO67	VCC_LCD	0x40E1_04C4
GPIO68	VCC_LCD	0x40E1_04C8
GPIO69	VCC_LCD	0x40E1_04CC
GPIO70	VCC_LCD	0x40E1_04D0
GPIO71	VCC_LCD	0x40E1_04D4
GPIO72	VCC_LCD	0x40E1_04D8
GPIO73	VCC_LCD	0x40E1_04DC
GPIO14_2	VCC_LCD	0x40E1_04E0
GPIO15_2	VCC_LCD	0x40E1_04E4

Copyright © 2009 Marvell

Table 15: PXA32x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO16_2	VCC_LCD	0x40E1_04E8
GPIO17_2	VCC_LCD	0x40E1_04EC
GPIO74	VCC_LCD	0x40E1_04F0
GPIO75	VCC_MSL	0x40E1_04F4
GPIO76	VCC_MSL	0x40E1_04F8
GPIO77	VCC_MSL	0x40E1_04FC
GPIO78	VCC_MSL	0x40E1_0500
GPIO79	VCC_MSL	0x40E1_0504
GPIO80	VCC_MSL	0x40E1_0508
GPIO81	VCC_MSL	0x40E1_050C
GPIO82	VCC_MSL	0x40E1_0510
GPIO83	VCC_IO6	0x40E1_0514
GPIO84	VCC_IO6	0x40E1_0518
GPIO85	VCC_IO6	0x40E1_051C
GPIO86	VCC_IO6	0x40E1_0520
GPIO87	VCC_IO6	0x40E1_0524
GPIO88	VCC_IO6	0x40E1_0528
GPIO89	VCC_IO6	0x40E1_052C
GPIO90	VCC_IO6	0x40E1_0530
GPIO91	VCC_IO6	0x40E1_0534
GPIO92	VCC_IO6	0x40E1_0538
GPIO93	VCC_IO6	0x40E1_053C
GPIO94	VCC_IO6	0x40E1_0540
GPIO95	VCC_IO6	0x40E1_0544
GPIO96	VCC_IO6	0x40E1_0548
GPIO97	VCC_IO6	0x40E1_054C
GPIO98	VCC_IO6	0x40E1_0550
GPIO99	VCC_IO6	0x40E1_0600
GPIO100	VCC_IO6	0x40E1_0604
GPIO101	VCC_IO6	0x40E1_0608
GPIO102	VCC_IO6	0x40E1_060C
GPIO103	VCC_IO6	0x40E1_0610

Copyright © 2009 Marvell

Table 15: PXA32x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO104	VCC_IO6	0x40E1_0614
GPIO105	VCC_IO6	0x40E1_0618
GPIO106	VCC_IO6	0x40E1_061C
GPIO107	VCC_IO6	0x40E1_0620
GPIO108	VCC_IO6	0x40E1_0624
GPIO109	VCC_IO6	0x40E1_0628
GPIO110	VCC_IO6	0x40E1_062C
GPIO111	VCC_IO6	0x40E1_0630
GPIO112	VCC_IO6	0x40E1_0634
TSI_XP	VCC_TSI	0x40E1_0000
TSI_XM	VCC_TSI	0x40E1_0000
TSI_YP	VCC_TSI	0x40E_10000
TSI_YM	VCC_TSI	0x40E_10000
GPIO113	VCC_IO1	0x40E1_0638
GPIO114	VCC_IO1	0x40E1_063C
GPIO115	VCC_IO1	0x40E1_0640
GPIO116	VCC_IO1	0x40E1_0644
GPIO117	VCC_IO1	0x40E1_0648
GPIO118	VCC_IO1	0x40E1_064C
GPIO119	VCC_IO1	0x40E1_0650
GPIO120	VCC_IO1	0x40E1_0654
GPIO121	VCC_IO1	0x40E1_0658
GPIO122	VCC_IO1	0x40E1_065C
GPIO123	VCC_IO1	0x40E1_0660
GPIO124	VCC_IO1	0x40E1_0664
GPIO125	VCC_IO1	0x40E1_0668
GPIO126	VCC_IO1	0x40E1_066C
GPIO127	VCC_IO1	0x40E1_0670
GPIO0_2	VCC_IO1	0x40E1_0674
GPIO1_2	VCC_IO1	0x40E1_0678
GPIO2_2	VCC_IO1	0x40E1_067C
GPIO3_2	VCC_IO1	0x40E1_0680

Copyright © 2009 Marvell

Table 15: PXA32x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO4_2	VCC_IO1	0x40E1_0684
GPIO5_2	VCC_IO1	0x40E1_0688

Table 16: PXA31x Processor Pad Control Addresses

Pin Name	Power Supply	Pad Control Address
DF_IO0	VCC_DF	0x40E1_0220
DF_IO1	VCC_DF	0x40E1_0228
DF_IO2	VCC_DF	0x40E1_0230
DF_IO3	VCC_DF	0x40E1_0238
DF_IO4	VCC_DF	0x40E1_0258
DF_IO5	VCC_DF	0x40E1_0260
DF_IO6	VCC_DF	0x40E1_0268
DF_IO7	VCC_DF	0x40E1_0270
DF_IO8	VCC_DF	0x40E1_0224
DF_IO9	VCC_DF	0x40E1_022C
DF_IO10	VCC_DF	0x40E1_0234
DF_IO11	VCC_DF	0x40E1_023C
DF_IO12	VCC_DF	0x40E1_025C
DF_IO13	VCC_DF	0x40E1_0264
DF_IO14	VCC_DF	0x40E1_026C
DF_IO15	VCC_DF	0x40E1_0274
DF_CLE_nOE	VCC_DF	0x40E1_0240
DF_ALE_nWE	VCC_DF	0x40E1_020C
DF_SCLK_E	VCC_DF	0x40E1_0250
nCS0	VCC_DF	0x40E1_00C4
nCS1	VCC_DF	0x40E1_00C0
nBE0	VCC_DF	0x40E1_0204
nBE1	VCC_DF	0x40E1_0208
nLUA	VCC_DF	0x40E1_0244
nLLA	VCC_DF	0x40E1_0254
DF_ADDR0	VCC_DF	0x40E1_0210
DF_ADDR1	VCC_DF	0x40E1_0214

Copyright © 2009 Marvell

Table 16: PXA31x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
DF_ADDR2	VCC_DF	0x40E1_0218
DF_ADDR3	VCC_DF	0x40E1_021C
DF_INT_RnB	VCC_DF	0x40E1_00C8
DF_NCS0	VCC_DF	0x40E1_0248
DF_NCS1	VCC_DF	0x40E1_0278
DF_nWE	VCC_DF	0x40E1_00CC
DF_nRE	VCC_DF	0x40E1_0200
GPIO0	VCC_DF	0x40E1_00B4
GPIO1	VCC_DF	0x40E1_00B8
GPIO2	VCC_DF	0x40E1_00BC
GPIO3	VCC_CARD1	0x40E1_027C
GPIO4	VCC_CARD1	0x40E1_0280
GPIO5	VCC_CARD1	0x40E1_0284
GPIO6	VCC_CARD1	0x40E1_0288
GPIO7	VCC_CARD1	0x40E1_028C
GPIO8	VCC_CARD1	0x40E1_0290
GPIO9	VCC_CARD2	0x40E1_0294
GPIO10	VCC_CARD2	0x40E1_0298
GPIO11	VCC_CARD2	0x40E1_029C
GPIO12	VCC_CARD2	0x40E1_02A0
GPIO13	VCC_CARD2	0x40E1_02A4
GPIO14	VCC_CARD2	0x40E1_02A8
GPIO15	VCC_CARD2	0x40E1_02AC
GPIO16	VCC_CARD2	0x40E1_02B0
GPIO17	VCC_IO3	0x40E1_02B4
GPIO18	VCC_IO3	0x40E1_02B8
GPIO19	VCC_IO3	0x40E1_02BC
GPIO20	VCC_IO3	0x40E1_02C0
GPIO21	VCC_IO3	0x40E1_02C4
GPIO22	VCC_IO3	0x40E1_02C8
GPIO23	VCC_IO3	0x40E1_02CC
GPIO24	VCC_IO3	0x40E1_02D0
GPIO25	VCC_IO3	0x40E1_02D4

Copyright © 2009 Marvell

Table 16: PXA31x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO26	VCC_IO3	0x40E1_02D8
GPIO27	VCC_IO3	0x40E1_0400
GPIO28	VCC_IO3	0x40E1_0404
GPIO29	VCC_IO3	0x40E1_0408
ULPI_STP	VCC_ULPI	0x40E1_040C
ULPI_NXT	VCC_ULPI	0x40E1_0410
ULPI_DIR	VCC_ULPI	0x40E1_0414
GPIO30	VCC_ULPI	0x40E1_0418
GPIO31	VCC_ULPI	0x40E1_041C
GPIO32	VCC_ULPI	0x40E1_0420
GPIO33	VCC_ULPI	0x40E1_0424
GPIO34	VCC_ULPI	0x40E1_0428
GPIO35	VCC_ULPI	0x40E1_042C
GPIO36	VCC_ULPI	0x40E1_0430
GPIO37	VCC_ULPI	0x40E1_0434
GPIO38	VCC_ULPI	0x40E1_0438
GPIO39	VCC_CI	0x40E1_043C
GPIO40	VCC_CI	0x40E1_0440
GPIO41	VCC_CI	0x40E1_0444
GPIO42	VCC_CI	0x40E1_0448
GPIO43	VCC_CI	0x40E1_044C
GPIO44	VCC_CI	0x40E1_0450
GPIO45	VCC_CI	0x40E1_0454
GPIO46	VCC_CI	0x40E1_0458
GPIO47	VCC_CI	0x40E1_045C
GPIO48	VCC_CI	0x40E1_0460
GPIO49	VCC_CI	0x40E1_0464
GPIO50	VCC_CI	0x40E1_0468
GPIO51	VCC_CI	0x40E1_046C
GPIO52	VCC_CI	0x40E1_0470
GPIO53	VCC_LCD	0x40E1_0474
GPIO54	VCC_LCD	0x40E1_0478
GPIO55	VCC_LCD	0x40E1_047C

Copyright © 2009 Marvell

Table 16: PXA31x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO56	VCC_LCD	0x40E1_0480
GPIO57	VCC_LCD	0x40E1_0484
GPIO58	VCC_LCD	0x40E1_0488
GPIO59	VCC_LCD	0x40E1_048C
GPIO60	VCC_LCD	0x40E1_0490
GPIO61	VCC_LCD	0x40E1_0494
GPIO62	VCC_LCD	0x40E1_0498
GPIO63	VCC_LCD	0x40E1_049C
GPIO64	VCC_LCD	0x40E1_04A0
GPIO65	VCC_LCD	0x40E1_04A4
GPIO66	VCC_LCD	0x40E1_04A8
GPIO67	VCC_LCD	0x40E1_04AC
GPIO68	VCC_LCD	0x40E1_04B0
GPIO69	VCC_LCD	0x40E1_04B4
GPIO70	VCC_LCD	0x40E1_04B8
GPIO71	VCC_LCD	0x40E1_04BC
GPIO72	VCC_LCD	0x40E1_04C0
GPIO73	VCC_LCD	0x40E1_04C4
GPIO74	VCC_LCD	0x40E1_04C8
GPIO75	VCC_LCD	0x40E1_04CC
GPIO76	VCC_LCD	0x40E1_04D0
GPIO77	VCC_MSL	0x40E1_04D4
GPIO78	VCC_MSL	0x40E1_04D8
GPIO79	VCC_MSL	0x40E1_04DC
GPIO80	VCC_MSL	0x40E1_04E0
GPIO81	VCC_MSL	0x40E1_04E4
GPIO82	VCC_MSL	0x40E1_04E8
GPIO83	VCC_MSL	0x40E1_04EC
GPIO84	VCC_MSL	0x40E1_04F0
GPIO85	VCC_MSL	0x40E1_04F4
GPIO86	VCC_MSL	0x40E1_04F8
GPIO87	VCC_MSL	0x40E1_04FC
GPIO88	VCC_MSL	0x40E1_0500

Copyright © 2009 Marvell

Table 16: PXA31x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO89	VCC_MSL	0x40E1_0504
GPIO90	VCC_MSL	0x40E1_0508
GPIO91	VCC_IO1	0x40E1_050C
GPIO92	VCC_IO1	0x40E1_0510
GPIO93	VCC_IO1	0x40E1_0514
GPIO94	VCC_IO1	0x40E1_0518
GPIO95	VCC_IO1	0x40E1_051C
GPIO96	VCC_IO1	0x40E1_0520
GPIO97	VCC_IO1	0x40E1_0524
GPIO98	VCC_IO1	0x40E1_0528
GPIO99	VCC_IO1	0x40E1_0600
GPIO100	VCC_IO1	0x40E1_0604
GPIO101	VCC_IO1	0x40E1_0608
GPIO102	VCC_IO1	0x40E1_060C
GPIO103	VCC_IO1	0x40E1_0610
GPIO104	VCC_IO1	0x40E1_0614
GPIO105	VCC_IO1	0x40E1_0618
GPIO106	VCC_IO1	0x40E1_061C
GPIO107	VCC_IO1	0x40E1_0620
GPIO108	VCC_IO1	0x40E1_0624
GPIO109	VCC_IO1	0x40E1_0628
GPIO110	VCC_IO1	0x40E1_062C
GPIO111	VCC_IO1	0x40E1_0630
GPIO112	VCC_IO1	0x40E1_0634
GPIO113	VCC_IO1	0x40E1_0638
GPIO114	VCC_IO1	0x40E1_063C
GPIO115	VCC_IO1	0x40E1_0640
GPIO116	VCC_IO1	0x40E1_0644
GPIO117	VCC_IO1	0x40E1_0648
GPIO118	VCC_IO1	0x40E1_064C
GPIO119	VCC_IO1	0x40E1_0650
GPIO120	VCC_IO1	0x40E1_0654
GPIO121	VCC_IO1	0x40E1_0658

Copyright © 2009 Marvell

Table 16: PXA31x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO122	VCC_IO1	0x40E1_065C
GPIO123	VCC_IO1	0x40E1_0660
GPIO124	VCC_IO1	0x40E1_0664
GPIO125	VCC_IO1	0x40E1_0668
GPIO126	VCC_IO1	0x40E1_066C
GPIO127	VCC_IO1	0x40E1_0670
GPIO0_2	VCC_IO1	0x40E1_0674
GPIO1_2	VCC_IO1	0x40E1_0678
GPIO2_2	VCC_IO3	0x40E1_02DC
GPIO3_2	VCC_IO3	0x40E1_02E0
GPIO4_2	VCC_IO3	0x40E1_02E4
GPIO5_2	VCC_IO3	0x40E1_02E8
GPIO6_2	VCC_IO3	0x40E1_02EC
GPIO7_2	VCC_IO1	0x40E1_052C
GPIO8_2	VCC_IO1	0x40E1_0530
GPIO9_2	VCC_IO1	0x40E1_0534
GPIO10_2	VCC_IO1	0x40E1_0538

Table 17: PXA30x Processor Pad Control Addresses

Pin Name	Power Supply	Pad Control Address
DF_IO0	VCC_DF	0x40E1_0220
DF_IO1	VCC_DF	0x40E1_0228
DF_IO2	VCC_DF	0x40E1_0230
DF_IO3	VCC_DF	0x40E1_0238
DF_IO4	VCC_DF	0x40E1_0258
DF_IO5	VCC_DF	0x40E1_0260
DF_IO6	VCC_DF	0x40E1_0268
DF_IO7	VCC_DF	0x40E1_0270
DF_IO8	VCC_DF	0x40E1_0224
DF_IO9	VCC_DF	0x40E1_022C
DF_IO10	VCC_DF	0x40E1_0234
DF_IO11	VCC_DF	0x40E1_023C

Copyright © 2009 Marvell

Table 17: PXA30x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
DF_IO12	VCC_DF	0x40E1_025C
DF_IO13	VCC_DF	0x40E1_0264
DF_IO14	VCC_DF	0x40E1_026C
DF_IO15	VCC_DF	0x40E1_0274
DF_CLE_nOE	VCC_DF	0x40E1_0240
DF_ALE_nWE	VCC_DF	0x40E1_020C
DF_SCLK_E	VCC_DF	0x40E1_0250
nCS0	VCC_DF	0x40E1_00C4
nCS1	VCC_DF	0x40E1_00C0
nBE0	VCC_DF	0x40E1_0204
nBE1	VCC_DF	0x40E1_0208
nLUA	VCC_DF	0x40E1_0244
nLLA	VCC_DF	0x40E1_0254
DF_ADDR0	VCC_DF	0x40E1_0210
DF_ADDR1	VCC_DF	0x40E1_0214
DF_ADDR2	VCC_DF	0x40E1_0218
DF_ADDR3	VCC_DF	0x40E1_021C
DF_INT_RnB	VCC_DF	0x40E1_00C8
DF_NCS0	VCC_DF	0x40E1_0248
DF_NCS1	VCC_DF	0x40E1_0278
DF_nWE	VCC_DF	0x40E1_00CC
DF_nRE	VCC_DF	0x40E1_0200
GPIO0	VCC_DF	0x40E1_00B4
GPIO1	VCC_DF	0x40E1_00B8
GPIO2	VCC_DF	0x40E1_00BC
GPIO3	VCC_CARD1	0x40E1_027C
GPIO4	VCC_CARD1	0x40E1_0280
GPIO5	VCC_CARD1	0x40E1_0284
GPIO6	VCC_CARD1	0x40E1_0288
GPIO7	VCC_CARD1	0x40E1_028C
GPIO8	VCC_CARD1	0x40E1_0290
GPIO9	VCC_CARD2	0x40E1_0294
GPIO10	VCC_CARD2	0x40E1_0298

Copyright © 2009 Marvell

Table 17: PXA30x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO11	VCC_CARD2	0x40E1_029C
GPIO12	VCC_CARD2	0x40E1_02A0
GPIO13	VCC_CARD2	0x40E1_02A4
GPIO14	VCC_CARD2	0x40E1_02A8
GPIO15	VCC_CARD2	0x40E1_02AC
GPIO16	VCC_CARD2	0x40E1_02B0
GPIO17	VCC_IO3	0x40E1_02B4
GPIO18	VCC_IO3	0x40E1_02B8
GPIO19	VCC_IO3	0x40E1_02BC
GPIO20	VCC_IO3	0x40E1_02C0
GPIO21	VCC_IO3	0x40E1_02C4
GPIO22	VCC_IO3	0x40E1_02C8
GPIO23	VCC_IO3	0x40E1_02CC
GPIO24	VCC_IO3	0x40E1_02D0
GPIO25	VCC_IO3	0x40E1_02D4
GPIO26	VCC_IO3	0x40E1_02D8
GPIO27	VCC_IO3	0x40E1_0400
GPIO28	VCC_IO3	0x40E1_0404
GPIO29	VCC_IO3	0x40E1_0408
GPIO30	VCC_IO3	0x40E1_040C
GPIO31	VCC_IO3	0x40E1_0410
GPIO32	VCC_IO3	0x40E1_0414
GPIO33	VCC_IO3	0x40E1_0418
GPIO34	VCC_IO3	0x40E1_041C
GPIO35	VCC_IO3	0x40E1_0420
GPIO36	VCC_IO3	0x40E1_0424
GPIO37	VCC_IO3	0x40E1_0428
GPIO38	VCC_IO3	0x40E1_042C
GPIO39	VCC_CI	0x40E1_0430
GPIO40	VCC_CI	0x40E1_0434
GPIO41	VCC_CI	0x40E1_0438
GPIO42	VCC_CI	0x40E1_043C
GPIO43	VCC_CI	0x40E1_0440

Copyright © 2009 Marvell

Table 17: PXA30x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO44	VCC_CI	0x40E1_0444
GPIO45	VCC_CI	0x40E1_0448
GPIO46	VCC_CI	0x40E1_044C
GPIO47	VCC_CI	0x40E1_0450
GPIO48	VCC_CI	0x40E1_0454
GPIO49	VCC_CI	0x40E1_0458
GPIO50	VCC_CI	0x40E1_045C
GPIO51	VCC_CI	0x40E1_0460
GPIO52	VCC_CI	0x40E1_0464
GPIO53	VCC_LCD	0x40E1_0468
GPIO54	VCC_LCD	0x40E1_046C
GPIO55	VCC_LCD	0x40E1_0470
GPIO56	VCC_LCD	0x40E1_0474
GPIO57	VCC_LCD	0x40E1_0478
GPIO58	VCC_LCD	0x40E1_047C
GPIO59	VCC_LCD	0x40E1_0480
GPIO60	VCC_LCD	0x40E1_0484
GPIO61	VCC_LCD	0x40E1_0488
GPIO62	VCC_LCD	0x40E1_048C
GPIO63	VCC_LCD	0x40E1_0490
GPIO64	VCC_LCD	0x40E1_0494
GPIO65	VCC_LCD	0x40E1_0498
GPIO66	VCC_LCD	0x40E1_049C
GPIO67	VCC_LCD	0x40E1_04A0
GPIO68	VCC_LCD	0x40E1_04A4
GPIO69	VCC_LCD	0x40E1_04A8
GPIO70	VCC_LCD	0x40E1_04AC
GPIO71	VCC_LCD	0x40E1_04B0
GPIO72	VCC_LCD	0x40E1_04B4
GPIO73	VCC_LCD	0x40E1_04B8
GPIO74	VCC_LCD	0x40E1_04BC
GPIO75	VCC_LCD	0x40E1_04C0
GPIO76	VCC_LCD	0x40E1_04C4

Copyright © 2009 Marvell

Table 17: PXA30x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO77	VCC_MSL	0x40E1_04C8
GPIO78	VCC_MSL	0x40E1_04CC
GPIO79	VCC_MSL	0x40E1_04D0
GPIO80	VCC_MSL	0x40E1_04D4
GPIO81	VCC_MSL	0x40E1_04D8
GPIO82	VCC_MSL	0x40E1_04DC
GPIO83	VCC_MSL	0x40E1_04E0
GPIO84	VCC_MSL	0x40E1_04E4
GPIO85	VCC_MSL	0x40E1_04E8
GPIO86	VCC_MSL	0x40E1_04EC
GPIO87	VCC_MSL	0x40E1_04F0
GPIO88	VCC_MSL	0x40E1_04F4
GPIO89	VCC_MSL	0x40E1_04F8
GPIO90	VCC_MSL	0x40E1_04FC
GPIO91	VCC_IO1	0x40E1_0500
GPIO92	VCC_IO1	0x40E1_0504
GPIO93	VCC_IO1	0x40E1_0508
GPIO94	VCC_IO1	0x40E1_050C
GPIO95	VCC_IO1	0x40E1_0510
GPIO96	VCC_IO1	0x40E1_0514
GPIO97	VCC_IO1	0x40E1_0518
GPIO98	VCC_IO1	0x40E1_051C
GPIO99	VCC_IO1	0x40E1_0600
GPIO100	VCC_IO1	0x40E1_0604
GPIO101	VCC_IO1	0x40E1_0608
GPIO102	VCC_IO1	0x40E1_060C
GPIO103	VCC_IO1	0x40E1_0610
GPIO104	VCC_IO1	0x40E1_0614
GPIO105	VCC_IO1	0x40E1_0618
GPIO106	VCC_IO1	0x40E1_061C
GPIO107	VCC_IO1	0x40E1_0620
GPIO108	VCC_IO1	0x40E1_0624
GPIO109	VCC_IO1	0x40E1_0628

Copyright © 2009 Marvell

Table 17: PXA30x Processor Pad Control Addresses (Continued)

Pin Name	Power Supply	Pad Control Address
GPIO110	VCC_IO1	0x40E1_062C
GPIO111	VCC_IO1	0x40E1_0630
GPIO112	VCC_IO1	0x40E1_0634
GPIO113	VCC_IO1	0x40E1_0638
GPIO114	VCC_IO1	0x40E1_063C
GPIO115	VCC_IO1	0x40E1_0640
GPIO116	VCC_IO1	0x40E1_0644
GPIO117	VCC_IO1	0x40E1_0648
GPIO118	VCC_IO1	0x40E1_064C
GPIO119	VCC_IO1	0x40E1_0650
GPIO120	VCC_IO1	0x40E1_0654
GPIO121	VCC_IO1	0x40E1_0658
GPIO122	VCC_IO1	0x40E1_065C
GPIO123	VCC_IO1	0x40E1_0660
GPIO124	VCC_IO1	0x40E1_0664
GPIO125	VCC_IO1	0x40E1_0668
GPIO126	VCC_IO1	0x40E1_066C
GPIO127	VCC_IO1	0x40E1_0670
GPIO0_2	VCC_IO1	0x40E1_0674
GPIO1_2	VCC_IO1	0x40E1_0678
GPIO2_2	VCC_IO3	0x40E1_02DC
GPIO3_2	VCC_IO3	0x40E1_02E0
GPIO4_2	VCC_IO3	0x40E1_02E4
GPIO5_2	VCC_IO3	0x40E1_02E8
GPIO6_2	VCC_IO3	0x40E1_02EC

4.8 Multi-Function Pin Description

The multi-function pin has a relatively simple function. Several logical functions are connected via a multiplexer to a single pin and one is selected via MFPRxx[AF_SEL]. Each logical function consists of some or all of the following:

- Data-out signal
- Output-enable signal
- Input signal
- Pullup control
- Pulldown control

Copyright © 2009 Marvell

An alternate function can be an output, an input, or a bidirectional function. Any function that is not selected is disconnected logically from the pins both for input AND output (the output signals being ignored and the input signals being forced to a default state as they are fed to the internal device).

Note

The pin direction is controlled by hardware in all cases. In effect, the configuration of the alternate function connects the pin to its driving logic and that logic is responsible for determining whether the pin is an input or an output. For a GPIO function, the direction is controlled by the GPIO function (that is, the direction register).

For output, a pin can be set to drive a 0, 1, pull high or pull low, or to maintain a three-state condition. Each option can be selected dynamically, allowing functions such as drive 0 and pull high.

Multiple pins can be allocated to the same function, the output data and enable signals being duplicated, and the input signals being OR'ed together (or NANDed if active low) to form a single function. Software is responsible for avoiding situations where multiple pins are assigned to the same function (or if this situation is preferred, to manage the configurations).

Example

A single pin can be defined as having two alternate functions, in this example, a UART data input and an SSP data-out signal.

The SSP pin can drive 0 or 1, or three-state. It has no data input. The UART function has no data output (therefore, it is in effect a three-state) but does have a data input. SSP is Function 0 and UART is Function 1. Function 0 has the data-out and the output-enable connections connected to the SSP, but the data-input connections are not used. Function 1 has the output-enable connection set to a 0 (forcing the output to be a three-state), and the data output unconnected. The data input is connected to the UART.

When Function 0 is selected, the output pin follows the SSP signals. The data-enable output from the UART is ignored and the data input is forced to be a 0 (the standard default case). When Function 1 is selected, the SSP outputs are ignored, the output is three-state, and the input connected to the UART. This configuration raises the issue of potentially floating inputs if some device is not always driving the input.

4.9 Edge-Detect Operation

Each multi-function pin can detect a rising or falling edge on the input (when enabled, this monitors the pin at all times so an output transition would trigger it). The detection is controlled via the edge_clear (which enables and disables the function), and the edge_rise (if enabled, causes an event on rising edge), and the edge_fall (if enabled causes an event on falling edge) register bits.

This purely combinational function detects an event based on the edge and not on a clock sampling a pulse (although a small minimum pulse width is required). For example, for rising edge detect, an event is detected when the signal is 0 initially, and then later it is 1.

Both edge detectors can be enabled, thus producing an event whenever the signal changed state (even briefly).

Note

This edge-detect function provides a wake-up event in all power modes to the device and is NOT connected to the edge-detection function of the GPIO controller.

Copyright © 2009 Marvell

4.10 Low-Power Mode Operation

Low-power modes are defined as the state where the internal logical function is not operational but the pin must have a valid value (typically the default case). Such is the case during S0/D1/C2, S0/D2/C2, and S2/D3/C4 modes of operation (in S3/D4/C4 mode the actual pins are powered off and no function is required).The multi-function pins in this example are controlled by a combination of bits programmed into the MFPRxx registers for each pin. See Table 18 and Table 21, "MFPR Bit Definitions" for more information.

In most situations, the S0/D1/C2, S0/D2/C2 and S2/D3/C4 low-power modes behavior of a pin is as a logical output (driving either 1 or 0), a pull high or low, or as a logical input. This behavior can be controlled via the SLEEP_DATA bit (determining whether the output should be 1 or 0) and the SLEEP_OE bit (determining whether the pin should be an output or an input). The pullup and pulldown functions are enabled only via the state of the PULLUP_EN and PULLDOWN_EN bits (the PULL_SEL signal is overridden to 1 automatically when in the low power modes). Thus, any possible state can be programmed in advance for S0/D1/C2, S0/D2/C2 and S2/D3/C4 low-power modes and then activated upon entering the low-power mode.

Table 18: Low-Power Mode States

SLEEP_OE_N	SLEEP_DATA	PULLUP_EN	PULLDOWN_EN	Output Value
0	0	0	0	0
0	1	0	0	1
1	X	1	0	Pull high
1	X	0	1	Pull low
1	X	0	0	Three-state
NOTE: Output of three-state is identical to a simple input case. In this example, the output driver does not impact the input state. However, the board must always drive this pin while in this mode to avoid any floating- node issues. This requirement is the same as previous generations of processors with the change that this must be done in ALL states (except for S3/D4/C4) of the device as there is no provision for disabling the input path.				

When SLEEP_SEL = 0 and the processor is in the S0/D0/C0 state, the multi-function pins value is determined by the functional unit associated with the multi-function pins. If the processor transitions to S0/D1/C2, S0/D2/C2, or S2/D3/C4 low power modes, the multi-function pin value is determined by the bit settings shown in Table 18, "Low-Power Mode States". Transitioning from S0/D1/C2 or S0/D2/C2 to S0/D0/C0 state results in the multi-function pin value being determined by the functional unit associated with the multi-function pins. Transitioning from S2/D3/C4 to S0/D0/C0 results in the multi-function pin value being determined by the functional unit associated with the multi-function pin; in this example, the functional unit associated with the multi-function pin is reset.

When SLEEP_SEL = 1, the multi-function pin value depends on the ASCR[RDH] bit. In S0/D0/C0 mode with ASCR[RDH] cleared, the multi-function pin value is determined by the functional unit associated with the multi-function pin. A transition to S0/D1/C2, S0/D2/C2 or S2/D3/C4 low-power modes results in hardware automatically setting the RDH bit and the multi-function pin value is determined by the bit settings shown in Table 18, "Low-Power Mode States". Transitioning from S0/D1/C2 or S0/D2/C2 states to S0/D0/C0 results in the RDH bit being cleared by hardware and the multi-function pin value being determined by the functional unit associated with the multi-function pin. Transitioning from S2/D3/C4 to S0/D0/C0 results in the multi-function pin holding the value shown in Table 18, "Low-Power Mode States" until the RDH bit is cleared by software; this holding of the S2/D3/C4 multi-function pin value until the RDH bit is cleared is known as *delayed-release functionality*.

Copyright © 2009 Marvell

See Table 19, "SLEEP_SEL and RDH Multi-function Pin State Summary" for a summary of multi-function pin states dependencies upon SLEEP_SEL, ASCR[RDH], and low power modes.

Table 19: SLEEP_SEL and RDH Multi-function Pin State Summary

SLEEP_SEL	Power Mode	ASCR[RDH]	Multi-function Pin State
0	S0/D0/C0	X	Normal run mode, multi-function pin state determined by functional unit associated with multi-function pin
0	S0/D0/C0 transition to S0/D1/C2, S0/D2/C2, or S2/D3/C4	X	Table 18
0	S0/D1/C2 or S0/D2/C2 transition to S0/D0/C0	X	Multi-function pin state determined by functional unit associated with multi-function pin
0	S2/D3/C4 transition to S0/D0/C0	X	Multi-function pin state determined by functional unit associated with multi-function pin; functional unit is reset
1	S0/D0/C0	0	Normal run mode, multi-function pin state determined by functional unit associated with multi-function pin
1	S0/D0/C0 transition to S0/D1/C2, S0/D2/C2, or S2/D3/C4	1 - set by hardware	Table 18
1	S0/D1/C2 or S0/D2/C2 transition to S0/D0/C0	0 - cleared by hardware	multi-function pins state determined by functional unit associated with multi-function pins
1	S2/D3/C4 transition to S0/D0/C0	1	Table 18

Setting the SLEEP_SEL bit to employ the delayed-release functionality while exiting the S2/D3/C4 low- power mode depends on the use of the multi-function pin. There are instances where the delayed-release functionality is almost always used, for example, for multi-function used for USB transceiver pins to avoid errors on the USB transceiver pins during S2/D3/C4 low-power mode exit. There are also instances where the delayed release functionality may or may not be used, for example, for multi-function pins used for SSP ports, depending upon the characteristics of the SSP devices. Some multi-function pins (DFI pins, for example) would never use this functionality.

4.11 Register Description

Each Multi-Function Pin Register (MFPR) is identical in structure and definitions.

4.11.1 Register Summary

Table 21 shows the register associated with controlling the multi-function pins.

Table 20: Pins and Control Register Summary

Address	Description	Page
0x40E1_0000 - 0x40E1_FFFF	Multi-Function Pin Registers (MFPR)	129

Copyright © 2009 Marvell

4.11.2 Multi-Function Pin Registers (MFPR)

Table 21, "MFPR Bit Definitions" defines the settings for each multi-function pin. Each multi-function pin register is identical in structure and definition.

Table 21: MFPR Bit Definitions

Physical Address **MFPR xx**
0x40E1_0000 - 0x40E1_FFFF

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																PULL_SEL	PULLUP_EN	PULLDOWN_EN	DRIVE			SLEEP_SEL	SLEEP_DATA	SLEEP_OE_N	EDGE_CLEAR	EDGE_FALL_EN	EDGE_RISE_EN	Reserved	AF_SEL		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	See Table Note 1 below									1	0	0	?	0	0	0

Bits	Access	Name	Description
31:16	—	Reserved	Reserved
15	Read/Write	PULL_SEL	**PULL State Select** This bit is used to select between two sets of controls for the pull-up and pull-down functionality. 0 = The pullup and down resistors are controlled by the selected alternate function for the pin. 1 = The pullup and pulldown resistors are controlled by the PULLUP_EN and PULLDOWN_EN bits in this register, overriding the function indicated by the selected alternate function **NOTE:** During low-power modes this bit is overridden to a 1 and control is thus via the PULLxx_EN bits. In these low-power states, the PULL_SEL bit behaves as a 1 although the register value is not changed.
14	Read/Write	PULLUP_EN	**PULLUP Enable** 0 = The internal pullup resistor of the pin is disabled. 1 = The internal pullup resistor of the pin is enabled. **NOTE:** This controls the output function only when PULL_SEL is set (0b1). Refer to Table 22, "Pullup/Pulldown State Table" for pullup/pulldown states
13	Read/Write	PULLDOWN_EN	**PULLDOWN Enable** 0 = The internal pulldown resistor of the pin is disabled. 1 = The internal pulldown resistor of the pin is enabled. **NOTE:** This controls the output function only when PULL_SEL is set (0b1). Refer to Table 22, "Pullup/Pulldown State Table" for pullup/pulldown states

Table 21: MFPR Bit Definitions (Continued)

Physical Address 0x40E1_0000 - 0x40E1_FFFF **MFPR xx**

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																PULL_SEL	PULLUP_EN	PULLDOWN_EN	DRIVE			SLEEP_SEL	SLEEP_DATA	SLEEP_OE_N	EDGE_CLEAR	EDGE_FALL_EN	EDGE_RISE_EN	Reserved	AF_SEL		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	See Table Note 1 below									1	0	0	?	0	0	0

Bits	Access	Name	Description
12:10	Read/Write	DRIVE	**Drive strength and slew rate for this pin** (in functional mode when driving a 0 or a 1): 0b000 = Fast 1 mA 0b001 = Fast 2 mA 0b010 = Fast 3 mA 0b011 = Fast 4 mA 0b100 = Slow 6 mA 0b101 = Fast 6 mA 0b110 = Slow 10 mA 0b111 = Fast 10 mA
9	Read/Write	SLEEP_SEL	**Low Power Selection** Selection control for S0/D1/C2, S0/D2/C2 and S2/D3/C4 low power mode entry signal (see Section 4.10 for more information).
8	Read/Write	SLEEP_DATA	**Low Power Data Selection** The data on the pin sent during S0/D1/C2, S0/D2/C2 and S2/D3/C4 low power modes (see Section 4.10 for more information).
7	Read/Write	SLEEP_OE_N	**Low power mode output enable** Enables outputs during S0/D1/C2, S0/D2/C2 and S2/D3/C4 low power modes 0 = Enable as an output during low power modes. 1 = Disable output during low power modes. See Table 18 for possible states of this signal depending upon values of the PULLUP_EN and PULLDOWN_EN bits.(see Section 4.10 for more information).
6	Read/Write	EDGE_CLEAR	**Edge Detect Enable**[2] This is an enable for EDGE_FALL_EN and EDGE_RISE_EN control bits. 0 = The edge detection logic is enabled and ready to detect an edge. 1 = The edge detection logic is disabled and no edge will be detected.
5	Read/Write	EDGE_FALL_EN	**Falling Edge Enable**[2] In order to detect a falling edge on this pin (see Section 4.9 for more information) this bit must be set to 1 and the EDGE_CLEAR bit must be set to 0. 0 = Do not detect a falling edge. 1 = Detect a falling edge.

Copyright © 2009 Marvell

Table 21: MFPR Bit Definitions (Continued)

Physical Address **MFPR xx**
0x40E1_0000 - 0x40E1_FFFF

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																PULL_SEL	PULLUP_EN	PULLDOWN_EN	DRIVE			SLEEP_SEL	SLEEP_DATA	SLEEP_OE_N	EDGE_CLEAR	EDGE_FALL_EN	EDGE_RISE_EN	Reserved	AF_SEL		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	See Table Note 1 below									1	0	0	?	0	0	0

Bits	Access	Name	Description
4	Read/Write	EDGE_RISE_EN	**Rising Edge Enable**[2] To detect a rising edge on this pin (see Section 4.9 for more information) this bit must be set to 1 and the EDGE_CLEAR bit must be set to 0. 0 = Do not detect a rising edge. 1 = Detect a rising edge
3	—	Reserved	Reserved
2:0	Read/Write	AF_SEL	**Alternate function selection** This selects between the 8 possible alternate functions for the pin. Alternate function 0 is always the reset case. 0b000 = Primary function at reset (Alternate function 0) 0b001 = Alternate function 1 0b010 = Alternate function 2 0b011 = Alternate function 3 0b100 = Alternate function 4 0b101 = Alternate function 5 0b110 = Alternate function 6 0b111 = Alternate function 7

NOTE:
(1) The reset value for these bits are defined on a pin-by-pin basis. Do not rely on the reset value of these bits. These bits must be configured by software to the required settings.
(2) Bits 4:6 are present only when a pin has been defined as potentially waking up on an edge. If the device is not configured in this manner, the registers are reserved and writing to them has no effect. Reading these bits on pins with no wakeup will return an indeterminate value.

Table 22: Pullup/Pulldown State Table

PULLUP_EN	PULLDOWN_EN	Pull Up/Pull Down State
0	0	Float - No pullup or pulldown
0	1	Pulldown enabled
1	0	Pullup enabled
1	1	Float - No pullup or pulldown

Copyright © 2009 Marvell

Copyright © 2009 Marvell

5 General-Purpose I/O Unit

The General-Purpose I/O module (GPIO) provides 128 general-purpose input/output GPIO ports for use in generating and capturing application-specific input and output. All ports are brought out of the processor through the alternate function multiplexing. When programmed as an input, a GPIO port can also serve as an interrupt source. At the assertion of all resets, all 128 ports are configured as inputs and remain inputs until they are configured either by the boot process or by user software.

Note

GPIO is the physical I/O ball and multi-function pin (MFP) is the alternate function logic.

The direction of the GPIO ports is controlled by writing to the GPIO Pin Direction register (GPDRx).

When the GPIO pin is programmed as an output, the port is set by writing to the GPIO Pin Output Set Register (GPSRx) and cleared by writing to the GPIO Pin Output Clear Register (GPCRx). The Set and Clear registers can be written regardless of whether the port is configured as an input or an output. The programmed output state takes effect when the port is reconfigured as an output.

When the GPIO pin is programmed as an input, it can be configured to detect a rising edge, falling edge, or both through the GPIO Rising-Edge Detect Enable Register (GRERx) and the GPIO Falling-Edge Detect Enable Register (GFERx). The state of the edge-detect can be read through the GPIO Edge Detect Status Register (GEDRx). These edge-detects can be programmed to generate interrupts.

The value of each GPIO port is read through the GPIO Pin-Level Register (GPLRx). This register can be read at any time and can confirm the port state for both input and output configurations.

In the PXA25x and PXA27x processors, the alternate functions (peripherals) are routed through the GPIO controller to the GPIO pins. For the PXA3xx processors, the alternate functions (peripherals and GPIO) are multiplexed at the pad, so the term “GPIO” refers to both the GPIO peripheral controller (port) and the GPIO pin.

A GPIO pin can have one of several alternate functions. The MFPR registers are used to select the function for a particular GPIO pin. One of the alternate functions that can be programmed for each pin is General-Purpose I/O (GPIO) port. Refer to the “Pins and Pin Control” chapter for details on configuring the pins as GPIOs. Knowledge of the operation of both the GPIO control logic and the multi-function pin control logic is necessary to understand the operation of the GPIOs. A system can use the GPIO functions internally and not actually connect to a physical pin.

The registers discussed here are reset during non-state-retentive Power modes, and the interrupt events are not produced during low-power modes and therefore cannot cause wakeup events.

Copyright © 2009 Marvell

5.1 PXA3xx Processor Differences

Table 23 shows the GPIO Controller differences among the PXA32x, PXA31x and PXA30x processors.

Table 23: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
second instantiation pins	GPIO<6:0>_2	GPIO<10:0>_2	GPIO<17:0>_2

5.2 Features

The general features of the GPIOs include the following:

- As inputs they can be programmed to generate an interrupt from a rising edge, a falling edge, or both.
- As inputs the values can be read individually.
- As outputs they can be cleared or set individually.

5.3 Signals

This section describes the internal signals that are inputs or outputs of the GPIO controller (see Table 24.

Table 24: GPIO Controller Interface Signals Summary

Internal Name	Type	Description	Notes
GPIO<127:0>	In/Out	General-Purpose IO Ports. The GPIOs are programmable for: • Inputs or outputs • Interrupts or wake-up sources • Rising or falling edge detection	
GPIO<6_0>_2	In/Out	General-Purpose IO Ports, second instantiation. The same signals as GPIO<6:0>, but they are given separate names because they are configured on separate multi-function pins. The recommended configuration is for both multi-function pin instantiations of these GPIOs *not* to be configured simultaneously for GPIO functionality. The preferred configuration for dual instantiation GPIO usage (that is, using both GPIO<6:0> and GPIO<6:0>_2 functions) is to configure one multi-function pin as a GPIO and the second multi-function pin instantiation for an alternate function other than a GPIO function. Configuring GPIO<6:0> to be present on two separate multi-function pins is not recommended. However, when GPIO<6:0> is configured as an output and both multi-function pin instantiations are programmed for GPIO function, the primary and secondary instantiation of the multi-function pins are both outputs from the common GGPIO<6:0> signal. However, when GPIO<6:0> is an input, the inputs from both multi-function pin instantiations are ORed together before the result is sent to the internal GPIO<6:0> input logic.	PXA30x Only

Copyright © 2009 Marvell

Table 24: GPIO Controller Interface Signals Summary (Continued)

Internal Name	Type	Description	Notes
GPIO<10:0>_2	In/Out	General-Purpose IO Ports, second instantiation. The same signals as GPIO<10:0>, but they are given separate names because they are configured on separate multi-function pins. The recommended configuration is for both multi-function pin instantiations of these GPIOs *not* to be configured simultaneously for GPIO functionality. The preferred configuration for dual instantiation GPIO usage (that is, using both GPIO<10:0> and GPIO<10:0>_2 functions) is to configure one multi-function pin as a GPIO and the second multi-function pin instantiation for an alternate function other than a GPIO function. Configuring GPIO<10:0> to be present on two separate multi-function pins is not recommended. However, when GPIO<10:0> is configured as an output and both multi-function pin instantiations are programmed for GPIO function, the primary and secondary instantiation of the multi-function pins are both outputs from the common GPIO<10:0> signal. However, when GPIO<10:0> is an input, the inputs from both multi-function pin instantiations are ORed together before the result is sent to the internal GPIO<10:0> input logic.	PXA31x Only
GPIO<17:0>_2	In/Out	General-Purpose IO Ports, second instantiation. The same signals as GPIO<17:0>, but they are given separate names because they are configured on separate multi-function pins. The recommended configuration is for both multi-function pin instantiations of these GPIOs *not* to be configured simultaneously for GPIO functionality. The preferred configuration for dual instantiation GPIO usage (that is, using both GPIO<17:0> and GPIO<17:0>_2 functions) is to configure one multi-function pin as a GPIO and the second multi-function pin instantiation for an alternate function other than a GPIO function. Configuring GPIO<17:0> to be present on two separate multi-function pins is not recommended. However, when GPIO<17:0> is configured as an output and both multi-function pin instantiations are programmed for GPIO function, the primary and secondary instantiation of the multi-function pins are both outputs from the common GPIO<17:0> signal. However, when GPIO<17:0> is an input, the inputs from both multi-function pin instantiations are ORed together before the result is sent to the internal GPIO<17:0> input logic.	PXA32x Only

5.4 Register Descriptions

There are fifty-two 32-bit registers within the GPIO control block, six distinct register functions, and four instances of each of the 13 registers to serve 128 GPIOs. The various functions of the 13 registers corresponding to each GPIO port are described in the following list:

- 4 registers monitor port state: GPLRx (read only).
- 12 registers control port direction:
  - 4 GPDRx (read/write)
  - 4 bit-wise Set registers GSDRx (write only)—Modifies value of GPDR
  - 4 bit-wise Clear registers GCDRx (write only)—Modifies value of GPDR

- 8 registers control port state:
  - GPSRx (write only)—Sets GPIO output port
  - GPCRx (write only)—Clears GPIO output port
- 12 registers indicate whether a rising edge should be detected:
  - 4 GRERx - (read/write)
  - 4 bit-wise Set registers GSRERx (write only)—Modifies value of GRER
  - 4 bit-wise Clear registers GCRERx (write only)—Modifies value of GRER
- 12 registers indicate whether a falling edge should be detected:
  - 4 GFERx - (read/write)
  - 4 bit-wise Set registers GSFERx (write only)—Modifies value of GFER
  - 4 bit-wise Clear registers GCFERx (write only)—Modifies value of GFER
- 4 registers indicate when specified edge types have been detected on ports
  - (GEDRx)

Note

The Pin Direction register (GPDRx) is initialized by reset, thus assuring that at reset, all ports appear as GPIO input ports. All other GPIO registers are also initialized to 0x0 at reset.

5.4.1 Register Descriptions

Table 25 summarizes the registers associated with the GPIO block and the physical addresses used to access them. The remainder of this section discusses each of these registers in detail. For easy reference, the summary tables include the page number of the detailed description for each register. On the PXA32x processor, GPIO56 and GPIO<62:59> are missing because by default these signals are alternate functions rather than GPIO signals. On the PXA30x processor and PXA31x processor, GPIO functionality for GPIO46 and GPIO<52:49> are on alternate functions other than the default alternate function (Alternate Function 0). For details on these alternate functions, refer to the Pins and Control chapter.

Table 25: GPIO Register Summary

Address	GPIO Pin Assignments	Description	Page
0x40E0_0000	GPIO<31:0>	GPIO Pin-Level Register 0 (GPLR0)	page 139
0x40E0_0004	GPIO<63:32>	GPIO Pin-Level Register 1(GPLR1)	page 139
0x40E0_0008	GPIO<95:64>	GPIO Pin-Level Register 2(GPLR2)	page 139
0x40E0_000C	GPIO<31:0>	GPIO Pin Direction Register 0 (GPDR0)	page 140
0x40E0_0010	GPIO<63:32>	GPIO Pin Direction Register 1 (GPDR1)	page 140
0x40E0_0014	GPIO<95:64>	GPIO Pin Direction Register 2 (GPDR2)	page 140
0x40E0_0018	GPIO<31:0>	GPIO Pin Output Set Register 0 (GPSR0)	page 143
0x40E0_001C	GPIO<63:32>	GPIO Pin Output Set Register 1 (GPSR1)	page 143

Copyright © 2009 Marvell

Table 25: GPIO Register Summary (Continued)

Address	GPIO Pin Assignments	Description	Page
0x40E0_0020	GPIO<95:64>	GPIO Pin Output Set Register 2 (GPSR2)	page 143
0x40E0_0024	GPIO<31:0>	GPIO Pin Output Clear Register 0 (GPCR0)	page 143
0x40E0_0028	GPIO<63:32>	GPIO Pin Output Clear Register 1 (GPCR1)	page 143
0x40E0_002C	GPIO<95:64>	GPIO Pin Output Clear Register 2 (GPCR2)	page 143
0x40E0_0030	GPIO<31:0>	GPIO Rising-Edge Detect-Enable Register 0 (GRER0)	page 145
0x40E0_0034	GPIO<63:32>	GPIO Rising-Edge Detect-Enable Register 1 (GRER1)	page 145
0x40E0_0038	GPIO<95:64>	GPIO Rising-Edge Detect-Enable Register 2 (GRER2)	page 145
0x40E0_003C	GPIO<31:0>	GPIO Falling-Edge Detect-Enable Register 0 (GFER0)	page 147
0x40E0_0040	GPIO<63:32>	GPIO Falling-Edge Detect-Enable Register 1 (GFER1)	page 147
0x40E0_0044	GPIO<95:64>	GPIO Falling-Edge Detect-Enable Register 2 (GFER2)	page 147
0x40E0_0048	GPIO<31:0>	GPIO Edge Detect Status Register 0 (GEDR0)	page 150
0x40E0_004C	GPIO<63:32>	GPIO Edge Detect Status Register 1 (GEDR1)	page 150
0x40E0_0050	GPIO<95:64>	GPIO Edge Detect Status Register 2 (GEDR02)	page 150
0x40E0_0054–0x40E0_00FC	—	Reserved	
0x40E0_0100	GPIO<127:96>	GPIO Pin-Level Register 3 (GPLR3)	page 139
0x40E0_0104–0x40E0_0108	—	Reserved	
0x40E0_010C	GPIO<127:96>	GPIO Pin Direction Register 3 (GPDR3)	page 140
0x40E0_0110–0x40E0_0114	—	Reserved	
0x40E0_0118	GPIO<127:96>	GPIO Pin Output Set Register 3 (GPSR3)	page 143
0x40E0_011C–0x40E0_0120	—	Reserved	
0x40E0_0124	GPIO<127:96>	GPIO Pin Output Clear Registers 3 (GPCR3)	page 143
0x40E0_0128–0x40E0_012C	—	Reserved	
0x40E0_0130	GPIO<127:96>	GPIO Rising-Edge Detect-Enable Register 3 (GRER3)	page 145
0x40E0_0134–0x40E0_0138	—	Reserved	
0x40E0_013C	GPIO<127:96>	GPIO Falling-Edge Detect-Enable Register 3 (GFER3)	page 147

Copyright © 2009 Marvell

Table 25: GPIO Register Summary (Continued)

Address	GPIO Pin Assignments	Description	Page
0x40E0_0140–0x40E0_0144	—	Reserved	
0x40E0_0148	GPIO<127:96>	GPIO Edge Detect Status Register 3 (GEDR3)	page 150
0x40E0_014C–0x40E0_03FC		Reserved	
0x40E0_0400	GPIO<31:0>	GPIO Pin Bit-Wise Set Direction Register 0 (GSDR0)	page 141
0x40E0_0404	GPIO<63:32>	GPIO Pin Bit-Wise Set Direction Register 1 (GSDR1)	page 141
0x40E0_0408	GPIO<95:64>	GPIO Pin Bit-Wise Set Direction Register 2 (GSDR2)	page 141
0x40E0_040C	GPIO<127:96>	GPIO Pin Bit-Wise Set Direction Register 3 (GSDR3)	page 141
0x40E0_0410–0x40E0_041C	—	Reserved	
0x40E0_0420	GPIO<31:0>	GPIO Pin Bit-Wise Clear Direction Register 0 (GCDR0)	page 142
0x40E0_0424	GPIO<63:32>	GPIO Pin Bit-Wise Clear Direction Register 1 (GCDR1)	page 142
0x40E0_0428	GPIO<95:64>	GPIO Pin Bit-Wise Clear Direction Register 2 (GCDR2)	page 142
0x40E0_042C	GPIO<127:96>	GPIO Pin Bit-Wise Clear Direction Register 3 (GCDR3)	page 142
0x40E0_0430–0x40E0_043C	—	Reserved	
0x40E0_0440	GPIO<31:0>	GPIO Bit-Wise Set Rising-Edge Register 0 (GSRER0)	page 146
0x40E0_0444	GPIO<63:32>	GPIO Bit-Wise Set Rising-Edge Register 1 (GSRER1)	page 146
0x40E0_0448	GPIO<95:64>	GPIO Bit-Wise Set Rising-Edge Register 2 (GSRER2)	page 146
0x40E0_044C	GPIO<127:96>	GPIO Bit-Wise Set Rising-Edge Register 3 (GSRER3)	page 146
0x40E0_0450–0x40E0_045C		Reserved	
0x40E0_0460	GPIO<31:0>	GPIO Bit-wise Clear Rising-Edge (GCRER0) Detect-Enable Register 0	page 145
0x40E0_0464	GPIO<63:32>	GPIO Bit-wise Clear Rising-Edge (GCRER1) Detect-Enable Register 1	page 145
0x40E0_0468	GPIO<95:64>	GPIO Bit-wise Clear Rising-Edge (GCRER2) Detect-Enable Register 2	page 145
0x40E0_046C	GPIO<127:96>	GPIO Bit-wise Clear Rising-Edge (GCRER3) Detect-Enable Register 3	page 145
0x40E0_0470–0x40E0_047C	—	Reserved	

Copyright © 2009 Marvell

Table 25: GPIO Register Summary (Continued)

Address	GPIO Pin Assignments	Description	Page
0x40E0_0480	GPIO<31:0>	GPIO Bit-Wise Set Falling-Edge Register 0 (GSFER0)	page 148
0x40E0_0484	GPIO<63:32>	GPIO Bit-Wise Set Falling-Edge Register 1 (GSFER1)	page 148
0x40E0_0488	GPIO<95:64>	GPIO Bit-Wise Set Falling-Edge Register 2 (GSFER2)	page 148
0x40E0_048C	GPIO<127:96>	GPIO Bit-Wise Set Falling-Edge Register 3 (GSFER3)	page 148
0x40E0_0490–0x40E0_049C	—	Reserved	
0x40E0_04A0	GPIO<31:0>	GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 0 (GCFER0)	page 147
0x40E0_04A4	GPIO<63:32>	GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 1 (GCFER1)	page 147
0x40E0_04A8	GPIO<95:64>	GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 2 (GCFER2)	page 147
0x40E0_04AC	GPIO<127:96>	GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 3 (GCFER3)	page 147
0x40E0_04B0–0x40EF_FFFF	—	Reserved	

5.4.2 GPIO Pin-Level Registers (GPLRx)

GPLRx, shown in Table 26, make visible the state of each of the GPIO ports. Each bit corresponds to the port number:

5.4.2.1 GPIO Pin-Level Register 0 (GPLR0)

- Corresponds to GPIO<31:0>

5.4.2.2 GPIO Pin-Level Register 1(GPLR1)

- Corresponds to GPIO<63:32>

5.4.2.3 GPIO Pin-Level Register 2(GPLR2)

- Corresponds to GPIO<95:64>

5.4.2.4 GPIO Pin-Level Register 3 (GPLR3)

- Corresponds to GPIO<127:96>

These read-only registers show the current value of a particular port (regardless of the programmed port direction). All the GPIO pins default to inputs when the processor comes out of reset. The value of the input is determined by the default pullup/pulldown state and whether external chips are driving them. Table 26 shows the locations of the 32 port-level bits within the GPLR0 registers.

Table 26: GPLRx Bit Definitions

Physical Address		
0x40E0_0000	GPLR0	GPIO Controller
0x40E0_0004	GPLR1	
0x40E0_0008	GPLR2	
0x40E0_0100	GPLR3	

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

Bits	Access	Name	Description
n	R	PL{n}	GPIO port level n (where n = 0 through 31) 0 = Port state is low 1 = Port state is high

5.4.3 GPIO Pin Direction Registers (GPDRx)

GPDRx, shown in Table 27, allow users to control port direction. The GPDR registers contain one direction control bit for each GPIO pin.

5.4.3.1 GPIO Pin Direction Register 0 (GPDR0)

- Corresponds to GPIO<31:0>

5.4.3.2 GPIO Pin Direction Register 1 (GPDR1)

- Corresponds to GPIO<63:32>

5.4.3.3 GPIO Pin Direction Register 2 (GPDR2)

- Corresponds to GPIO<95:64>

5.4.3.4 GPIO Pin Direction Register 3 (GPDR3)

- Corresponds to GPIO<127:96>

If a direction bit is set, the GPIO pin is an output. If it is cleared, it is an input. A pair of set/clear registers (GSDRx and GCDRx) is also provided to enable the setting and clearing of individual bits of the GPDRx register. Table 27 shows the location of each port direction bit in the GPIO Pin Direction Register, GPDR0.

Copyright © 2009 Marvell

Table 27: GPDRx Bit Definitions

Physical Address		GPIO Controller
0x40E0_000C	GPDR0	
0x40E0_0010	GPDR1	
0x40E0_0014	GPDR2	
0x40E0_010C	GPDR3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PD[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	R/W	PD{n}	**GPIO port direction n** (where n = 0 through 31) 0 = Port configured as an input 1 = Port configured as an output

5.4.4 GPIO Pin Bit-Wise Set Direction Registers (GSDRx)

GSDRx, shown in Table 28, allow users to control port direction. These registers contain one direction control bit for each GPIO pin:

5.4.4.1 GPIO Pin Bit-Wise Set Direction Register 0 (GSDR0)

- Corresponds to GPIO<31:0>

5.4.4.2 GPIO Pin Bit-Wise Set Direction Register 1 (GSDR1)

- Corresponds to GPIO<63:32>

5.4.4.3 GPIO Pin Bit-Wise Set Direction Register 2 (GSDR2)

- Corresponds to GPIO<95:64>

5.4.4.4 GPIO Pin Bit-Wise Set Direction Register 3 (GSDR3)

- Corresponds to GPIO<127:96>

If a direction bit is set, the corresponding bit in GPDRx is set and the GPIO function is configured as an output. If it is cleared, no change in the GPIO functionality or the GPDRx register occurs. Table 28 shows the location of each port direction bit in GSDR0.

Table 28: GSDRx Bit Definitions

Physical Address		GPIO Controller
0x40E0_0400	GSDR0	
0x40E0_0404	GSDR1	
0x40E0_0408	GSDR2	
0x40E0_040C	GSDR3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PD[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	PD{n}	Set GPIO port direction n (where n = 0 through 31) 0 = GPDR bit not affected. 1 = GPDR bit is set and GPIOx function is set to OUTPUT.

5.4.5 GPIO Pin Bit-Wise Clear Direction Registers (GCDRx)

GCDRx, shown in Table 29, are used for clearing GPDR bits, turning the corresponding GPIO signals into inputs:

5.4.5.1 GPIO Pin Bit-Wise Clear Direction Register 0 (GCDR0)

- Corresponds to GPIO<31:0>

5.4.5.2 GPIO Pin Bit-Wise Clear Direction Register 1 (GCDR1)

- Corresponds to GPIO<63:32>

5.4.5.3 GPIO Pin Bit-Wise Clear Direction Register 2 (GCDR2)

- Corresponds to GPIO<95:64>

5.4.5.4 GPIO Pin Bit-Wise Clear Direction Register 3 (GCDR3)

- Corresponds to GPIO<127:96>

If a direction bit is set, the corresponding bit in GPDR is cleared and the GPIO function is configured as an input. If it is cleared, no change in the GPIO functionality or the GPDR register occurs. Table 29 shows the location of each port direction bit in GCDR0.

Copyright © 2009 Marvell

Table 29: GCDRx Bit Definitions

Physical Address		
0x40E0_0420	GCDR0	GPIO Controller
0x40E0_0424	GCDR1	
0x40E0_0428	GCDR2	
0x40E0_042C	GCDR3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PD[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	PD{n}	Set GPIO port direction n (where n = 0 through 31) 0 = GPDR bit is not affected. 1 = GPDR bit is cleared and a GPIO n function is set to INPUT.

5.4.6 GPIO Pin Output Set Registers (GPSRx) and Pin Output Clear Registers (GPCRx)

GPSRx and GPCRx gives users control of the state of the port when a GPIO is configured as an output. An output port is set by writing a 1 to its corresponding bit in the GPSR. To clear an output port, a 1 is written to the corresponding bit within the GPCRx (write-only registers; reads return unpredictable values). Clearing any GPSRx or GPCRx bit has no effect on the port state. Setting a GPSRx or GPCRx bit corresponding to a port that is configured as an input takes effect only after the port is configured as an output. The GPSRx and GPCRx registers contain one output set and one output clear control bit, respectively, for each GPIO pin:

5.4.6.1 GPIO Pin Output Set Register 0 (GPSR0)

- Corresponds to GPIO<31:0>

5.4.6.2 GPIO Pin Output Set Register 1 (GPSR1)

- Corresponds to GPIO<63:32>

5.4.6.3 GPIO Pin Output Set Register 2 (GPSR2)

- Corresponds to GPIO<95:64>

5.4.6.4 GPIO Pin Output Set Register 3 (GPSR3)

- Corresponds to GPIO<127:96>

5.4.6.5 GPIO Pin Output Clear Register 0 (GPCR0)

- Corresponds to GPIO<31:0>

5.4.6.6 GPIO Pin Output Clear Register 1 (GPCR1)

- Corresponds to GPIO<63:32>

5.4.6.7 GPIO Pin Output Clear Register 2 (GPCR2)

- Corresponds to GPIO<95:64>

5.4.6.8 GPIO Pin Output Clear Registers 3 (GPCR3)

- Corresponds to GPIO<127:96>

If a 1 is written to both registers at the same location, the last Write takes effect (that is, an output port cannot be both set and cleared at the same time). Table 30 shows the locations of the GPSRx bits. Table 31 shows the locations of the GPCRx bits.

Table 30: GPSRx Bit Definitions

Physical Address		
0x40E0_0018	GPSR0	GPIO Controller
0x40E0_001C	GPSR1	
0x40E0_0020	GPSR2	
0x40E0_0118	GPSR3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PS[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	PS{n}	GPIO output port set n (where n = 0 through 31) 0 = Port level unaffected. 1 = If the port is configured as an output, set the port level logic high.

Table 31: GPCRx Bit Definitions

Physical Address		
0x40E0_0024	GPCR0	GPIO Controller
0x40E0_0028	GPCR1	
0x40E0_002C	GPCR2	
0x40E0_0124	GPCR3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PC[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	PC{n}	GPIO output port clear n (where n = 0 through 31) 0 = Port level unaffected. 1 = If the port is configured as an output, clear the port level logic low.

Copyright © 2009 Marvell

5.4.7 GPIO Rising-Edge Detect-Enable Registers (GRERx)

GRERx, shown in Table 32, and GFERx (shown in Table 35) select the type of transition on a GPIO port that causes a bit within the GEDRx to be set. Each GPIO can be programmed to detect a rising edge, falling edge, or either transition on a port. Refer to the *PXA3xx Processor Electrical, Mechanical, and Thermal Specification* for the specific product to find the minimum pulse width to guarantee a detection. A status bit is set when an edge is detected that matches the type of edge programmed for the port.

For a given GPIO port, its corresponding GRERx bit is set to cause a GEDRx status bit to be set when the port transitions from Logic Level low to Logic Level high. Similarly, System Busx is used to set the corresponding GEDRx status bit during a transition from Logic Level high to Logic Level low. When the corresponding bits are set in both registers, either a falling- or a rising-edge transition causes the corresponding GEDRx status bit to be set. The GRERx registers contain one rising-edge detect control bit for each GPIO pin:

5.4.7.1 GPIO Rising-Edge Detect-Enable Register 0 (GRER0)

- Corresponds to GPIO<31:0>

5.4.7.2 GPIO Rising-Edge Detect-Enable Register 1 (GRER1)

- Corresponds to GPIO<63:32>

5.4.7.3 GPIO Rising-Edge Detect-Enable Register 2 (GRER2)

- Corresponds to GPIO<95:64>

5.4.7.4 GPIO Rising-Edge Detect-Enable Register 3 (GRER3)

- Corresponds to GPIO<127:96>

Table 32 shows the rising-edge enable bit locations corresponding to all 32 ports of GRER0. A pair of set/clear registers are also provided to enable the setting and clearing of individual bits of the GRERx registers.

Table 32: GRERx Bit Definitions

Physical Address		
0x40E0_0030	GRER0	GPIO Controller
0x40E0_0034	GRER1	
0x40E0_0038	GRER2	
0x40E0_0130	GRER3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RE[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	R/W	RE{n}	GPIO port n rising-edge detect enable (where n = 0 through 31) 0 = Disable rising-edge detect enable. 1 = Set the corresponding GEDR status bit when a rising edge is detected on the GPIO port.

5.4.8 GPIO Bit-Wise Set Rising-Edge (GSRERx) and GPIO Bit-wise Clear Rising-Edge (GCRERx) Detect-Enable Registers

GSRERx and GCRERx, shown in Table 33 and Table 34, directly allow users to change the programming of GRERx. The GSRERx registers contain one detection-level set control bit for each GPIO pin. If a bit is set, the corresponding bit in GRERx is set and the GPIO function is configured to cause a GEDRx status bit to be set when the port transitions from Logic Level zero (0) to Logic Level one (1). If the bit is cleared, no change in the GPIO functionality or GRERx occurs.

The GCRERx registers contain one detection-level clear control bit for each GPIO pin. If a bit is set, the corresponding bit in GRERx is cleared and the GPIO function is configured to *not* cause a GEDR status bit to be set when the port transitions from Logic Level zero (0) to Logic Level one (1). If the bit is cleared, no change in the GPIO functionality or the GRERx register occurs.

5.4.8.1 GPIO Bit-Wise Set Rising-Edge Register 0 (GSRER0)

- Corresponds to GPIO<31:0>

5.4.8.2 GPIO Bit-Wise Set Rising-Edge Register 1 (GSRER1)

- Corresponds to GPIO<63:32>

5.4.8.3 GPIO Bit-Wise Set Rising-Edge Register 2 (GSRER2)

- Corresponds to GPIO<95:64>

5.4.8.4 GPIO Bit-Wise Set Rising-Edge Register 3 (GSRER3)

- Corresponds to GPIO<127:96>

5.4.8.5 GPIO Bit-wise Clear Rising-Edge (GCRER0) Detect-Enable Register 0

- Corresponds to GPIO<31:0>

5.4.8.6 GPIO Bit-wise Clear Rising-Edge (GCRER1) Detect-Enable Register 1

- Corresponds to GPIO<63:32>

5.4.8.7 GPIO Bit-wise Clear Rising-Edge (GCRER2) Detect-Enable Register 2

- Corresponds to GPIO<95:64>

5.4.8.8 GPIO Bit-wise Clear Rising-Edge (GCRER3) Detect-Enable Register 3

- Corresponds to GPIO<127:96>

Copyright © 2009 Marvell

Table 33: GSRERx Bit Definitions

Physical Address		
0x40E0_0440	GSRER0	GPIO Controller
0x40E0_0444	GSRER1	
0x40E0_0448	GSRER2	
0x40E0_044C	GSRER3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SRE[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	SRE{n}	Set GPIO Rising Edge detect enable n (where n = 0 through 31) 0 = GRER bit not affected. 1 = GRER bit is set.

Table 34: GCRERx Bit Definitions

Physical Addresses		
0x40E0_0460	GCRER0	GPIO Controller
0x40E0_0464	GCRER1	
0x40E0_0468	GCRER2	
0x40E0_046C	GCRER3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CRE[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	CRE{n}	Clear GPIO Rising Edge detect enable n (where n = 0 through 31) 0 = GRER bit not affected. 1 = GRER bit is cleared.

5.4.9 GPIO Falling-Edge Detect-Enable Registers (GFERx)

Each GPIO can be programmed to detect a rising edge, falling edge, or either transition on a port. Refer to the *PXA3xx Processor Electrical, Mechanical, and Thermal Specification* for the minimum pulse width to guarantee a detection. A status bit is set when an edge is detected that matches the type of edge programmed for the port.

The GPIO Rising-Edge and Falling-Edge Detect-Enable registers (GRERx and GFERx, respectively) select the type of transition on a GPIO port that causes a bit within the GPIO Edge-Detect Enable Status register (GEDRx) to be set. For a given GPIO port, its corresponding GFERx bit is set to cause a GEDRx status bit to be set when the port transitions from logic-level high to logic-level low. Likewise, GRERx is used to set the corresponding GEDRx status bit when a transition from logic-level low to logic-level high occurs. When the corresponding bits are set in both

registers, either a falling- or a rising-edge transition causes the corresponding GEDRx status bit to be set.

The GFERx registers contain one falling-edge detect control bit for each of the GPIO pins.

5.4.9.1 GPIO Falling-Edge Detect-Enable Register 0 (GFER0)

- Corresponds to GPIO<31:0>

5.4.9.2 GPIO Falling-Edge Detect-Enable Register 1 (GFER1)

- Corresponds to GPIO<63:32>

5.4.9.3 GPIO Falling-Edge Detect-Enable Register 2 (GFER2)

- Corresponds to GPIO<95:64>

5.4.9.4 GPIO Falling-Edge Detect-Enable Register 3 (GFER3)

- Corresponds to GPIO<127:96>

Table 35 shows the falling-edge enable bit locations corresponding to all 32 pins of GFERx. A pair of set/clear registers is also provided to enable the setting and clearing of individual bits of the GFERx registers.

Table 35: GFERx Bit Definitions

Physical Address		
0x40E0_003C	GFER0	GPIO Controller
0x40E0_0040	GFER1	
0x40E0_0044	GFER2	
0x40E0_013C	GFER3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FE[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	R/W	FE{n}	GPIO port falling-edge detect enable n (where n = 0 through 31) 0 = No falling-edge detect enable. 1 = Set the corresponding GEDR status bit when a falling edge is detected on the GPIO port.

5.4.10 GPIO Bit-Wise Set Falling-Edge (GSFERx) and GPIO Bit-wise Clear Falling-Edge (GCFERx) Detect-Enable Registers

GSFERx and GCFERx, shown in Table 36 and Table 37, directly allow users to change the programming of GFERx. The GSFERx contain one detection-level set control bit for each GPIO pin. If a bit is set, the corresponding bit in GFERx is set and the GPIO function is configured to cause a GEDRx status bit to be set when the port transitions from logic-level one (1) to logic-level zero (0). If the bit is cleared, no change in the GPIO functionality or the GFERx register occurs.

The GCFERx registers contain one detection-level clear control bit for each GPIO pin. If a bit is set, the corresponding bit in GFERx is cleared and the GPIO function is configured to *not* cause a

Copyright © 2009 Marvell

GEDRx status bit to be set when the port transitions from logic-level one (1) to logic-level zero (0) If the bit is cleared, no change occurs in the GPIO functionality or the GFERx register.

5.4.10.1 GPIO Bit-Wise Set Falling-Edge Register 0 (GSFER0)

- Correspond to GPIO<31:0>

5.4.10.2 GPIO Bit-Wise Set Falling-Edge Register 1 (GSFER1)

- Correspond to GPIO<63:32>

5.4.10.3 GPIO Bit-Wise Set Falling-Edge Register 2 (GSFER2)

- Correspond to GPIO<95:64>

5.4.10.4 GPIO Bit-Wise Set Falling-Edge Register 3 (GSFER3)

- Correspond to GPIO<127:96>

5.4.10.5 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 0 (GCFER0)

- Correspond to GPIO<31:0>

5.4.10.6 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 1 (GCFER1)

- Correspond to GPIO<63:32>

5.4.10.7 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 2 (GCFER2)

- Correspond to GPIO<95:64>

5.4.10.8 GPIO Bit-wise Clear Falling-Edge Detect-Enable Register 3 (GCFER3)

- Correspond to GPIO<127:96>

Table 36: GSFERx Bit Definitions

Physical Address		
0x40E0_0480	GSFER0	GPIO Controller
0x40E0_0484	GSFER1	
0x40E0_0488	GSFER2	
0x40E0_048C	GSFER3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SFE[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	SFE{n}	Set GPIO Falling Edge detect enable n (where n = 0 through 31) 0 = GFER bit not affected 1 = GFER bit is set

Table 37: GCFERx Bit Definitions

Physical Address		
0x40E0_04A0	GCFER0	GPIO Controller
0x40E0_04A4	GCFER1	
0x40E0_04A8	GCFER2	
0x40E0_04AC	GCFER3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CFE[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	W	CFE{n}	Clear GPIO Falling Edge detect enable n (where n = 0 through 31) 0 = GFER bit not affected. 1 = GFER bit is cleared.

5.4.11 GPIO Edge Detect Status Registers (GEDRx)

GEDRx, shown in Table 38, contains a status bit for each GPIO pin. When an edge-detect occurs on a port that matches the type of edge programmed in the GRERx and/or GFERx registers, the corresponding status bit is set in GEDRx. When a GEDRx bit is set, the CPU must clear it. GEDRx status bits are cleared by writing a 1 to them. Writing a 0 has no effect. Each edge-detect that sets the corresponding GEDRx status bit for GPIO pins 0 – 127 can trigger an interrupt request. Pins 2 – 127 form a group that can cause one interrupt request to be triggered when any one of the GEDR status bits 2 –127 is set. GPIO pins 0 and 1 each cause their own, independent first-level interrupt:

5.4.11.1 GPIO Edge Detect Status Register 0 (GEDR0)

- Correspond to GPIO<31:0>

5.4.11.2 GPIO Edge Detect Status Register 1 (GEDR1)

- Correspond to GPIO<63:32>

5.4.11.3 GPIO Edge Detect Status Register 2 (GEDR02)

- Correspond to GPIO<95:64>

5.4.11.4 GPIO Edge Detect Status Register 3 (GEDR3)

- Correspond to GPIO<127:96>

Copyright © 2009 Marvell

Table 38: GEDRx Bit Definitions

Physical Addresses		
0x40E0_0048	GEDR0	GPIO Controller
0x40E0_004C	GEDR1	
0x40E0_0050	GEDR2	
0x40E0_0148	GEDR3	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ED[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
n	R/Write 1 to clear	ED{n}	GPIO edge detect status n (where n = 0 through 31) 0 = No edge detect on the port as specified in GRER and/or GFER. 1 = Edge detect on the port as specified in GRER and/or GFER.

Copyright © 2009 Marvell

6 Clock Controllers and Power Management

The processor has a modular system-on-chip architecture that partitions power management and clock generation functions into a master controller known as the services unit and a slave controller known as the *application subsystem unit*. The services (Services Clock Control Unit (SCCU) and Services Power Management Unit (SPMU)) and application subsystem (Application Subsytem Clock Control Unit (ACCU) and Application Subsystem Power Management Unit (APMU)) units work together to manage overall processor power consumption by manipulating clocks and voltage domains. Power management software determines the most efficient power mode and operating point for the highest performance and lowest power consumption.

6.1 PXA3xx Processor Differences

Table 39 shows the clocks and power management differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 39: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
S0/D1/C2	Supported	Supported	Not Supported
S0/D2/C2	Supported	Supported	Not Supported
DDR SDRAM clock frequencies[1]	15 MHz	15 MHz or 30 MHz	15 MHz
Supported Operating Points[2]	624/416/208/104 MHz	624/416/208/104 MHz	806/624/416/208/104 MHz
Internal SRAM banks	2	2	6

1. The SDRAM clock frequency is selected using the DDR_D0CS bit in the Application Subsystem Clock Configuration Register (ACCR).
2. Refer to Table 53, Table 55 and Table 56 for detalis on operating points.

6.2 Features

Features of the services unit are as follows:

- External clock source control to provide all clock sources for the processor
- Voltage control for the processor
- Real-time clock
- Reset and wakeup management
- Temperature sensor to determine the die temperature
- Power I^2C controller to communicate with an external power management IC (PMIC) to change voltages dynamically depending on frequencies and power modes

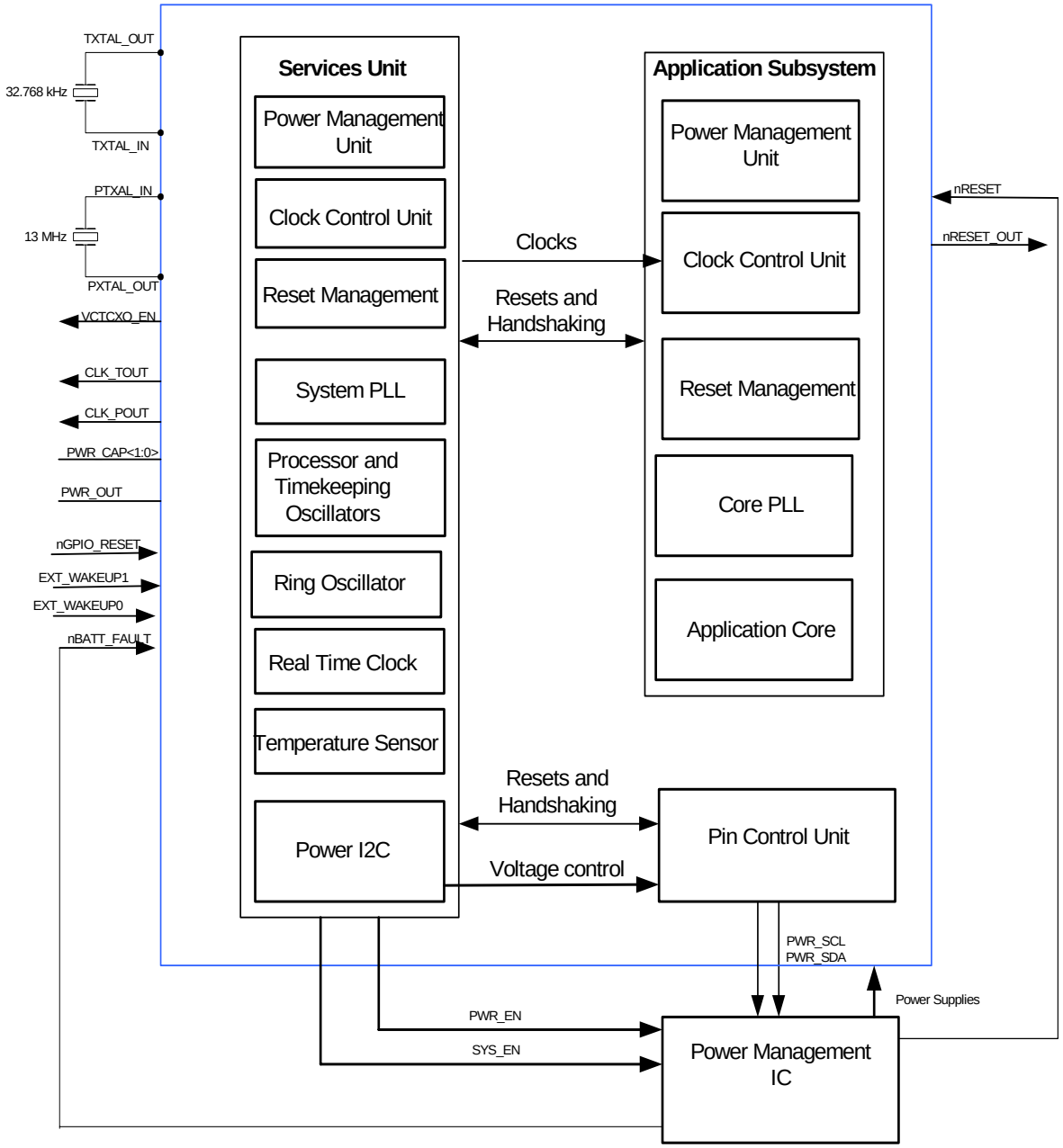
Features of the application subsystem unit are as follows:

- Power management, wakeup, and reset functions for the application subsystem
- Clocks needed by the application subsystem by using clock sources from the services unit

Figure 7 shows the services and application subsystem units and the external signals for power management. The following sections describe the power management architecture that is visible to software and how software can use it to manage overall processor power consumption.

Copyright © 2009 Marvell

Figure 7: Clocks and Power Management Units



6.3 External Power Management and Clock Signals

Table 40 lists the power management and clock signals.

Table 40: PXA3xx Processor Family Signals

Name	Type	Description	Notes
Crystal and Clock Signals			
PXTAL_IN	Analog	**Processor Crystal Input**	
PXTAL_OUT	Analog	**Processor Crystal Output**	
TXTAL_IN	Analog	**Timekeeping Crystal Input**	
TXTAL_OUT	Analog	**Timekeeping Crystal Output**	
CLK_TOUT	Output	**Timekeeping Clock Output**	
CLK_POUT	Output	**13 MHz Processor Clock Output**	
VCTCXO_EN	Output	**13 MHz Processor Clock Request (If external clock source is used)**	
Power and Voltage Control Signals			
PWR_EN	Output	**Power Enable for Core Power Supplies** Enables the external core (low-voltage power supplies (VCC_APPS and VCC_SRAM). De-asserting PWR_EN informs the external PMIC that the processor is entering S2/D3/C4 (Sleep) or S3/D4/C4 (Deep Sleep), and the low-voltage core power supplies (VCC_APPS and VCC_SRAM) are to be shut down. Asserting PWR_EN causes the PMIC to turn on the low-voltage supplies to resume normal operation. The PMIC must preserve the previous state of its regulators so that after re-enabling the low-power supplies, the regulators return to their last known voltage levels.	2
SYS_EN	Output	**System Enable for System Peripheral Power Supplies** Enables the external system I/O (high-voltage) power supplies. De-asserting SYS_EN informs the PMIC that the processor is entering S3/D4/C4 and that the high-voltage system power supplies are to be shut down. Assertion and de-assertion of SYS_EN occurs in the correct sequence with PWR_EN or I^2C commands to ensure the correct sequencing of power supplies when the various voltage domains are powered on or off. To resume normal operation, the PMIC must first turn on the system I/O (high-voltage) supplies in response to SYS_EN assertion and then turn on the core (low-voltage) supplies in response to PWR_EN assertion or I^2C commands. The PMIC must preserve the previous state of its regulators so that after re-enabling the low-power supplies, the regulators return to their last known voltage levels.	

Copyright © 2009 Marvell

Table 40: PXA3xx Processor Family Signals (Continued)

Name	Type	Description	Notes
nBATT_FAULT	Input	**Main Battery Fault** The processor monitors this digital input driven by the external PMIC that indicates power-supply regulator status and primary battery condition. nBATT_FAULT permits a combination of hardware and software management for power-fault conditions. nBATT_FAULT is asserted when the main battery is low or is removed from the system, indicating to the processor that power is going to shut down very soon. The processor can operate for a limited time from a coin-cell backup battery or from a super cap, which can supply a few cycles of full-run power. If the system is powered from an AC main source (90 VAC to 240 VAC or equivalent) with no battery, nBATT_FAULT can be pulled high. The processor recognizes EXT_WAKEUP<1:0> as a wakeup event when the nBATT_FAULT signal is asserted. If the processor enters S3/D4/C4 state due to the assertion of nBATT_FAULT, EXT_WAKEUP<1:0> wakeup events are detected while nBATT_FAULT is asserted, but the processor waits for nBATT_FAULT to be de-asserted before it exits S3/D4/C4 state.	
nRESET	Input	**Primary Hardware Reset** Notifies the processor to enter the hardware reset state. The assertion of nRESETcauses the processor to enter a complete and unconditional reset state. The processor does not recognize external events while nRESET is asserted. nRESET typically is driven directly by an external power-management integrated circuit (PMIC). Refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)* for timing information.	
nGPIO_RESET	Input	**GPIO Reset (Soft Reset)** Generates a reset in all power modes except S3/D4/C4. The pad for nGPIO_RESET contains an internal resistive pullup that is enabled during power-on, hardware, watchdog, and GPIO resets by programming the PUDH bit in the Power Management Unit General Configuration Register (PCFR).	3, 4
nRESET_OUT	Output	**Reset Out** Notifies the system that the processor is in the reset state. The processor asserts nRESET_OUT during the Start Of Day (SOD) and reset sequences. If the SL_ROD bit in the Power Management Unit General Configuration Register (PCFR) is cleared, the nRESET_OUT signal is also asserted during S2/D3/C4 and S3/D4/C4 states. If the PCFR[GP_ROD] bit is cleared, the nRESET_OUT pin is also asserted during GPIO reset. nRESET_OUT is de-asserted near the end of the reset sequence to signal other system components that the processor is coming out of reset. The de-assertion does not mean that the processor is out of the reset. The first boot ROM access to memory indicates that the processor is out of reset.	

Table 40: PXA3xx Processor Family Signals (Continued)

Name	Type	Description		Notes
PWR_SCL	Bidirectional	**Power Manager I^{2}C Clock** Contains an internal resistive pullup that is enabled during power-on, hardware, watchdog, and GPIO resets and is disabled when the PCFR[PUDH] bit is set.		
PWR_SDA	Bidirectional	**Power Manager I^{2}C Data** Contains an internal resistive pullup that is enabled during power-on, hardware, watchdog, and GPIO resets and is disabled when the PCFR[PUDH] bit is set.		
PWR_CAP<1:0>	Analog	**Power Capacitor<1:0>** Connects to an external capacitor used with internal DC-DC converter circuitry. The PWR_CAP signals must be connected because the DC-DC converter is used at all times. The PWR_CAP<1:0> signal balls must have an external 0.1-µF capacitor connected between them. If a polarized capacitor is used, the + plate must be connected to PWR_CAP1.		
PWR_OUT	Analog	**Internal Regulator Out** The DC-DC converter requires an external 0.1-µF capacitor to be connected between the PWR_OUT signal and ground. This connection is the only allowed connection or load on the PWR_OUT signal ball. This function is not optional and must be connected through a 0.1-µF capacitor to ground for guaranteed operation.		
EXT_WAKEUP<1:0>	Input	**External Wakeup<1:0>** A processor S2/D3/C4 and S3/D4/C4 wakeup source that functions as S3/D4/C4 wakeup even if S3/D4/C4 is entered because nBATT_FAULT is asserted. If nBATT_FAULT caused entry into this mode, EXT_WAKEUP<1:0> is the only wakeup source. Therefore, if nBATT_FAULT is used to indicate power supply health, this pin must be provided as a dedicated input. The pad for EXT_WAKEUP<1:0> contains an internal resistive pulldown that is enabled during power-on, hardware, global watchdog. GPIO resets are disabled when the PUDH bit in the Power Management Unit General Configuration Register (PCFR) is set. The EXT_WAKEUP<1:0> is not detected as a wakeup from S2/D3/C4 mode until after the processor enters this mode. S2/D3/C4 mode entry is complete when the PWR_EN signal is de-asserted. Similarly, EXT_WAKEUP<1:0> is not detected as a wakeup from S3/D4/C4 mode until after the processor enters this mode. S3/D4/C4 state entry is complete when the SYS_EN signal is de-asserted.		5
Power Supplies		Enable		
VCC_BBATT	High Voltage Power Supply		Battery or Backup Battery Supply, supplies the internal always-on domain (RTC, Services Unit, internal DC/DC converter)	
VCC_MVT	High Voltage Power Supply	SYS_EN	Ring Oscillator Voltage	
VCC_BG	High Voltage Power Supply	SYS_EN	Bandgap Voltage	

Copyright © 2009 Marvell

Table 40: PXA3xx Processor Family Signals (Continued)

Name	Type	Description		Notes
VCC_PLL	High Voltage Power Supply	SYS_EN	PLL Voltage	
VCC_OSC13M	High Voltage Power Supply	SYS_EN	13 MHz Oscillator Voltage	
VCC_APPS	Low Voltage Power Supply	PWR_EN	Applications Processor Core Voltage	
VCC_SRAM	Low Voltage Power Supply	PWR_EN	Internal SRAM Voltage	
VCC_IO1 VCC_IO3 VCC_MSL	High Voltage Power Supply	SYS_EN	Fixed supplies for standard CMOS I/Os interfacing to external component I/Os, which are also powered from this same supply. The I/Os for external components connected to the corresponding signals on the processor must be supplied from the same regulator.	
VCC_IO4 VCC_IO6	High Voltage Power Supply	SYS_EN	Fixed supplies for standard CMOS I/Os interfacing to external component I/Os, which are also powered from this same supply. The I/Os for external components connected to the corresponding signals on the processor must be supplied from the same regulator.	(PXA32x Only)
VCC_LCD	High Voltage Power Supply	SYS_EN	Power for output drivers to LCD panel. Optionally, this domain can be strapped to another I/O supply at the required voltage, if preferred.	
VCC_MEM	High Voltage Power Supply	SYS_EN	DDR SDRAM Controller Voltage	

Copyright © 2009 Marvell

Table 40: PXA3xx Processor Family Signals (Continued)

Name	Type	Description		Notes
VCC_CARD1 VCC_CARD2	High Voltage Power Supply	SYS_EN	Power for I/Os to an external removable storage or I/O card, including Universal Subscriber Identity Module (USIM) cards. These can have independent power supplies operating at different voltages. The VCC_CARDx voltage generated by the PMIC is software configurable at settings of 1.8 V, 3.0 V, or 3.3 V or disabled. The software voltage control uses I^2C commands or the PMIC decodes the processor UVS0, nUVS1, and nUVS2 outputs. USIM cards do not support 3.3 V. However, some alternate functions can be configured using the pins powered by VCC_CARD1 and VCC_CARD2. Therefore, these domains can be required to operate at 3.3 V.	
VCC_USB VCC_BIAS	High Voltage Power Supply	SYS_EN	Power for USB for standard differential USB I/Os interfacing to external components, which are also supplied from a fixed 3.3 V supply. VCC_USB is used for USB and iSRAM back bias on PXA32x and PXA30x. VCC_BIAS is used only for iSRAM back bias on PXA31x. VCC_USB powers the I/O for the USB interfaces. The USB differential signals D+ and D- is out of compliance with the USB specification if VCC_USB is below 2.8 V. The +5 V VBUS source from the USB host controller, which must be available for bus-powered peripherals, is supplied from an external source such as the PMIC. It is not part of the processor silicon.	
VCC_ULPI	High Voltage Power Supply	SYS_EN	Power for USB for standard differential USB digital I/Os interfacing to external components.	(PXA31x Only)
VCC_DF	High Voltage Power Supply	SYS_EN	Data Flash Interface Voltage	

Copyright © 2009 Marvell

Table 40: PXA3xx Processor Family Signals (Continued)

Name	Type	Description		Notes
VCC_CI	High Voltage Power Supply	SYS_EN	Camera Interface Voltage	
NOTE: 1. For detailed descriptions of each input signal, refer to the Pins and Control Chapter. 2. VCC_APPS and VCC_SRAM can also be controlled with two-wire serial interface (TWSI) commands. All low-voltage power supplies must be removed before any high-voltage power supply is removed. 3. If nBATT_FAULT is de-asserted and nGPIO_RESET is asserted after S3/D4/C4 entry starts but before SYS_EN is de-asserted, nGPIO_RESET generates a S3/D4/C4 wakeup and the processor returns to S0/D0/C0 state. 4. If nBATT_FAULT is de-asserted and nGPIO_RESET is asserted after SYS_EN is de-asserted, nGPIO_RESET does not generate a S3/D4/C4 wakeup and the processor remains in S3/D4/C4 until an enabled wakeup is detected. 5. EXT_WAKEUP<1> is only valid for the PXA32x processor.				

6.4 Power States

Power states are defined as configurations of voltage supplies and clocks for the services (S-states), the application subsystem (D-states), and the application subsystem core (C-states). A power mode is defined as a processor configuration of voltage domains and clocks composed of S-states, D-states, and C-states. Each valid combination of S/D/C states is given a power-mode mnemonic for convenience. Figure 8 shows the power states and the transitions between each state. Table 41 through Table 43 describe each power state.

Figure 8: Services and Application Subsystem Power States

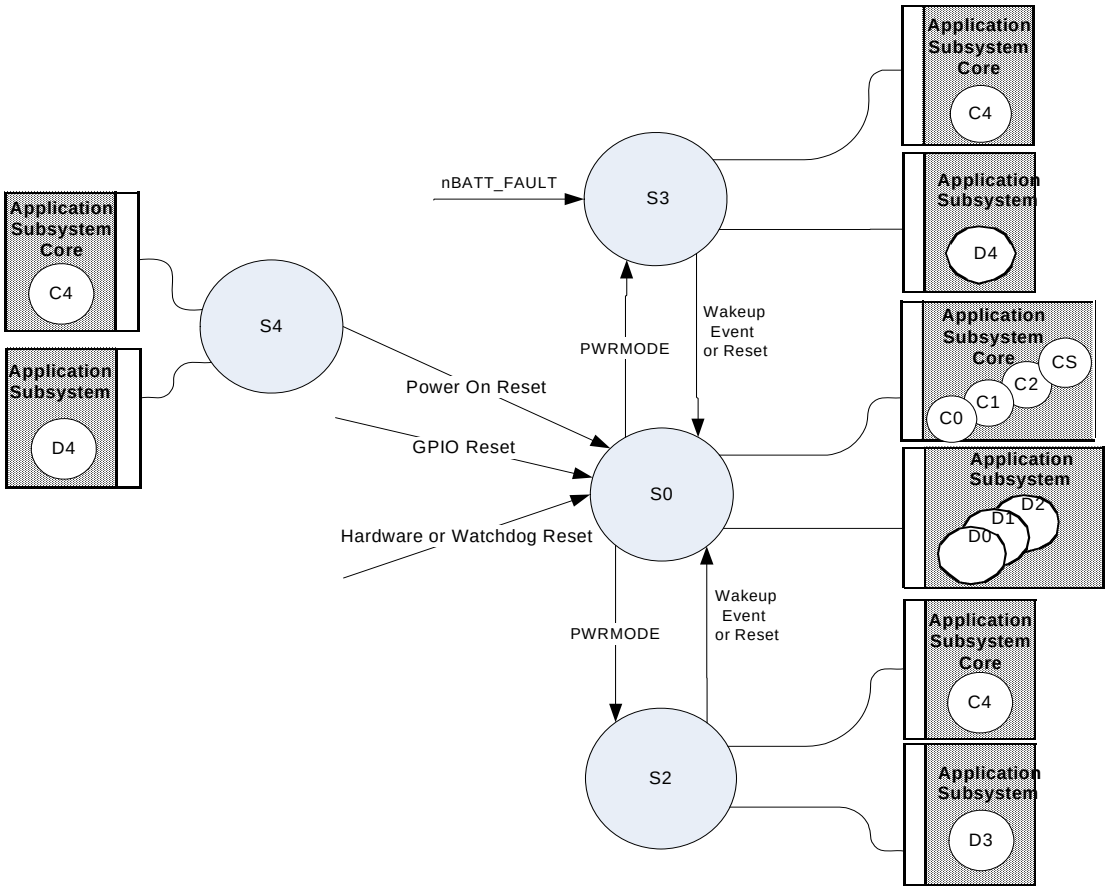


Table 41: Services Power States (S-States)

System State	Description
S0	• All internal power domains and external power supplies can be fully powered and functional. • All internal clocks can be running. • The application subsystem can be placed into D0, D1, or D2 power states and is controlled by the APMU.
S2	• PWR_EN is de-asserted. • Most internal power domains in the application subsystem can be powered down, including the core. • All clock sources except to the SPMU, APMU, and RTC are disabled. • External low-voltage power supplies (VCC_APPS and VCC_SRAM) can be disabled; remaining internal power domains are either powered down or placed into low-power state where state is retained but no activity is allowed. • Entry event: Core PWRMODE Register (CP14 Register 7) • Exit event: external and selected internal wakeup events; recovery requires system reboot because the program counter is not valid

Copyright © 2009 Marvell

Table 41: Services Power States (S-States) (Continued)

System State	Description
S3	• Both PWR_EN and SYS_EN are de-asserted. • All internal power domains except those for RTC, SPMU, 32 kHz oscillator, DC-DC converter, and Multi-Function Pins (MFP) can be powered down. • All power domains in the application subsystem are powered down. • All clock sources except to the SPMU and RTC are disabled. • Entry event: Core PWRMODE Register (CP14 Register 7) or nBATT_FAULT assertion. • Exit event: external and selected internal wakeup events. • EXT_WAKE<1:0> is the only wakeup event if entered because of nBATT_FAULT assertion. • Recovery requires a system reboot because the program counter is not valid.
S4	• Processor is powered off. • VCC_BBATT is not applied. • All power supplies are powered down. • All clocks are not functional. • Entry event: Removal of all power supplies. • Exit event: start-of-day operation occurs immediately after VCC_BBATT is applied.

Table 42: Application Subsystem Power States (D-States)

Sleeping State	Description
D0	• The application subsystem core can be in either run/turbo state or idle state. • All internal power domains and external power supplies are fully powered and functional. • SRAM arrays are powered on. • Application Subsystem Core is powered on. • All internal clocks can be running.
D1	• Some SRAM arrays and the mini-LCD are powered on. • The internal peripheral unit power domain is placed into low-power state where state is retained but no activity is allowed. • All clock sources to the peripheral units are disabled. • All power domains remain powered. • Mini-LCD operation is used to refresh the LCD screen; exit to either D0 or D2 state. • Entry event: wakeup event from D2 state or a software-issued D1 to the PWRMODE Register (CP14 Register 7). • Exit event: selected application subsystem wakeup event.
D2	• Core, peripheral units, and mini-LCD are in state-retaining mode. • SRAM arrays can be placed state-retaining mode or powered off. • All clock sources except to the APMU are disabled. • All power domains remain powered. • Exit to either D0 or D1 state. • Entry event: mini-LCD wakeup event from D1 state or a software issued D2 to the PWRMODE Register (CP14 Register 7). • Exit event: selected application subsystem wakeup event.

Copyright © 2009 Marvell

Table 42: Application Subsystem Power States (D-States) (Continued)

Sleeping State	Description
D3	• SRAM arrays are either in state-retaining mode or powered off. • Core and all peripherals are powered off. • All internal power domains but BPMU and selected SRAM arrays can be powered down. • All clock sources but the 32 kHz clock and the ring oscillator to the APMU are disabled. • Entry event: writing to the PWRMODE Register (CP14 Register 7). • Exit event: selected application subsystem wakeup (AD3ER) or services-unit wakeup event (PWER). • Recovery requires a system reboot because the program counter is not valid.
D4	• All internal power domains are powered down, all clocks are not functional, and all resets are asserted. • After exit from D4, a low-power mode-exit reset is asserted to the APMU. • Entry event: writing S3 to the PWRMODE register (CP14 Register 7). • Exit event: services unit wakeup event (PWER). • Recovery requires an entire system reboot.

Table 43: Application Subsystem Core Power States (C-States)

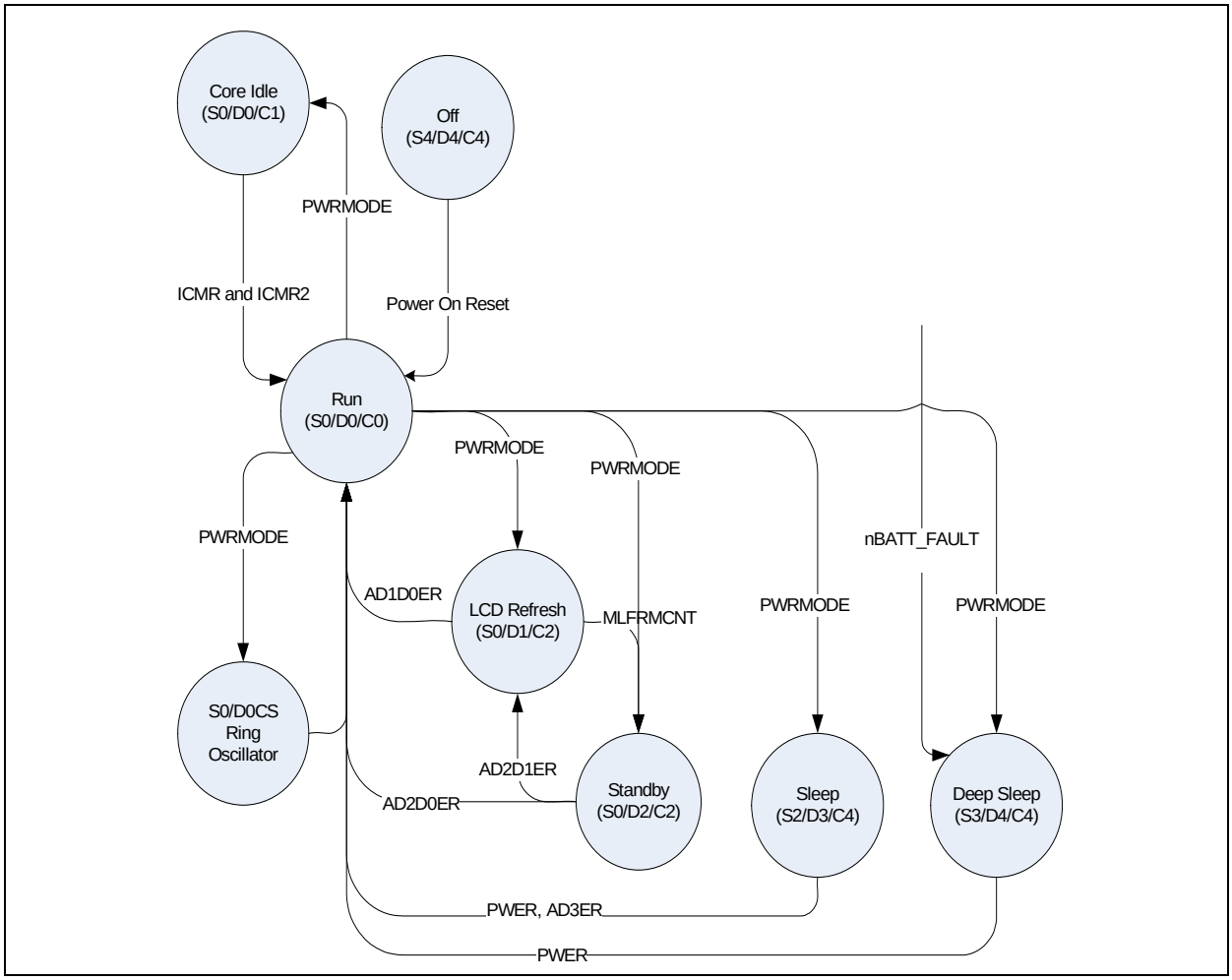
Sleeping State	Description
C0	• Core is powered on.
C1	• Core clock is off.
C2	• Core is powered off but is in state-retaining mode.
C4	• Power to the core is off and all state is lost.

6.5 Power Modes

The processor power modes are composed of the valid combinations of services and application subsystem power states. Figure 9 shows these modes and the transitions to each mode. Power management software enters each of these modes by writing to the Core PWRMODE Register (CP14 Register 7) and exits these modes through a wakeup event as described in Section 6.11, Wakeup Events and Sources. The following sections describe each processor power mode and list the procedures and requirements for entering and exiting each mode.

Copyright © 2009 Marvell

Figure 9: Processor Power Modes



6.5.1 S0/D0/C0 Run Mode

S0/D0/C0 run mode is the highest performance power mode. The processor core can run at clock speeds of 104–624 MHz depending on application requirements. Dynamic frequency control for the core enables software to change the core frequency without affecting the peripherals. The processor enters S0/D0/C0 because of an interrupt from one of the following sources:

- S0/D0/C1 or S0/D0CS/C1
- Wakeup event from S0/D1/C2, S0/D2/C2, S2/D3/C4, S3/D4/C4
- Reset from S0/D0/C1, S0/D0CS/C0, S0/D0CS/C1, S0/D1/C2, S0/D2/C2, S2/D3/C4, S3/D4/C4

Power management software must determine the interrupt source for S0/D0/C0 by examining and clearing status registers as shown in Table 44.

Table 44: S0/D0/C0 Entry Registers

Previous Power Mode	Status Register	Enable Registers	Description
S4/D4/C4 (Off)	ARSR[HWR][2]		Hardware or power-on reset since the last time software cleared this bit.
S3/D4/C4 (Deep Sleep)	PSR[SS3S][2], PSR[BFS][2,3], ARSR[GPR][1,5], ARSR[WDT][1,6], ARSR[LPMR][1], PSR[BFS][3], PSR[SS3S][4], PWSR[1], PECR[1]	PWER, PECR[1], PMER[7]	Software must enable the wakeup sources after a reset.
S2/D3/C4 (Sleep)	ASCR[D3S][1], PSR[SS2S][2], ARSR[GPR][1,5], RSR[WDT][1,6], AD3SR, PWSR[1], PECR[1]	AD3ER, PWER, PECR[1], PMER[7]	Software must enable the wakeup sources after a reset.
S0/D2/C2 (Standby)	ASCR[D2S][1], ARSR[GPR][1,5], ARSR[WDT][1,6], AD2D0SR	AD2D0ER, PMER[7]	
S0/D1/C2 (Standby with Mini-LCD)	ASCR[D1S][1],ARSR[GPR][1,5],ARSR[WDT][1,6], AD1D0SR	AD1D0ER, PMER[7]	
S0/D0CS/C0 (Ring Oscillator mode)	ARSR[GPR][1,5],ARSR[WDT][1,6]		
S0/D0CS/C1 (Ring Oscillator with core idle)	ARSR[GPR][1,5], ARSR[WDT][1,6]	ICMR[8], ICMR2[8]	
S0/D0/C1 (Run with core idle)	ARSR[GPR][1,5], ARSR[WDT][1,6]	ICMR[8], ICMR2[8]	

NOTE:
1. Write 0b1 to clear. Return to reset states with hardware or power-on resets.
2. Write 0b1 to clear. Not cleared by hardware or power-on resets.
3. If the processor enters S3/D4/C4 because of nBATT_FAULT, the Power Management Unit Status Register (PSR) BFS bit and Application Subsystem Reset Status Register (ARSR) LPMR bit are both set.
4. If the processor enters S3/D4/C4 because of a software write to PWRMODE[M, the Power Management Unit Status Register (PSR) SS3S bit and Application Subsystem Reset Status Register (ARSR) LPMR bit are both set.
5. A GPIO reset occurred since the last time software or a hardware or power-on reset cleared this bit.
6. A watchdog reset occurred since the last time software or a hardware or power-on reset cleared this bit.
7. PMER is a reserved system register and must not be written or read from outside the scope of the defined PWR_I2C software accesses.
8. Refer to the interrupt controller chapter for details on the Interrupt Controller Mask Registers (ICMR and ICMR2)

6.5.2 S0/D0/C1, Core Idle Mode

Core Idle mode saves power by stopping the core clock and retaining core state while allowing all other units to operate normally. To enter this mode, software must write a 0x1 to the M field in the

Copyright © 2009 Marvell

Core PWRMODE Register (CP14 Register 7). Any enabled interrupt can wake the core from Idle mode, regardless of the state of the Interrupt Controller Mask Register (ICMR). If normal interrupt-masking behavior is required, this feature can be disabled by setting the Disable Idle Mask (DIM) bit in the Interrupt Controller Control Register (ICCR). If the DIM bit is cleared and an Idle mode wakeup from a given unit is not wanted, the unit interrupt must be disabled at the unit level.

6.5.3 S0/D0CS/C0 Ring Oscillator Mode

An internal ring oscillator is used to clock the processor and system for lower-power operation. The ring oscillator can be used for MP3 Playback mode. In this mode, the core PLL and system PLL can be turned off to save power. Because of frequency limitations, not all peripherals can run. The core runs at 60 MHz from the ring oscillator, the DDR runs at up to 30 MHz, and internal SRAM runs at 60 MHz.

6.5.3.1 Entering S0/D0CS/C0

Software must perform the following steps to enter S0/D0CS/C0 state:

1. Software must disable interrupts so that the following sequence completes without interruption.
2. Ensure that System Bus 2 is enabled (D0CKEN_B[CKEN9] = 1).
3. Set the D0 Mode Clock Select (S0/D0CS/C0) bit in the Application Subsystem Clock Configuration Register (ACCR).
4. Write 0x5 to the M field in the Core PWRMODE Register (CP14 Register 7).

   All processor activity stops and all interrupt requests to the processor are held while the following sequence completes.

   a) All loads are completed and stores are sent to the system bus.

   b) The APMU drains the system by waiting until the following peripherals complete their outstanding transactions:

   - USB host
   - USB 2.0 high-speed client controller
   - 2D graphics controller
   - Quick-Capture Interface
   - NAND Flash controller
   - Dynamic memory controller
   - DMA controller
5. Wait for the RO_S status bit in the Application Subsystem Clock Status Register (ACSR) to set.
6. Disable the core PLL by setting the XPDIS in the Application Subsystem Clock Configuration Register (ACCR).
7. Wait for the core PLL to be disabled by reading the XPDIS_S bit in the Application Subsystem Clock Status register (ACSR) until they are set.
8. Wait 100 µs for the clocks to switch to the ring oscillator before disabling the system PLL by setting the SPDIS bits in the Application Subsystem Clock Configuration Register (ACCR).
9. Wait for the system PLL to be disabled by reading the SPDIS_S bit in the Application Subsystem Clock Status Register (ACSR) until they are set.

Refer to Table 50 to learn the requirements of each peripheral unit for entering the S0/D0CS/C0 state.

Note

The hardware voltage sequencer does not send PWR_I2C commands when entering S0/D0CS/C0.

6.5.3.2 Exiting S0/D0CS/C0

Software must perform the following steps to exit the S0/D0CS/C0 state:

1. Software must disable interrupts so that the following sequence completes without interruption.
2. Ensure that System Bus 2 is enabled (D0CKEN_B[CKEN9] = 1).
3. The DDR SDRAM delay lines must be recalibrated when exiting S0/D0CS/C0. For more information on delay line calibrations refer to the Dynamic Memory Controller chapter in the *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.
4. Clear the Application Subsystem Clock Configuration Register (ACCR)[D0CS] bit.
5. Set the Application Subsystem Interrupt Control/Status Register (AICSR)[PCIE] bit.

   When the PCIE bit is set, an interrupt is generated to the core. If the PCIE bit is cleared, no interrupt is generated and software must poll the status of the PLLs (ACSR[XPDIS_S] = 0 and ACSR[SPDIS_S] = 0) and their lock bits (ACSR[XPLCK] = 1 and ACSR[SPLCK] = 1]) to determine when to switch to PLL-based operation.
6. Enable the core PLL and system PLL by clearing the XPDIS and SPDIS bits in the Application Subsystem Clock Configuration Register (ACCR).
7. Wait for the core and system PLL to be enabled by reading the XPDIS_S and SPDIS_S bits in the Application Subsystem Clock Status Register (ACSR) until they are cleared.
8. Write 0x5 to the M field in the Core PWRMODE Register (CP14 Register 7).

   All processor activity stops and all interrupt requests to the processor are held while the following sequence completes.

   a) All loads are completed and stores sent to the system bus.

   b) The APMU drains the system by waiting until the following peripheral units complete their outstanding transactions:

   - USB host
   - USB 2.0 high-speed client controller
   - 2D graphics controller
   - Quick-Capture Interface
   - NAND Flash controller
   - Dynamic memory controller
   - DMA controller

   A power mode change interrupt (AICSR[PCIS]) occurs when the core clock switches to the PLL clocks after the power mode change.
9. Wait 100 µs after the RO_S status bit in the Application Subsystem Clock Status Register (ACSR) is set.
10. Disable the ring oscillator by setting the RO_CTRL bit in the Application Subsystem General Purpose Register (AGENP).

Refer to Table 50 to learn the requirements of each peripheral unit for entering the S0/D0CS/C0 state.

Copyright © 2009 Marvell

Note

Hardware does not disable the ring oscillator when the processor exits S0/D0CS/C0. Software can disable the ring oscillator after the processor exits S0/D0CS/C0 to save power in other power modes that do not require the ring oscillator. Disabling the ring oscillator increases the time required to enter S0/D0CS/C0, but it saves power in modes that do not use it.

6.5.3.3 CP14 Commands in Ring Oscillator Mode

Software can use the Coprocessor 14 interface registers (Table 63) to issue commands to the ACCU while the processor operates in S0/D0CS/C0 mode, as follows:

- *Low-power mode (D1, D2, and so on) entry command*. The processor behaves as if the command were issued in PLL Run mode (D0). Upon exit from the low-power mode, the processor returns to S0/D0CS/C0 mode. The clocks are not switched to PLL source even if the PLLs are enabled. In Run mode (either S0/D0/C0 or S0/D0CS/C0), the state of the PLLs is determined exclusively by the state of the XPDIS and SPDIS bits in the Application Subsystem Clock Configuration Register (ACCR).
- *Core Idle mode operation.* The processor enters Idle mode just as it does in PLL Run mode (S0/D0/C0). The processor returns to S0/D0CS/C0 mode when an interrupt occurs.
- *Another* S0/D0CS/C0 *entry or exit sequence.* A second entry into S0/D0CS/C0 while the processor is already in that mode is meaningless and is ignored. Software is allowed to proceed but with no state change. An exit sequence takes the processor out of S0/D0CS/C0 and places it into regular PLL Run mode (S0/D0/C0).

Warning

A core clock frequency/turbo change operation is not allowed in S0/D0CS/C0.

6.5.3.4 Non-CP14 Commands in Ring Oscillator Mode

In addition to the CP14 commands, software can issue commands by writing to various control bits in the Application Subsystem Clock Configuration Register (ACCR). Writing to the XPDIS and SPDIS bits enables or disables the respective PLLs. For details, see Section 6.6.4, Functional Clock Gating. Updates to the various frequency-selection fields do not result in an immediate frequency change (see Section 6.7.1, Core Frequency Changes). Rather, these values are held until the processor exits S0/D0CS/C0 and returns to PLL run mode (S0/D0/C0). At this point, the peripheral frequency-change process is initiated, and the frequencies of these units are updated accordingly. Software must poll the corresponding status-update fields in Application Subsystem Clock Status Register (ACSR) to determine when the frequency-change process is complete.

6.5.4 S0/D1/C2, Standby with LCD Refresh Mode

In standby with LCD Refresh mode, the core and most of the peripheral modules retain state but no activity is allowed, as follows:

- Both the core and system PLLs are turned off and all the application subsystem internal clocks are derived from the ring oscillator and the 32 kHz oscillator. If any peripherals are receiving an external clock, it continues to be used.
- Core and all peripherals are stopped in low-leakage state with context retained (except the mini-LCD and SRAM, which continue to run).

- The APMU, OS timers, mini-internal memory controller, SRAM arrays, and mini-LCD remain operational, so an external LCD can continue to be refreshed.
- Mini-LCD controller, instead of the primary LCD controller, refreshes the LCD from internal SRAM.
- The Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER) is used to determine the wakeup event.

6.5.4.1 Entering S0/D1/C2 Using Software

Software must perform the following steps to enter S0/D1/C2 power mode:

1. Program the appropriate registers to enable wakeups from low-power modes as described in Section 6.10.2, Preparing for Power Mode Transitions.
2. Ensure that the LCD controller finishes displaying the current frame buffer.
3. To prevent unexpected USB wakeup events from S0/D1/C2, disable the USB host functional clock by clearing the CKEN2 bit in the D0 Mode Clock Enable Register A (D0CKEN_A).
4. Enable the mini-LCD clock by setting the CKEN17 bit in the D0 Mode Clock Enable Register B (D0CKEN_B).

   All peripheral units but the mini-LCD, mini-internal memory controller, and the OS timer must be idle. The clocks to these peripherals stop when the processor enters S0/D1/C2 mode.

   All new transactions from the USB host, USB 2.0 high-speed client controller, 2-D graphics controller, Quick-Capture Interface, or DMA controller are ignored.
5. Ensure that System Bus 2 is enabled (D0CKEN_B[CKEN9] = 1).
6. Write 0x2 to the M field in the Core PWRMODE Register (CP14 Register 7).

   The processor places the DDR SDRAM into auto-refresh mode. For details, refer to the dynamic memory controller chapter in *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.
7. All processor activity stops and all interrupt requests to the processor are held while the following sequence completes:

   a) The core is halted.

   b) All GPIO pins are configured to the appropriate alternate functions.

   c) The internal SRAM banks are configured to their defined states.

   d) The core and system PLLs are disabled.

If a voltage change sequence is required while PVCR[FVE] is set, the APMU automatically requests a voltage-change sequence for VCC_APPS and VCC_SRAM for S0/D1/C2 mode. If PVCR[FVE] is cleared, power management software must change VCC_APPS and VCC_SRAM voltage levels to the appropriate levels defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Refer to Section 6.8, Voltage Management for details on voltage changes.

6.5.4.2 Entering S0/D1/C2 from a Wakeup Event from S0/D2/C2

When WERTC is set in the Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER), the real-time clock (RTC) alarm is used to wake up from S0/D2/C2 to S0/D1/C2. The internal SRAM banks selected by the Application Subsystem D2 Configuration Register (AD2R) are placed into their operational state.

6.5.4.3 Exiting S0/D1/C2

A wakeup event is sent to the processor to exit from S0/D1/C2 power mode. The following events cause a wakeup from S0/D1/C2 to a different power mode:

- After assertion of any preprogrammed wakeup event enabled in the Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER), the processor exits to S0/D0/C0.

Copyright © 2009 Marvell

The exit occurs immediately if the mini-LCD is requesting the wakeup. Otherwise, the exit occurs after the mini-LCD displays the current frame.

- Exit to S0/D0/C0 when MLFRMCNT[WKUP_EN] is set.
- Exit to S0/D2/C2 when MLFRMCNT[FWKUP_EN] is set.

Note

For more information on the MLFMCNT[FWKUP_EN] and MLFRMCNT[WKUP_EN] wakeups, refer to the mini-LCD chapter in *PXA300 Processor and PXA310 Processor Vol. III: Graphics and Input Controller Configuration Developers Manual*.

Software must perform the following steps after a wakeup event:

1. Issue PWR_I^{2}C commands to change VCC_APPS and VCC_SRAM voltages to the correct levels for the operating point (PVCR[FVE] = 1).
2. Enable the system PLL.
3. Enable the core PLL and reprogram the corresponding values in the Application Subsystem Clock Configuration Register (ACCR) register.
4. Restart the clocks for the units selected by the CKEN bits in the D0 Mode Clock Enable Register A (D0CKEN_A) and D0 Mode Clock Enable Register B (D0CKEN_B).

The core interrupts are no longer held and execution continues at the next instruction after the Write to the Core PWRMODE Register (CP14 Register 7). The DDR SDRAM must be brought out of self-refresh mode and delay line calibrations must be performed before DDR SDRAM transactions can occur. For details, refer to the dynamic memory controller chapter in *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.

S0/D1/C2 mode exits to S0/D2/C2 state with the assertion of an interrupt from the mini-LCD signifying completion of the current frame buffer. The ring oscillator can be used as a temporary clock source for the core and application subsystem peripherals if the PCCE bit in the Application Subsystem Clock Configuration Register (ACCR) is set. Refer to Section 6.6.2, Power Mode Clocking for details.

6.5.5 S0/D2/C2, Standby Mode

In Standby mode, the core and most peripheral modules retain state but no activity is allowed, as follows:

- Core and all peripherals stop in a low leakage state with context retained.
- SRAM state retention is optional.
- Saves power when LCD operation is not needed.
- State retention allows quick wakeup to S0/D0/C0 or entry into S0/D1/C2.

6.5.5.1 Entering S0/D1/C2 Using Software

Software must perform the following steps to enter S0/D1/C2:

1. Program the appropriate registers to enable wakeups from low-power modes as described in Section 6.10.2, Preparing for Power Mode Transitions.
2. To prevent unexpected USB wakeup events from S0/D2/C2, disable the USB host functional clock by clearing the CKEN2 bit in the D0 Mode Clock Enable Register A (D0CKEN_A).
3. Ensure that the LCD controller finishes displaying the current frame buffer.
4. Verify that all peripheral units (except the OS timer) are idle.
   The clocks to these peripherals are stopped when the processor enters S0/D2/C2 mode.
5. Ensure that System Bus 2 is enabled (D0CKEN_B[CKEN9] = 1).

6. Write 0x3 to the M field in the Core PWRMODE Register (CP14 Register 7).

   The processor places the DDR SDRAM into auto-refresh mode. For details, refer to the dynamic memory controller chapter in *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.

7. All processor activity stops and all interrupt requests to the processor are held while the following sequence completes.

   a) The core is halted.

   b) All new transactions from the following peripheral units are ignored:

   - USB host
   - USB 2.0 high-speed client controller
   - 2-D graphics controller
   - Handheld multimedia port
   - Quick-Capture Interface
   - DMA controller

   c) All GPIO pins are configured to the appropriate alternate functions.

   d) The internal SRAM banks are configured to their defined states.

   e) The core and system PLLs are disabled.

   f) If a voltage change sequence is required while PVCR[FVE] is set, the APMU automatically requests a voltage-change sequence for VCC_APPS and VCC_SRAM for S0/D2/C2 mode. If PVCR[FVE] is cleared, software must change VCC_APPS and VCC_SRAM voltage levels to the appropriate levels defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Refer to Section 6.8, Voltage Management for details on voltage changes.

6.5.5.2 Entering S0/D2/C2 due to a Wakeup Event

When the Wakeup Frame Count (MLFRMCNT[FRCOUNT]) bit expires while Frame Wakeup Enable (MLFRMCNT[FWKUP_EN]) is set, the mini-LCD causes a wakeup event to S0/D2/C2.

6.5.5.3 Exiting S0/D2/C2

The processor exits S0/D2/C2 power mode when it receives a wakeup event. The following events cause a wakeup from S0/D2/C2 to a different power mode:

- After assertion of any preprogrammed S0/D2/C2 mode wakeup event enabled in the Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER), the processor exits to S0/D0/C0.
- After assertion of any preprogrammed S0/D2/C2 mode wakeup event enabled in the Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER), the processor exits to S0/D1/C2.

Exiting S0/D2/C2 to S0/D0/C0

Software must perform the following steps after a wakeup event occurs to exit to S0/D0/C0:

1. Enable the system PLL.
2. Enable the core PLL and reprogram it with the corresponding values in the Application Subsystem Clock Configuration Register (ACCR) register.
3. Restart the clocks for units selected by the CKEN bits in the D0 Mode Clock Enable Register A (D0CKEN_A) and D0 Mode Clock Enable Register B (D0CKEN_B).

   The core interrupts are no longer held and execution continues at the next instruction following the write to the Core PWRMODE Register (CP14 Register 7).

The DDR SDRAM must be brought out of self-refresh mode and delay line calibrations must be performed before DDR SDRAM transactions can occur. For details, refer to the dynamic memory

Copyright © 2009 Marvell

controller chapter in the *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.

Note

The ring oscillator can be a temporary clock source for the core and peripherals if the PCCE bit in the Application Subsystem Clock Configuration Register (ACCR) is set. Refer to Section 6.6.2, Power Mode Clocking for details.

Exiting S0/D2/C2 to S0/D1/C2

An exit from the S0/D2/C2 state to the S0/D1/C2 mode occurs with the assertion of any enabled S0/D2/C2 mode wakeup event selected in the Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER). The wakeup process proceeds as follows:

1. The processor initiates voltage change sequence for VCC_APPS and VCC_SRAM for S0/D1/C2 mode, if needed.
2. The internal SRAM banks selected by the Application Subsystem D1 Configuration Register (AD1R) are configured to their operational state.

6.5.6 S2/D3/C4, Sleep Mode

In Sleep mode, power to the core and processor peripheral modules is removed, and I/O remains powered. SRAM state retention is optional. All application subsystem clocks to the core and all peripherals except the APMU and OS timers are disabled. The processor exits Sleep mode through external and select internal wakeup events. Recovery requires a reboot because the program counter is not valid.

6.5.6.1 Entering S2/D3/C4

Software performs the following steps to enter S2/D3/C4 power mode:

1. Configure the appropriate registers to enable wakeups from low-power modes as described in Section 6.10.2, Preparing for Power Mode Transitions.
2. To prevent unexpected USB wakeup events from S2/D3/C4, disable the USB host functional clock by clearing the CKEN2 bit in the D0 Mode Clock Enable Register A (D0CKEN_A).
3. Ensure that the LCD controller finishes displaying the current frame buffer. The LCD can operate while the processor is in S2/D3/C4 mode using a smart panel with a built-in frame buffer.
4. Verify that all peripheral units (except the OS timer) are idle. The clocks to the peripherals stop when the processor enters S2/D3/C4 mode.
5. Ensure that System Bus 2 is enabled (D0CKEN_B[CKEN9] = 1).
6. Write 0x6 to the M-field in the Core PWRMODE Register (CP14 Register 7). The processor places the DDR SDRAM into auto-refresh mode. For details, refer to the dynamic memory controller chapter in *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.

   All processor activity stops and all interrupt requests to the processor are held while the following sequence completes.

   a) The core is halted.

   b) All new transactions from the following peripheral units are ignored:
   - USB host
   - USB 2.0 high-speed client controller
   - 2D graphics controller

- Handheld multimedia port, Quick-Capture Interface
- DMA controller

c) All GPIO pins are configured to the appropriate alternate functions.

d) The internal SRAM banks are configured to their defined states as defined in the AD3R register.

e) The core and system PLLs are disabled.

If a voltage change sequence is required while PVCR[PVE] and PVCR[FVE] are set, the APMU requests a voltage-change sequence for VCC_APPS and VCC_SRAM for S2/D3/C4 mode. If PVCR[FVE] is cleared, power management software must change VCC_APPS and VCC_SRAM voltage levels to the appropriate levels defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Refer to Section 6.8, Voltage Management for details on voltage changes.

Note

The VCC_SRAM supply can be disabled, lowered, or maintained at the current voltage depending on the SRAM banks selected by the Application Subsystem D3 Configuration Register (AD3R). The VCC_APPS supply is disabled in S2/D3/C4. When PVCR[PVE] is set, the voltage sequencer automatically sends the commands to the PMIC to disable the VCC_SRAM and VCC_APPS supplies.

6.5.6.2 Exiting S2/D3/C4

The processor exits S2/D3/C4 power mode when it receives a wakeup event. The following events cause a wakeup from S2/D3/C4 to a different power mode:

- Assertion of any enabled S2/D3/C4 mode wakeup event selected in the Application Subsystem Wakeup from D3 Enable Register (AD3ER) and Power Manager Wake-Up Enable Register (PWER).
- Assertion of nBATT_FAULT.

Normal boot-up sequencing begins after a wakeup. All units in the application subsystem start with their predefined reset conditions. Software must examine the Application Subsystem Reset Status Register (ARSR) to determine whether the reset source is an S3 low-power state exit reset. Software must examine the Application Subsystem Power Status/Configuration Register (ASCR) to verify that the application subsystem was in S2/D3/C4 mode. If the Power Management Unit Scratch-Pad register (PSPR) in the SPMU is used for saving any general processor state during S2/D3/C4 mode, the state can be recovered.

6.5.7 S3/D4/C4, Deep Sleep Mode

In Deep Sleep mode, all power but VCC_BBATT (backup battery pin) is removed. All external power supplies except VCC_BBATT are powered down. The backup battery replaces the main battery. This is the mode used when devices are shipped. All clock sources are disabled except to the 32 kHz oscillator. Recovery is through external and select internal wakeup events. Recovery requires a reboot because the program counter is not valid.

6.5.7.1 Entering S3/D4/C4

The processor enters S3/D4/C4 mode either through software control by setting the mode bits in the Core PWRMODE Register (CP14 Register 7), or through hardware control by asserting nBATT_FAULT. The following procedures detail how to put the processor into S3/D4/C4 power mode.

Copyright © 2009 Marvell

Entering S3/D4/C4 Using Software

Software puts the processor into S3/D4/C4 mode by performing the following steps:

1. Program the appropriate registers to enable wakeups from low-power modes as described in Section 6.10.2, Preparing for Power Mode Transitions.
2. Ensure that the LCD controller finishes displaying the current frame buffer.
3. Ensure that all peripheral units (except the OS timer) are idle.

   The clocks to these peripherals are stopped when S3/D4/C4 mode is entered.
4. Write 0x7 to the M-field in the Core PWRMODE Register (CP14 Register 7).
5. All processor activity stops and all interrupt requests to the processor are held while the following sequence completes.

   a) The core is halted.

   b) All new transactions from the following peripheral units are ignored:
   - USB host
   - USB 2.0 high-speed client controller
   - 2D graphics controller
   - Quick-Capture Interface
   - DMA controller

   c) The core and system PLLs are disabled.

If a voltage change sequence is required while PVCR[PVE] is set, the APMU requests a voltage-change sequence for VCC_APPS and VCC_SRAM for S3/D4/C4 mode. If PVCR[FVE] is cleared, power management software must change VCC_APPS and VCC_SRAM voltage levels to the appropriate levels defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Refer to Section 6.8, Voltage Management for more information on voltage changes.

Entering S3/D4/C4 from nBATT_FAULT Assertion

Hardware automatically puts the processor into S3/D4/C4 mode when nBATT_FAULT is asserted.

6.5.7.2 Exiting S3/D4/C4

S3/D4/C4 mode exits to S0/D0/C0 state with the assertion of any enabled S3 state wakeup event selected in the Power Manager Wake-Up Enable Register (PWER). Then, normal boot-up sequencing begins. All units in the application subsystem start with their predefined reset conditions. Software must examine the Application Subsystem Reset Status Register (ARSR) to determine whether the reset source is a low-power mode exit reset. If the Power Management Unit Scratch-Pad Register (PSPR) is used to save any general processor state during S3/D4/C4 mode, the state can be recovered.

6.5.8 S4/D4/C4, Powered Off Mode

In Powered Off mode (S4/D4/C4), VCC_BBATT is not applied (SPMU and RTC are disabled). The processor is off in S4/D4/C4 mode with no clocks or units operating The processor enters this mode when VCC_BBATT is de-asserted. Conversely, start-of-day operation occurs immediately after a positive power edge is applied to VCC_BBATT. A normal start-of-day sequence starts when VCC_BBATT reaches the voltage level defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Refer to Section 6.12.1, Power-on Reset for more information on the start-of-day reset sequence.

6.5.9 Coupling Voltage Change with Power Modes

S0/D1/C2, S0/D2/C2, S2/D3/C4, and S3/D4/C4 power modes can be used to reduce power consumption significantly by gating clocks, reducing leakage, or powering down large sections of a

subsystem when they are not needed. Power modes can be coupled more efficiently with voltage changes using the sequences described here.

The following is the procedure to change VCC_APPS and VCC_SRAM when the processor enters a low-power mode:

1. When PVCR[FVE] is set, the processor initiates a voltage-change sequence to change VCC_APPS and VCC_SRAM supplies to the voltage required by the new low-power mode. If FVE is cleared, software must initiate any voltage changes as defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.
2. When PVCR[PVE] is set and the processor must turn off one of the supplies, it initiates a hardware voltage-change sequence to turn off the corresponding supplies. For example, the processor turns off the VCC_APPS supply when it enters S2/D3/C4 mode.
3. The APMU and SPMU enter the low-power mode selected and remain in that state until an enabled wakeup is detected. Then they exit the low-power mode. The processor performs the following steps as part of the exit sequence for S0/D0/C0 operation:

   a) When PVE is set, the processor turns on the corresponding power supplies.

   b) When FVE is set, the processor initiates a voltage-change sequence to change VCC_APPS and VCC_SRAM supplies to the voltage required by the new operating point. If FVE is cleared, software must initiate any voltage changes as defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*

Note

Clear the PVCR PVE and FVE bits if a PMIC is not in use. Software must verify that there are no outstanding voltage changes before writing to the Core PWRMODE Register (CP14 Register 7) to initiate a power mode change. When the hardware voltage change sequencer is used for power mode changes (PVCR[FVE] = 1 and PVCR[PVE] = 1), software must read the PVCR[VCSA] bit to verify that it is cleared to ensure that the PWR_I^2C module is idle.

6.5.10 Power Mode Status

The Power Management Unit Status Register (PSR) and Application Subsystem Power Status/Configuration Register (ASCR) registers are written when the processor enters low-power modes (see Table 45). The core can read these registers to determine whether the processor was previously present in the S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4 modes. Software must clear these registers after reading them as the processor exits a low-power mode.

Table 45: Low Power Mode Status Registers

Power Mode	Status Register
S0/D1/C2	ASCR[D1S][1]
S0/D2/C2	ASCR[D2S][1]
S2/D3/C4	ASCR[D3S][1],PSR[SS2S][2]
S3/D4/C4	PSR[SS3S][2], PSR[BFS][2,3]

Copyright © 2009 Marvell

Table 45: Low Power Mode Status Registers

Power Mode	Status Register

NOTE:
1. Write 0b1 to clear. Return to reset states with hardware or power-on resets.
2. Write 0b1 to clear. Not cleared by hardware during power-on resets.
3. If the processor enters S3/D4/C4 because nBATT_FAULT is asserted, the Power Management Unit Status Register (PSR) BFS bit and PSR[SS3S] bit are both set.

6.5.11 Power Mode Latencies

Table 46 lists the low-power mode entry latencies for the application subsystem. All latencies are measured from the Write to the PWRMODE register. All latencies are based on the minimum time for system-level responses to power-mode-entry requests. For example, if the SPMU requests a power-mode entry from the DDR, the latencies are based on the assumption that the DDR responds in the minimum time possible. In addition, the latencies are based on no transfers pending in the core when the Core PWRMODE Register (CP14 Register 7) is written and no time allocated for power supply enable/disable or ramping.

Table 46: Low-Power Mode Entry Latencies

Transition	PVE	FVE	Power Reduction occurs[1] (µs)	Entry Done[2] (µs)
S0/D0/C0 -> S0/D0CS/C0	0	0	8.5[4]	10[3]
S0/D0/C0 -> S0/D1/C2	0	0	21	72
	-	1	370	430
S0/D0/C0 -> S0/D2/C2	0	0	21	1415
	1	1	310	1800
S0/D1/C2 -> S0/D2/C2	-	-	1.5[5]	1600
S0/D0/C0 -> S2/D3/C4[6]	0	0	10	1230
	1	1	275	1410
S0/D0/C0 -> S3/D4/C4	0	0	6	1600
	1	-	6	1800

Table 46: Low-Power Mode Entry Latencies (Continued)

Transition	PVE	FVE	Power Reduction occurs[1] (µs)	Entry Done[2] (µs)

1. Time where the power mode transition has occurred and the low-power mode reduced power consumption occurs.
2. Entry into the low-power mode has completed and transition to a different power mode can begin.
3. No PMIC commands are sent when entering S0/D0CS/C0. User must execute frequency change to lowest core frequency before entering the mode. This frequency change executes dynamic voltage management (DVM). (If the frequency change is removed, then DVM time must be added).
4. Time given is to application subsystem core clock re-started using ring oscillator
5. Time given is to Mini-LCD clock disabled.
6. PWR_EN is de-asserted 55 µs after Core PWRMODE Register (CP14 Register 7) write occurs.

NOTE: The following are the assumptions for entry times -

- There are no transfers pending within the system that cause entry sequence to stall.
- There are no external transfers pending that cause entry sequence to stall.
- All counters that are user programmable that can cause entry sequence to stall are set to minimum values.
- LPM_DEL = 0; PWR_DEL = 0; SYS_DEL = 0.
- Entry times provided are typical.
- Entry done times to S0/D2/C2, S2/D3/C4 and S3/D4/C4 do not include possible synchronization times with 32 kHz oscillator clock. Since each of these sequences require clocks to switch to 32 kHz and signals to be synchronized to 32 kHz clock, the maximum difference between fast clock and 32 kHz clock has not been included. Entry times may be longer if maximum difference between clocks are used.

Table 47 lists the low-power mode exit times for the application subsystem. For S0/D1/C2, S0/D2/C2, and S2/D3/C4 modes, the times are measured from the assertion of an application subsystem wakeup until the wakeup is asserted to the core (S0/D1/C2 and S0/D2/C2) or until the first bus request from the core (S2/D3/C4). For S3/D4/C4, the times are from the assertion of an external wakeup to the services unit until the first bus request from the core. All times assume a minimum response from any units requiring handshaking.

Table 47: Low Power Mode Exit Latencies

Transition	PVE	FVE	PCCE	Wakeup Asserted to Core[1] (µs)	Exit Done[2,3] (µs)
S0/D0CS/C0 -> S0/D0/C0	-	0	N/A	3	5
	-	1	N/A	3	5[6]
S0/D1/C2 -> S0/D0/C0	-	0	0	160	390
	-	0	1	140	190
	-	1	0	170	390
S0/D2/C2 -> S0/D0/C0	0	0	1	640	670
	1	1	0	650	865
S0/D2/C2 -> S0/D1/C2	-	-	-	202[5]	203
S2/D3/C4 -> S0/D0/C0	1	1	0	990[4]	1135

Copyright © 2009 Marvell

Table 47: Low Power Mode Exit Latencies

Transition	PVE	FVE	PCCE	Wakeup Asserted to Core[1] (µs)	Exit Done[2,3] (µs)
S3/D4/C4 -> S0/D0/C0	0	-	-	990[4]	1000
	1	-	-	1268[4]	1270

1. First instruction begins executing.
2. Exit to the next power mode is complete and all transitions are complete
3. Exit times do not include the 13 MHz processor clock stabilization time defined using the VXCOST bit in the Oscillator Configuration Register (OSCC). Clock stabilization times range between 1000 µs and 7000 µs.
4. Time is until first core bus access for S2/D3/C4 and S3/D4/C4 -> S0/D0/C0 transitions.
5. Time is to Mini-LCD clock started.
6. No PMIC commands are sent as a result of D0CS exit. User must exit to lowest frequency and then execute frequency change after exit. If requirement to exit to lowest frequency is removed, PMIC command and supply ramp time must be added.

NOTE: The following are the assumptions for exit times -

- There are no transfers pending within the system that cause exit sequence to stall.
- There are no external transfers pending that cause exit sequence to stall.
- All counters that are user programmable that can cause exit sequence to stall are set to minimum values: PCFR[SYS_DEL], PCFR[PWR_DEL], PCFR[LPM_DEL] and OSCC[VCXOST] = 0x0.
- Exit times provided are typical.
- Exit times from S0/D2/C2, S2/D3/C4and S3/D4/C4 do not include possible synchronization times with 32 kHz oscillator clock. Since each of these sequences require clocks to switch from 32 kHz and signals to be synchronized to 32 kHz clock, the maximum difference between fast clock and 32 kHz clock has not been included. Exit times may be longer if maximum difference between clocks are used.

Table 48: PXA31x Low Power Mode Exit Latencies

Transition	PVE	FVE	PCCE	PXA31x Wakeup Asserted to Core[1] (µs)	PXA31x Exit Done[2,3] (µs)
S0/D0CS/C0 -> S0/D0/C0	-	0	N/A	3	5
	-	1	N/A	3	5[6]
S0/D1/C2 -> S0/D0/C0	-	0	0	345	590
	-	0	1	340	390
	-	1	0	350	590
S0/D2/C2 -> S0/D0/C0	0	0	1	630	662
	1	1	0	690	905
S0/D2/C2 -> S0/D1/C2	-	-	-	202[5]	203

Table 48: PXA31x Low Power Mode Exit Latencies (Continued)

Transition	PVE	FVE	PCCE	PXA31x Wakeup Asserted to Core[1] (µs)	PXA31x Exit Done[2,3] (µs)
S2/D3/C4 -> S0/D0/C0	0	0	1	750[4]	850
	1	1	0	1000[4]	1145
S3/D4/C4 -> S0/D0/C0	0	-	-	990[4]	1000
	1	-	-	1268[4]	1270

1. First instruction begins executing.
2. Exit to the next power mode is complete and all transitions are complete.
3. Exit times do not include the 13 MHz processor clock stabilization time defined using the VXCOST bit in the Oscillator Configuration Register (OSCC). Clock stabilization times range between 1000 µs and 7000 µs.
4. Time is until first core bus access for S2/D3/C4 and S3/D4/C4 -> S0/D0/C0 transitions.
5. Time is to Mini-LCD clock started.
6. No PMIC commands are sent as a result of D0CS exit. User must exit to lowest frequency and then execute frequency change after exit. If requirement to exit to lowest frequency is removed, PMIC command and supply ramp time will need to be added.

NOTE: The following are the assumptions for exit times -

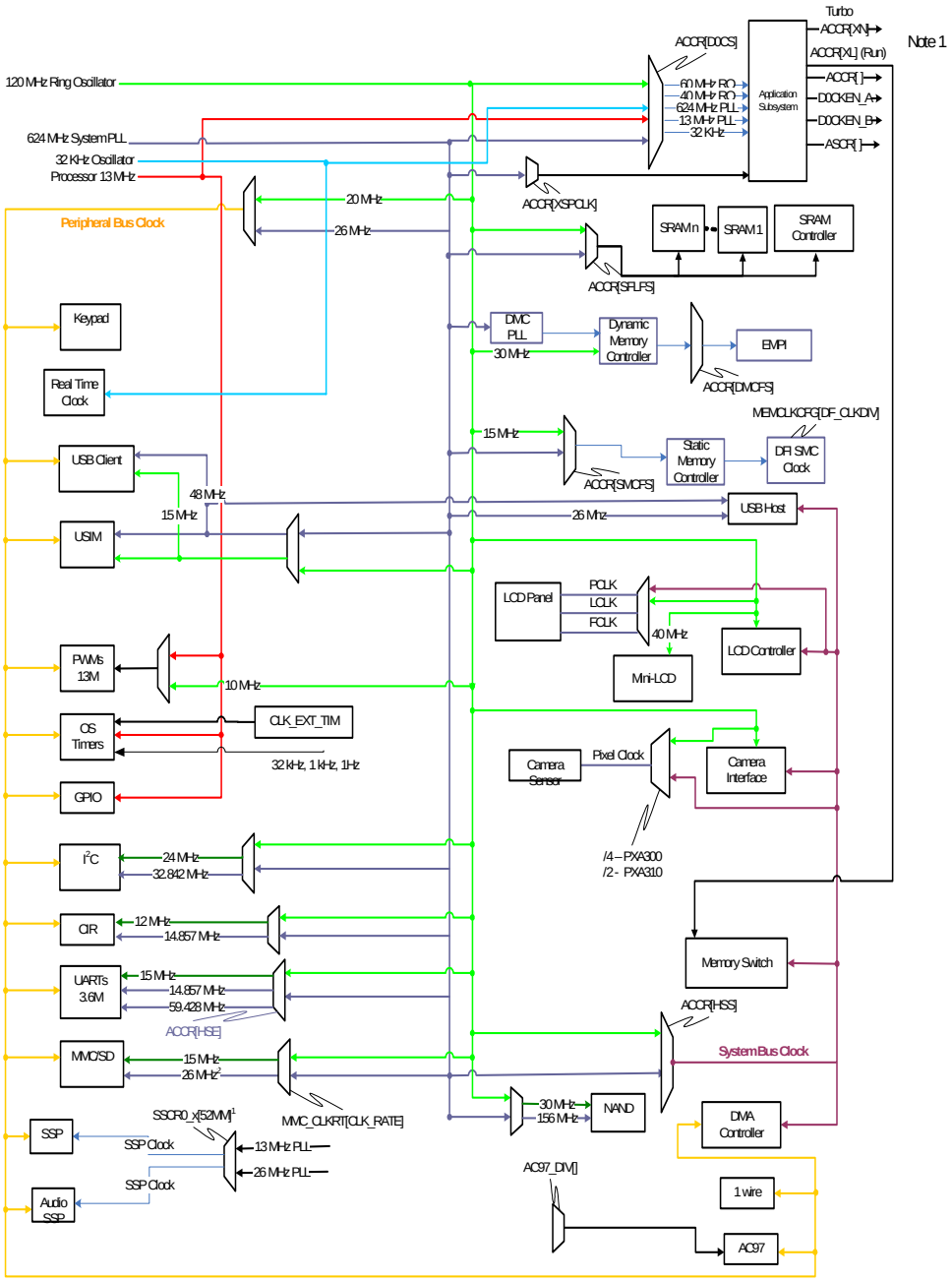
- There are no transfers pending within the system that cause exit sequence to stall.
- There are no external transfers pending that cause exit sequence to stall.
- All counters that are user programmable that can cause exit sequence to stall are set to minimum values:PCFR[SYS_DEL], PCFR[PWR_DEL], PCFR[LPM_DEL] and OSCC[VCXOST] = 0x0.
- Exit times provided are typical.
- Exit times from S0/D2/C2, S2/D3/C4and S3/D4/C4 do not include possible synchronization times with 32 kHz oscillator clock. Since each of these sequences require clocks to switch from 32 kHz and signals to be synchronized to 32 kHz clock, the maximum difference between fast clock and 32 kHz clock has not been included. Exit times may be longer if maximum difference between clocks are used.

6.6 Processor Clocks

Figure 10 shows the clock sources and distribution. Clock frequencies depend on the power mode and processor operating point. This section describes the operating points, behavior in low-power modes, and how to change frequencies in the system.

Copyright © 2009 Marvell

Figure 10: Clock Generation Diagram



6.6.1 Processor System Clocks

Table 49 describes the processor system clocks.

Table 49: System Clocks

Clock	Description
Processor oscillator (13 MHz)	The reference clock for the system and core PLLs as well as the functional clock for several peripheral units. The internal PLL frequency multipliers and several peripheral modules use the processor oscillator output as a reference. The CLK_POUT signal can be used to drive a buffered version of the PXTAL_IN signal for external system use. The PEN bit allows software to enable or disable the CLK_POUT signal for certain operating modes. The processor oscillator input can be generated in one of two ways: 1. An external crystal. For lowest power consumption, a 13 MHz crystal must be connected between the PXTAL_IN and PXTAL_OUT pins. 2. An external clock source on the PXTAL_IN pin. The external clock source can be 13 or 26 MHz and can be an external VCTCXO and clock limiter stabilizer or an external CMOS clock source that is enabled by VCTCXO_EN. After VCTCXO_EN is asserted, the Services Clock Control Unit waits the time programmed in VCXOST in the Oscillator Configuration Register (OSCC) before using the clock. Program the startup time to be from 1–230 32 kHz clock cycles (based on specifications for 13 MHz PXTAL_IN source used in the system). VCTCXO_EN is de-asserted in S2/D3/C4 and S3/D4/C4 and is asserted in S0/D0/C0 when the processor oscillator clock is required. The processor oscillator clock is required when the PEN bit in the Oscillator Configuration Register (OSCC) is set.
Core PLL (104–806 MHz)	Programmable frequency clocks for the application subsystem core. The core PLL is the clock source for the core in Run mode. The core PLL generates two output frequencies—one for the core clock (processor oscillator [13 MHz] * XL * XN) and one for the core system bus (processor oscillator [13 MHz] * XL). Core clock frequencies where XN is equal to 2 are called *turbo* frequencies. If XN = 1, it is referred to as *run* frequency.
System PLL (624 MHz)	Frequency clocks for the system buses, peripheral bus, and peripheral units in the processor. The system PLL creates a fixed-frequency clock used for generating many of the fixed-frequency clocks in the application subsystem.
DMC PLL (26–260 MHz)	Clock to the dynamic memory controller to generate the external DDR SDRAM clock frequency.
Ring oscillator (120 MHz)	Fixed-frequency clocks used during power-mode transitions to Run mode (S0/D0/C0), if enabled, and during Ring Oscillator mode (S0/D0CS/C0), if selected. The APMU and the application subsystem can use the ring-oscillator clocks when the APMU is in D0 or D1 power modes. When the APMU is in D0 mode, the 120 MHz ring oscillator is enabled to output the 120 MHz clock.
Ring oscillator (40 MHz)	Internal clock for the application subsystem power management unit in the D0 state and for the mini-LCD unit during Standby with LCD Refresh mode (S0/D1/C2). When the APMU is in D0 and D1 power mode, the 40 MHz ring oscillator is enabled to output the 40 MHz clock. In D1 power mode the 40 MHz clock is used for the mini-LCD panel controller and the APMU. In D0 mode, the APMU uses the 40 MHz ring oscillator.
Timekeeping oscillator (32.768 kHz)	Low-power, low-frequency oscillator that clocks the real-time clock, OS timer, and service power manager unit. The timekeeping oscillator is always enabled; the CLK_TOUT signal can be used to drive a buffered version of the TXTAL_IN pin and can be enabled using the TENS3, TENS2, and TENS0 bits in the Oscillator Configuration Register (OSCC). By default, CLK_TOUT is enabled in S0 mode, which includes power-on and hardware reset. For lowest power consumption, a 32.768 kHz crystal must be connected between the TXTAL_IN and TXTAL_OUT pins.

Copyright © 2009 Marvell

6.6.2 Power Mode Clocking

When the processor enters a low-power mode either because software writes to the Core PWRMODE Register (CP14 Register 7) or because the main battery is removed (nBATT_FAULT), bus and peripheral unit configurations are present for each mode. Table 50 shows each power mode and the operating frequency for each bus and peripheral unit in each power mode. Table 50 also defines requirements for peripheral units when they enter and exit different power modes.

Table 50: Power Mode Clock Frequencies

	S0/D0/C0 (Run) or S0/D0/C1 (Idle)	S0/D0CS/ C0 (Ring Oscillator Mode)	S0/D1/C2 (Standby with LCD Refresh)	S0/D2/C2 (Standby)	S2/D3/C4 (Sleep)	S3/D4/C4 (Deep Sleep)	S4/D4/C4 (Off)
Application Subsystem Core	ACCR[XL] ACCR[XN]	60	State Retentive	State Retentive	Off	Off	Off
System Bus	208, 156, 104[1]	60	State Retentive	State Retentive	Off	Off	Off
Peripheral Bus	26	20	State Retentive	State Retentive	Off	Off	Off
UART	14.857 or 59.428[2]	15[3,4]	State Retentive	State Retentive	Off	Off	Off
SSP	13 or 26[5]	13[6]	State Retentive	State Retentive	Off	Off	Off
MMC	19.5 or 26[7]	15	State Retentive	State Retentive	Off	Off	Off
LCD	208, 156, 104[1]	60[8]	State Retentive	State Retentive	Off	Off	Off
Mini-LCD	Off	Off	40 MHz	State Retentive	State Retentive	Off	Off
USB Host and Client	On	On[9]	State Retentive	State Retentive	Off	Off	Off
NAND Controller	156	30	State Retentive	State Retentive	Off	Off	Off
Static Memory Controller	208. 104, 78[10]	15	State Retentive	State Retentive	Off	Off	Off
Dynamic Memory Controller	130, 13[11]	15, 30[12,13]	State Retentive	State Retentive	Off	Off	Off
Dynamic Memory Controller Refresh Clock	13	13.33	State Retentive	State Retentive	State Retentive	Off	Off

Table 50: Power Mode Clock Frequencies (Continued)

	S0/D0/C0 (Run) or S0/D0/C1 (Idle)	S0/D0CS/ C0 (Ring Oscillator Mode)	S0/D1/C2 (Standby with LCD Refresh)	S0/D2/C2 (Standby)	S2/D3/C4 (Sleep)	S3/D4/C4 (Deep Sleep)	S4/D4/C4 (Off)
Camera Interface	104, 78, 52	30[14]	State Retentive	State Retentive	State Retentive	Off	Off
Internal SRAM	312, 208, 156, 104[15]	60	Powered on or State Retentive	Powered on or State Retentive	Powered on or State Retentive	Off	Off
Memory Switch	104, 208. 312[16]	60	On	On	State Retentive	Off	Off
AC97	312 / Divisor[17]	15	State Retentive	State Retentive	State Retentive	Off	Off
PWM	13	10	State Retentive	State Retentive	State Retentive	Off	Off
CIR	14.857	12	State Retentive	State Retentive	State Retentive	Off	Off
I^2C	32.842	24	State Retentive	State Retentive	State Retentive	Off	Off
Keypad	32 kHz	32 kHz	State Retentive	State Retentive	State Retentive	Off	Off
Real Time Clock	32 kHz	32 kHz	32 kHz	32 kHz	32 kHz	32 kHz	Off
OS Timer	13, 32 kHz	32 kHz	32 kHz	32 kHz	32 kHz	Off	Off

Copyright © 2009 Marvell

Table 50: Power Mode Clock Frequencies (Continued)

	S0/D0/C0 (Run) or S0/D0/C1 (Idle)	S0/D0CS/ C0 (Ring Oscillator Mode)	S0/D1/C2 (Standby with LCD Refresh)	S0/D2/C2 (Standby)	S2/D3/C4 (Sleep)	S3/D4/C4 (Deep Sleep)	S4/D4/C4 (Off)

NOTE:
1. ACCR[HSS]
2. IER[HSE] is used to select the high speed UART clock rate.
3. 921 Kbps maximum.
4. UART transactions must be idle before the processor enters or exits S0/D0CS/C0.
5. 26 MHz in master mode only (PXA31x only).
6. Normal operation except for audio mode, which requires an external clock source. SSP must be idle before the processor enters or exits S0/D0CS/C0.
7. MMC_CLKRT[CLKRATE].
8. Refresh rate is controlled through the LCCR3[PCD] register.
9. Can detect USB resumes, but controller cannot be used.
10. ACCR[SMCFS]
11. ACCR[DMCFS]. 13 MHz only supported on reset.
12. ACCR[DDR_D0CS] (PXA31x Only)
13. Recalibration of delay lines is required on exit from DOCS mode. The dynamic memory controller hardware automatically completes this recalibration process, but software must enable the recalibration feature. See the dynamic memory controller chapter in the *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual.*
14. Operates with limited functionality. C_PCLK pixel clock is limited to (18.75 MHz)/2.
15. ACCR[SFLFS]
16. ACCR[XL]
17. AC97_DIV[NUMERATOR; DENOMINATOR]

6.6.3 Clocks During Power Mode Changes

The PCCE bit in the Application Subsystem Clock Configuration Register (ACCR) enables the processor to use the ring oscillator output clock as a clock source when it transitions from any low-power mode to S0/D0/C0 mode. When PCCE is cleared, processor operation does not begin until the core and system PLLs are enabled and locked. Then, the processor clocks are started with the PLL outputs. No software intervention is possible.

When the processor enters a low-power mode while PCCE is set, the processor always wakes up in S0/D0CS/C0 mode regardless of the power mode it exits. Power management software must follow the procedures for exiting S0/D0CS/C0 to return to S0/D0/C0.

When the processor exits S0/D0CS/C0 with PCCE cleared, software must read the ROS bit in the Oscillator Configuration Register (OSCC) to check the status of the ring oscillator and wait until the ring oscillator is turned off before it sets PCCE for the next low-power-mode entry.

Note

PCCE affects the behavior of the clocks unit while the processor exits low-power mode. Therefore, software must never update the PCCE bit while the processor is exiting a low-power mode.

6.6.4 Functional Clock Gating

The ACCU contains two clock-enable registers:

- D0 Mode Clock Enable Register A (D0CKEN_A)
- D0 Mode Clock Enable Register B (D0CKEN_B).

When the processor is in S0/D0/C0 Run mode, the CKEN bits in these registers are used to enable or disable the clocks to individual peripheral units. The CKEN bits should be used only when a module is unused. After power-on or a hardware reset, disable the clock of any unused module to save power. Furthermore, if a module is temporarily idle, additional power savings are obtained using the CKEN bits to disable the unit clock. The CKEN bits enable/disable only the functional clock inputs to each module, so register Reads/Writes to that module still function properly. Consult the appropriate module chapter for more information.

6.7 Operating Points

The processor has a predefined set of operating points as listed in Table 53, Table 55 and Table 56. Each operating point consists of a set of valid register values that must be used to guarantee proper functionality. The Application Subsystem Clock Configuration Register (ACCR) is used to select the required frequency for each peripheral and bus to create the valid operating points. This section describes the operating points and the requirements for changing them.

Each operating point has predefined voltage levels for each voltage domain. For more information on the required voltage levels, refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. For more information on requirements for changing operating points and voltage levels, refer to Section 6.8, Voltage Management.

Table 53: PXA32x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 130 MHz)

XL	XN	Frequency (MHz)								
		Run (ACCR[XL])	Turbo (ACCR[XN])	Switch (ACCR[XL])	System Bus (ACCR[HSS])	LCD (ACCR[HSS])	Internal SRAM Controller (ACCR[SFLFS])	DDR Controller SDRAM Clock (ACCR[DMCFS])	Static Memory Controller (ACCR[SMCFS])	Data Flash Controller not programmable
Ring Oscillator[1]		60	—	60	60	60	60	30	15	30
8[2]	1	104 (0x01)	— (0x01)	104 (0x01)	104 (0x00)	104 (0x00)	104 (0x00)	26 (0x00)	78 (0x00)	19.5
8	1	104 (0x08)	— (0x01)	104 (0x08)	104 (0x00)	104 (0x00)	104 (0x00)	130 (0x03)	78 (0x00	156
16	1	208 (0x10)	— (0x01)	208 (0x10)	104 (0x00)	104 (0x00)	156 (0x01)	130 (0x03)	104 (0x02)	156
16	2	208 (0x10)	416 (0x02)	208 (0x10)	156 (0x01)	156 (0x01)	208 (0x02)	130 (0x03)	104 (0x02)	156

Copyright © 2009 Marvell

Table 53: PXA32x Core PLL, Turbo and Run Mode Output Frequencies (Continued)(ACCR[DMCFS] = 130 MHz)

XL	XN	Frequency (MHz)								
		Run (ACCR[XL])	Turbo (ACCR[XN])	Switch (ACCR[XL])	System Bus (ACCR[HSS])	LCD (ACCR[HSS])	Internal SRAM Controller (ACCR[SFLFS])	DDR Controller SDRAM Clock (ACCR[DMCFS])	Static Memory Controller (ACCR[SMCFS])	Data Flash Controller not programmable
24	2	312 (0x18)	624 (0x02)	312 (0x18)	208 (0x02)	208 (0x02)	312 (0x03)	130 (0x03)	208 (0x05)	156
31	2	403 (0x1F)	806 (0x02)	403 (0x1F)	208 (0x02)	208 (0x02)	312 (0x03)	130 (0x03)	208 (0x05)	156

NOTE:
1. Frequencies shown in this row are ideal-nominal and are based upon the ring oscillator, which has a ±2% tolerance; actual frequencies may deviate from values provided.
2. XL = 8, XN = 1 (DDR = 26 MHz) is the reset state used for hardware boot-up frequency only and must not be used for normal operation.

Table 54: PXA32x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 156 MHz)

XL	XN	Frequency (MHz)								
		Run (ACCR[XL])	Turbo (ACCR[XN])	Switch (ACCR[XL])	System Bus (ACCR[HSS])	LCD (ACCR[HSS])	Internal SRAM Controller (ACCR[SFLFS])	DDR Controller SDRAM Clock (ACCR[DMCFS])	Static Memory Controller (ACCR[SMCFS])	Data Flash Controller not programmable
Ring Oscillator[1]		60	—	60	60	60	60	30	15	30
8[2]	1	104 (0x01)	— (0x01)	104 (0x01)	104 (0x00)	104 (0x00)	104 (0x00)	26 (0x00)	78 (0x00)	19.5

Table 54: PXA32x Core PLL, Turbo and Run Mode Output Frequencies (Continued) (ACCR[DMCFS] = 156 MHz)

XL	XN	Frequency (MHz)								
		Run (ACCR[XL])	Turbo (ACCR[XN])	Switch (ACCR[XL])	System Bus (ACCR[HSS])	LCD (ACCR[HSS])	Internal SRAM Controller (ACCR[SFLFS])	DDR Controller SDRAM Clock (ACCR[DMCFS])	Static Memory Controller (ACCR[SMCFS])	Data Flash Controller not programmable
8	1	104 (0x08)	— (0x01)	104 (0x08)	104 (0x00)	104 (0x00)	104 (0x00)	156 (0x01)	78 (0x00	156
16	1	208 (0x10)	— (0x01)	208 (0x10)	104 (0x00)	104 (0x00)	156 (0x01)	156 (0x01)	104 (0x02)	156
16	2	208 (0x10)	416 (0x02)	208 (0x10)	156 (0x01)	156 (0x01)	208 (0x02)	156 (0x01)	104 (0x02)	156
24	2	312 (0x18)	624 (0x02)	312 (0x18)	208 (0x02)	208 (0x02)	312 (0x03)	156 (0x01)	208 (0x05)	156
31	2	403 (0x1F)	806 (0x02)	403 (0x1F)	208 (0x02)	208 (0x02)	312 (0x03)	156 (0x01)	208 (0x05)	156

NOTE:
1. Frequencies shown in this row are ideal-nominal and are based upon the ring oscillator, which has a ±2% tolerance; actual frequencies may deviate from values provided.
2. XL = 8, XN = 1 (DDR = 26 MHz) is the reset state used for hardware boot-up frequency only and must not be used for normal operation.

Copyright © 2009 Marvell

Table 55: PXA31x Core PLL, Turbo and Run Mode Output Frequencies (ACCR[DMCFS] = 130 MHz)

XL	XN	Frequency (MHz)									
		Run (ACCR[XL])	Turbo (ACCR[XN])	Switch (ACCR[XL])	System Bus (ACCR[HSS])	LCD (ACCR[HSS])	Internal SRAM Controller (ACCR[SFLFS])	DDR Controller SDRAM Clock (ACCR[DMCFS])	Static Memory Controller (ACCR[SMCFS])	Data Flash Controller not programmable	Video Accelerator Unit (ACCR[VAUF])
Ring Oscillator[1]		60	—	60	60	60	60	30	15	30	60
8[2]	1	104 (0x01)	— (0x01)	104 (0x01)	104 (0x00)	104 (0x00)	104 (0x00)	26 (0x00)	78 (0x00)	19.5	104 (0x00)
8	1	104 (0x08)	— (0x01)	104 (0x08)	104 (0x00)	104 (0x00)	104 (0x00)	130 (0x03)	78 (0x00	156	78 (0x03)
16	1	208 (0x10)	— (0x01)	208 (0x10)	104 (0x00)	104 (0x00)	156 (0x01)	130 (0x03)	104 (0x02)	156	78 (0x03)
16	2	208 (0x10)	416 (0x02)	208 (0x10)	156 (0x01)	156 (0x01)	208 (0x02)	130 (0x03)	104 (0x02)	156	104 (0x00)
24	2	312 (0x18)	624 (0x02)	312 (0x18)	208 (0x02)	208 (0x02)	312 (0x03)	130 (0x03)	208 (0x05)	156	156 (0x01)
24	2	312 (0x18)	624 (0x02)	312 (0x18)	312 (0x03)	312 (0x03)	312 (0x03)	130 (0x03)	208 (0x05)	156	156 (0x01)
31	2	312 (0x1F)	806 (0x02)	403 (0x18)	208 (0x02)	208 (0x02)	312 (0x03)	130 (0x03)	208 (0x05)	156	156 (0x01)
31	2	312 (0x1F)	806 (0x02)	403 (0x18)	312 (0x03)	312 (0x03)	312 (0x03)	130 (0x03)	208 (0x05)	156	156 (0x01)

NOTE:
1. Frequencies shown in this row are ideal-nominal and are based upon the ring oscillator, which has a ±2% tolerance; actual frequencies may deviate from values provided.
2. XL = 8, XN = 1 (DDR = 26 MHz) is the reset state used for hardware boot-up frequency only and must not be used for normal operation.

Table 56: PXA30x Core PLL, Turbo and Run Mode Output Frequencies

XL	XN	Frequency (MHz)								
		Run (ACCR[XL])	Turbo (ACCR[XN])	Switch (ACCR[XL])	System Bus (ACCR[HSS])	LCD (ACCR[HSS])	Internal SRAM Controller (ACCR[SFLFS])	DDR Controller (ACCR[DMCFS])	Static Memory Controller (ACCR[SMCFS])	Data Flash Controller not programmable
Ring Oscillator[1]		60	—	60	60	60	60	15	15	30
8[2]	1	104 (0x01)	— (0x01)	104 (0x01)	104 (0x00)	104 (0x00)	104 (0x00)	26 (0x00)	78	19.5
8	1	104 (0x08)	— (0x01)	104 (0x08)	104 (0x00)	104 (0x00)	104 (0x00)	130 (0x03)	78	156
16	1	208 (0x10)	— (0x01)	208 (0x10)	104 (0x00)	104 (0x00)	156 (0x01)	130 (0x03)	104	156
16	2	208 (0x10)	416 (0x02)	208 (0x10)	156 (0x01)	156 (0x01)	208 (0x02)	130 (0x03)	104	156
24	2	312 (0x18)	624 (0x02)	312 (0x18)	208 (0x02)	208 (0x02)	312 (0x03)	130 (0x03)	208	156

NOTE:
1. Frequencies shown in this row are ideal-nominal and are based upon the ring oscillator which has a ±2% tolerance; actual frequencies may deviate from values provided.
2. XL = 8, XN = 1 (DDR = 26 MHz) is the reset state used for hardware boot-up frequency only and must not be used for normal operation.

6.7.1 Core Frequency Changes

The core frequency is a multiple of the processor oscillator (13 MHz) and the XL and XN values in the Application Subsystem Clock Configuration Register (ACCR) given by this equation:

*Core frequency = 13 MHz * ACCR[XL] * ACCR[XN]*

There are two methods to change the core frequency depending on which parameters are modified:

- *Run mode frequency change*. Set the F-bit in the *Core Clock Configuration Register (XCLKCFG) (CP14 Register 6) after changing XL and/or XN values.*
- *Turbo mode frequency change*. Set the T-bit in XCLKCFG after changing XN values.

Copyright © 2009 Marvell

Note

Only values of XL and XN that correspond to valid operating points in Table 53 or Table 56 can be programmed. All other combinations are reserved and may result in undefined behavior.

6.7.1.1 Run Mode Frequency Change

Follow this sequence to transition to a new core frequency when XL and/or XN are to be changed:

1. Write the ACCR[XL] (and/or ACCR[XN]) fields with the preferred values.
2. Set the F-bit in the XCLKCFG register.
3. Wait until the new XL and XN values have updated in the Application Subsystem Clock Status Register (ACSR).

This procedure initiates a sequence in which the core pipeline is flushed, core clocks are stopped, the PLL is re-locked with the new XL and XN values, the core is restarted with the new clock frequency, and the ACSR register is updated.

While the PLL is re-locking, the system PLL can be used optionally to clock the core by setting the ACCR[XSPCLK] bit. Use of the system PLL significantly reduces the latency of the normal core frequency change. Use the frequency change interrupt bits in the Application Subsystem Interrupt Control/Status Register (AICSR) to determine when the clocks are switched back to the core PLL after a frequency change that uses the system PLL.

6.7.1.2 Turbo Mode Frequency Change

Follow this sequence to transition to a new core frequency when only XN is to be changed:

1. Write the ACCR[XN] field with the preferred XN value.
2. Set the T-bit in the Core Clock Configuration Register (XCLKCFG) (CP14 Register 6).
3. Wait until the new XN value has updated in the Application Subsystem Clock Status Register (ACSR).

This procedure initiates a sequence in which the core pipeline is flushed, core clocks are stopped, the core PLL output divider is reconfigured with the XN value, and then the core is restarted with the new clock frequency and the ACSR register is updated. New values in ACCR[XL] are ignored during a turbo frequency change.

6.7.2 Peripheral Frequency Changes

Distinct from the changes discussed in Section 6.7.1, Core Frequency Changes, the frequencies of select peripheral clocks on the processor can be changed individually. These frequencies are changed simply by writing to the appropriate fields in the ACCR. Software must wait for the appropriate status field in the ACSR to be updated before proceeding further; otherwise, results are unpredictable. More than one frequency can be changed at a time, but all ACSR fields corresponding to frequency changes must be updated before proceeding further.

Warning

All frequencies have to change in the same direction when software changes the frequency for two or more units at the same time.

The Static Memory Controller (SMC) external clock (DF_SCLK) and timing reference clock frequencies are derived from the SMCFS bit in the Application Subsystem Clock Configuration

Register (ACCR) and the DF_CLKDIV bit in the Clock Configuration Register (MEMCLKCFG). The ACCR[SMCFS] selects the speed that the SMC operates at and the MEMCLKCFG[DF_CLKDIV] divides that clock down to get the reference clock frequency. To ensure specifications are not exceeded, power management software must change both of these registers in specific order depending on whether the SMC controller frequency is increasing or decreasing.

When the SMC controller frequency is increasing, the MEMCLKCFG[DF_CLKDIV] must be programmed to match the higher frequency divisor value before changing the SMC controller frequency (ACCR[SMCFS]). When the SMC controller frequency is decreasing, the MEMCLKCFG[DF_CLKDIV] must be programmed to match the lower frequency divisor after changing the SMC controller frequency (ACCR[SMCFS]).

For more information on the Clock Configuration Register (MEMCLKCFG) refer to the Static Memory Controller chapter in the *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.

6.7.2.1 Coupling Voltage Changes with Operating Point Changes

An operating point frequency change can be used to alter clock frequencies of processor peripherals, including the application core frequency, without affecting the frequency of other processor peripherals. Operating point frequency changes can be coupled with a voltage change to optimize power consumption. Changing VCC_APPS and VCC_SRAM voltages when changing application subsystem frequencies may increase the time between frequency changes due to the additional time needed to modify the voltage. Optimize power consumption versus performance by using the MTS bits in the Application Subsystem Power Status/Configuration Register (ASCR) and the FVE bit in the Power Management Unit Voltage Change Control Register (PVCR).

The MTS bit is used with the XN bit to set voltage levels for each operating point. When both XN and MTS are set to 0x1, the voltage level remains at the voltage level for the current operating point. When XN is set to 0x1 and MTS is set to 0x2, the voltage level is set at the next higher operating point. For more information on the correct voltage levels, refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.

The MTS bit can be used to reduce the latency between operating points for a Turbo mode frequency change in which only the XN value changes. The MTS bit can be used to eliminate the PWR_I^{2}C commands issued by the voltage change sequencer during a Turbo mode change. For example, if the MTS bit is configured to 0x2 while the processor operates in the 208 MHz Run mode (XL = 16, XN = 1, and MTS = 2), the XN value is changed to 0x2, and then the T-bit is set, the voltage change sequencer does not issue the PWR_I^{2}C commands to initiate a voltage change for the 416 MHz Turbo mode (XL = 16, XN = 2 and MTS = 2). Power management software can change between 208 MHz run mode and 416 MHz Turbo mode without the delay of changing voltage levels. The voltage levels remain at the 416 MHz operating point level when changing back to 208 MHz run mode with MTS = 2.

Note

When the operating point frequencies are raised, the voltage must be raised first and then the clocks switched to the selected higher frequency. Conversely, when the operating point frequencies are lowered, the clocks must first be switched to the new lower frequency and then the voltage lowered.

VCC_APPS and VCC_SRAM are set to the highest operating voltage when the processor exits a hardware reset. Power management software must issue a frequency change to a valid operating point with the correct voltage levels.

When the PVCR[FVE] bit is set and software initiates an operating point frequency-change sequence, the PWR_I^{2}C initiates a VCC_APPS and VCC_SRAM voltage-change sequence if the

Copyright © 2009 Marvell

current voltage is not the required voltage. Software must handle the voltage changes if FVE is cleared.

The following sequence is required to change VCC_APPS and VCC_SRAM voltage levels during operating point changes while PVCR[FVE] is set:

1. Program the ASCR[MTS] bit to match the ACCR[XN] value for the next operating point.
2. Write to the ACCR, configuring XL and/or XN to the required values. When operating points are increased, the APMU initiates a voltage-change sequence to increase VCC_APPS and VCC_SRAM supplies to the voltage required by the new XL and XN settings in the ACCR and MTS settings in the ASCR. When software is used to change the voltages, it must change the voltage level and wait until the voltages ramp to the correct levels.
3. Initiate the operating point frequency change.
   - Set the F-bit in the Core Clock Configuration Register (XCLKCFG) (CP14 Register 6) when changing the XL and/or XN values.
   - Set the T-bit in the Core Clock Configuration Register (XCLKCFG) (CP14 Register 6) when changing only the XN values.

   The application core PLL can be programmed to use the new XL and XN values when the new voltages are achieved (after the PWR_I^2C commands are sent and the PCFR[LPM_DEL] timer expires). When operating points are increased, the processor halts software execution until the PCFR[LPM_DEL] expires.
4. Program the SMCFS, SFLFS, HSS or DMCFS fields in the ACCR. When operating points are decreased, the APMU initiates a voltage-change sequence to decrease VCC_APPS and VCC_SRAM supplies to the voltage required by the new XL and XN settings in the ACCR and MTS settings in the ASCR. When used to change the voltages, software must change the voltage level and wait until the voltages ramp to the correct levels.

   When operating points are decreased, the processor does not wait for the external voltage levels to decrease before executing code. The processor does not wait for the PCFR[LPM_DEL] delay to expire before allowing code to start executing, which saves time when operating points are lowered.
5. The processor switches the application subsystem core clocks to the system PLL if it is selected. Otherwise, the application subsystem core clocks are disabled.

   The XSPCLK bit in the ACCR is used to select the clocking source during frequency changes.
6. The Application Subsytem Clock Control Unit switches the application core clocks to the application core PLL clock outputs.
7. The frequency change sequence exits at the new voltage and frequency.

Note

Refer to Table 53 and Table 56 for correct ACCR register values.

Table 57 shows the latency times during changes among the different operating points.

Table 57: Operating Point Change Latencies

Operating Point Change[1]	PVCR[FVE]	ACCR [XSPCLK]	XCLKCFG[F]	XCLKCFG[T]	Change Time (µs)
Increasing Operating Points	0[2]	0	1	0	27.2
		1			2.2
	1[3]	0			227.2[6,7]
		1			202.2[6,7]
Decreasing Operating Points	0[2]	0	1	0	27.2
		1			2.2
	1[3]	0			27.2[8]
		1			2.2[8]
Turbo Mode Change[4]	0[2]	0	0	1	2.2
		1			2.2
	1[3]	0			2.2[9]
		1			2.2[9]

1. Supported operating points are defined in Table 53 and Table 56.
2. No hardware driven PWR_I^{2}C commands are sent out. Power management software must ensure that the power domains are are the correct voltage levels.
3. Hardware driven PWR_I^{2}C commands are sent out to change the power domains to the appropriate voltage levels.
4. For a Turbo mode change, the ASCR[MTS] = 0x2 with only the XN value changing for the operating point change. An example would be a 208 MHz (XL=16;XN=1;MTS=2) to 416 MHz (XL=16;XN=2;MTS=2) operating point change.
5. Times do not include the Power Management IC voltage ramp time (PCFR[LPM_DEL] = 0). Power management software must program the PCFR[LPM_DEL] to meet the ramp time requirements for the PMIC.
6. Time does not include the time for the PMIC to ramp the voltage PCFR[LPM_DEL]).
7. Time includes delay for the PWR_I2C commands to be issued to the PMIC (~200 µs).
8. When decreasing operating points, code execution begins at the same time the PWR_I2C commands are issued. Code execution is not halted for voltage ramp delays.
9. Time does not include any delays for voltage sequencing delays.

6.8 Voltage Management

The processor controls the external power management I/C (PMIC) device through a dedicated power I^{2}C[1] (PWR_ I^{2}C) module in the services unit that communicates with the external regulators. The voltage manager provides the following hardware-controlled voltage change sequences when PVCR[PVE] and PVCR[FVE] are set:

- For VCC_APPS and VCC_SRAM voltage changes coordinated with the application subsystem frequency changes to optimize power consumption.
- For configuring and enabling VCC_APPS and VCC_SRAM during SOD and low-power mode entry and exit.

The voltage manager consists of two primary components:

1. Future docs will refer to I^{2}C as the Two-Wire Serial Interface (TWSI).

Copyright © 2009 Marvell

- Dedicated I^2C module (PWR_I^2C)
- Command sequencer

The dedicated I^2C module is nearly identical to the I^2C described in I^2C bus interface unit chapter. The PWR_I^2C is located in the services unit and is optimized and dedicated for connection to the external PMIC only.

6.8.1 Programming Restrictions for the PWR_I^2C

The PWR_I^2C is a full-function I^2C module capable of all normal operations, including Receive and Transmit operation, but it is dedicated to power-management-initiated transfers. PWR_I^2C supports fast-mode operation of up to 380 Kbps. When used with the voltage-change sequencer, PWR_I^2C operates in fast-mode. The PWR_I^2C is always enabled during the SOD sequences. The voltage-change sequencer sends commands under hardware control during SOD, application core frequency changes, and power-mode changes. When the PWR_I^2C sends commands, the PVCR[VCSA] bit is set and the PWR_I^2C registers are not writable and read unknown values. Software checks the VCSA bit before reading or writing the PWR_I^2C and reads registers following a Write to ensure that the Read occurred.

Note

When the voltage-change sequencer is operating, Writes to the PVCR result in unpredictable operation.

6.8.2 External Voltage Regulator Requirements

Preserving functionality during any voltage change requires a special external voltage regulator. The regulator must have these features to be compatible with all voltage management features:

- Implement required registers
- Programmable buck converter output to drive VCC_APPS and VCC_SRAM with 32 output levels between the voltages 0.725–1.50 V
- I^2C controlled voltage level selection and voltage enable control
  - Limited overshoot and undershoot during voltage change

6.8.3 Hardware-Controlled Voltage-Change Sequencer

The processor contains a voltage-change sequencer, which automatically sends hard-coded command sequences to the external regulator when triggered by the following events:

- Start of day reset
- Operating point changes that require a change in voltage levels; refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)* for required voltage levels for each operating point
- Power mode changes
  - S3/D4/C4 entry due to nBATT_FAULT assertion

The sequencer can send hard-coded commands to enable the VCC_APPS and VCC_SRAM supplies and control the output voltage and voltage ramping of VCC_APPS and VCC_SRAM. The hardware-controlled sequencer can be disabled using the FVE and PVE bits in the PVCR.

The PWR_I^2C sends all hard-coded commands using a multi-byte command sequence with the following format:

START	Slave Address	R/W# 0	ACK	Register A Address	ACK	Register A Data	ACK	Register N Address	ACK	Register N Data	ACK	Register X Address	ACK	Register X Data	ACK	STOP

Registers A, N, and X do not have to be contiguous address values. The PWR_I^{2}C may send as many as 8 register address/data pairs. The PWR_ I^{2}C sends the slave address only one time and does not send a STOP until all register address/data pairs are sent.

Table 58 lists the default values that the voltage sequencer sends to the PMIC to set voltage levels when changing between different operating points or power modes. The values for these PWR_I^{2}C commands can be changed by writing to the VCC_APPS Voltage Control Register (AVCR) and VCC_SRAM Voltage Control Register (SVCR). For more information on the required voltage levels refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.

Table 58: Hardware PWR_I^{2}C command encoding

Supply	Voltage Encoding			
	Level 0	Level 1	Level 2	Level 3[9]
VCC_APPS	0x0B[1]	0x0F[2]	0x1A[3]	0x1B[4]
VCC_SRAM	0x0F[5]	0x13[6]	0x1B[7]	0x1B[8]

NOTE:
1. Defined as VCC_APPS_1 in the EMTS
2. Defined as VCC_APPS_2 in the EMTS
3. Defined as VCC_APPS_3 in the EMTS
4. Defined as VCC_APPS_4 in the EMTS
5. Defined as VCC_SRAM_1 in the EMTS
6. Defined as VCC_SRAM_2 in the EMTS
7. Defined as VCC_SRAM_3 in the EMTS
8. Defined as VCC_SRAM_4 in the EMTS
9. Level 3 is the default reset value at SOD and is not used for normal operation. Software must change operating points at SOD with PVCR[FVE] = 1 or change the voltage to a correct voltage level.

6.8.4 Start-of-Day (SOD) Operation

The PWR_I^{2}C and voltage-change sequencer enable and initially configure the power supplies when triggered during the SOD sequence. The SPMU controls the activation of the PWR_I^{2}C during the SOD sequence. The PMIC must be compatible with the processor to work with the hard-coded commands and external power control signals (PWR_EN and SYS_EN).

Note

If no PMIC is included in the system or cannot respond to the control registers, the PWR_I^{2}C times out and continues the SOD sequence without completing the PWR_I^{2}C commands.

6.8.5 Accessing PWR_I^{2}C Registers Directly through Software

Software must ensure that the code executes in single-thread mode and that all interrupts (except critical ones such as Batt Fault detected) are disabled before the following sequence is started. Such a process prevents another section of software or the hardware-driven power sequencer (PVCR[FVE] = 1 and PVCR[PVE] = 1) from causing a PMIC event, such as low-power mode entry or a GPIO reset in the middle of the transmission.

Software must perform the following steps to access the PWR_I^{2}C registers:

Copyright © 2009 Marvell

1. Read the PMER register and verify that the value read is 0x0 (if not keep looping).
   - This step ensures that SPMU is not attempting to service any of the asynchronous events: nBATT_FAULT
   - GPIO reset
2. Write 0x0F to PMER to mask nBATT_FAULT and GPIO resets.
3. Read the PMER register until the value read is 0xF to ensures that the masking of asynchronous events is complete.
4. Wait for ~100µs to allow any pending PWR_I^2C requests in services to go through.
5. Read the PVCR[VCSA] bit and verify that the value read is 0x0 to ensure that there are no PWR_I^2C command sequences in progress or that the PWR_I^2C module is idle. If the value read is non-zero, then continue looping until the bit is cleared.
6. Perform the required software access to the PWR_I^2C module.
7. Write 0x0 to PMER to enable execution of nBATT_FAULT and GPIO reset when they are detected.

See the PWR_I^2C chapter for details on the PWR_I^2C registers that can be controlled/accessed through this procedure.

Note

PMER is a reserved system register and must not be written to or read from outside the scope of the PWR_I^2C software access procedure presented here. GPIO resets must be masked during software PWR_I2C accesses to ensure the I^2C commands are complete before the reset occurs.

6.9 Internal Memory SRAM Power Management

The internal memory controller has up to six banks of SRAM arrays that can be controlled independently for different power modes to optimize power consumption. Figure 9 shows the six internal SRAM banks along with the address ranges for each.

Figure 11: Organization and Memory Mapping of SRAM Arrays

Upper Bank	Lower Bank
Upper Bank Array 2 0x5C0A_0000 to 0x5C0B_FFFF (PXA32x Only)	Lower Bank Array 2 0x5C08_0000 to 0x5C09_FFFF (PXA32x Only)
Upper Bank Array 1 0x5C06_0000 to 0x5C07_FFFF (PXA32x Only)	Lower Bank Array 1 0x5C04_0000 to 0x5C05_FFFF (PXA32x Only)
Upper Bank Array 0 0x5C02_0000 to 0x5C03_FFFF	Lower Bank Array 0 0x5C00_0000 to 0x5C01_FFFF

The SRAM arrays can be configured as follows:

- Normal: SRAM arrays can be accessed immediately.
- State retaining: the SRAM arrays are in a state retentive low-power mode
- Off: the SRAM arrays are powered off and their contents are lost.

6.9.1 Power Mode States

For each application subsystem low-power state (D1, D2, and D3 which correspond to power modes S0/D1/C2, S0/D2/C2 and S2/D3/C4), each SRAM array can be either powered off or in state-retaining mode. The following registers are used to select the state of each SRAM bank in the power modes:

- Application Subsystem D3 Configuration Register (AD3R)
- Application Subsystem D2 Configuration Register (AD2R)
- Application Subsystem D1 Configuration Register (AD1R)

6.9.2 Auto Power Down

The Internal Memory Power Management Control Register (IMPMCR) is used to put the upper bank of SRAM into a state-retentive mode after a time period specified in the IMPMCR[DDT] register elapses since the last access to the SRAM array. This state-retentive mode saves power but creates

Copyright © 2009 Marvell

a delay for the next access while the SRAM bank exits this mode. To wake up from a state retentive mode, the Upper Bank Array 0 wakeup enable (IMPMCR[AW1]) bit must also be set. For details on the IMPMCR register and the auto-powerdown feature, refer to the internal memory chapter in the *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual*.

6.10 Power Management Operation

Power management software must manage overall processor power consumption. The software manages power modes, clock sources and gating, and operating points. To optimize power and performance, each device driver uses the software to control clocks and operating points. Power management software uses the processor performance monitoring registers to determine processor usage and changes power modes and operating points depending on the drivers used and the resources required. For details on the performance monitoring registers, refer to the performance monitoring chapter.

6.10.1 Configuring Hardware Driven PWR_I^{2}C commands

The processor supports a hardware feature that sends out I^{2}C commands directly to the power management IC to change voltage levels. Power management software must enable this feature by setting the FVE and PVE bits in the Power Management Unit Voltage Change Control Register (PVCR).

The voltage levels for VCC_APPS and VCC_SRAM are hardcoded in VCC_APPS Voltage Control Register (AVCR) and VCC_SRAM Voltage Control Register (SVCR). Power Management software must program these registers for the appropriate levels as defined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. For more information on the PWR_I^{2}C operation and voltage management, refer to Section 6.8, Voltage Management.

6.10.2 Preparing for Power Mode Transitions

After a reset or a wakeup event to enter S0/D0/C0, power management software must prepare for transitioning among all other power modes. Software performs the following steps after entering S0/D0/C0 to enable all power mode transitions:

1. The following registers are used to determine the previous power mode or reset source. Power management software can use these registers to determine what actions are needed after entering S0/D0/C0:
   - Application Subsystem Reset Status Register (ARSR)
   - Power Management Unit Status Register (PSR)
   - Power Manager Wake-Up Status Register (PWSR)
   - Power Manager EXT_WAKEUP<1:0> Control Register (PECR)
   - Application Subsystem Wakeup from D2 to D0 Status Register (AD2D0SR)
   - Application Subsystem Wakeup from D1 to D0 Status Register (AD1D0SR)
   - Application Subsystem Power Status/Configuration Register (ASCR)
2. Depending on the S0/D0/C0 entry source, clear the ARSR, PSR, PWSR, PECR, AD2D0SR, AD1D0SR, or ASCR status registers.
3. To enter S0/D0/C0 from any power mode, enable all wakeup sources as defined in Table 44 using the following registers as this optimizes transitions between different power modes so that power management software does not have to enable these when preparing for the power modes.
   - Enable sources for S2/D3/C4 and S3/D4/C4 wakeups in the Power Manager Wake-Up Enable Register (PWER)
   - Configure EXT_WAKEUP<1:0> pin for wakeups in the Power Manager EXT_WAKEUP<1:0> Control Register (PECR)

- Configure the Power Manager Mask Event Register (PMER)
- Enable sources for S2/D3/C4 wakeups in the Application Subsystem Wakeup from D3 Enable Register (AD3ER)
- Enable sources for S0/D2/C2 wakeups in the Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER)
- Enable sources for S0/D1/C2 wakeups in the Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER)

4. Enable sources for S0/D2/C2 to S0/D1/C2 wakeups in the Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER).
5. Enable the mini-LCD wakeup events for S0/D1/C2 to S0/D2/C2 and S0/D1/C2 to S0/D0/C0 transitions in the MLFRMCNT register inside the mini-LCD controller, as follows:
   a) Program the FRCOUNT bits with the number of frames to be displayed by the mini-LCD controller before exiting S0/D1/C2. Do not clear the FRCOUNT bit.
   b) Set the WKUP_EN bit to enable S0/D1/C2 to S0/D0/C0 wakeups.
   c) Set the FKUP_EN to enable S0/D1/C2 to S0/D2/C2 wakeups. If WKUP_EN and D0CKEN_B[CKEN17] are not enabled prior to entering S0/D1/C2, the system cannot exit.
6. Set the S2/D3/C4 operational state for the internal SRAM banks by programming the Application Subsystem D3 Configuration Register (AD3R).
7. Set the S0/D2/C2 operational state for the internal SRAM banks by programming the Application Subsystem D2 Configuration Register (AD2R).
8. Set the S0/D1/C2 operational state for the internal SRAM banks by programming the Application Subsystem D1 Configuration Register (AD1R).
9. Enable interrupts to wake up from S0/D0/C1 to S0/D0/C0 by programming the Interrupt Controller Mask Registers (ICMR and ICMR2).
10. Configure the multi-function pins (GPIO) for power mode operation, including internal pullup and pulldown resistors for minimal power consumption. Refer to the Pins and Control Chapter for details on configuring the MFPRx registers.
11. Configure the SL_ROD bit in the Power Management Unit General Configuration Register (PCFR) to enable/disable nRESET_OUT assertion.

6.10.3 Boot ROM Requirements

The processor has an internal Boot ROM that handles the boot processes as well as resuming from resets and S2/D3/C4 and S3/D4/C4 power modes. There are specific requirements within the Boot ROM for each type of reset defined in Section 6.12, Resets as well as wakeups from S2/D3/C4 and S3/D4/C4 power modes. Refer to the *PXA3xx Processor Family and Tavor Processor Boot ROM Reference Manual* for specific details on these requirements.

6.11 Wakeup Events and Sources

Wakeup events cause a transition from a low-power mode to the Run mode (S0/D0/C0) or another low-power mode. There are three main sets of wakeup sources: services unit, peripheral controller, and external pins. A wakeup in this context is not necessarily the same as an interrupt. Table 59 lists all wakeup sources for the processor.

Copyright © 2009 Marvell

Table 59: Processor Wakeup Events

Unit	Events or Alternate Functions	Wakeup Detection Type	Slave Power Management Unit Dedicated Register Bits	Notes
External Wakeups	Non-maskable wakeup event: SOD and GPIO Reset	Services first-level wakeup source	—	
External Wakeups	EXT_WAKEUP<0>: Communicated by the services unit to the Application Subsystem Power Management Unit (APMU).	Services first-level wakeup source	ADxER[0] and ADxSR[0]	
	EXT_WAKEUP<1>: Communicated by the services unit to the Application Subsystem Power Management Unit (APMU).		ADxER[1] and ADxSR[1]	PXA32x Only
Reserved	—	—	ADxER[1] and ADxSR[1]	PXA31x and PXA30x Only
AC97	AC97_BITCLK, AC97_SDATA_IN0, AC97_SDATA_IN1	Pin control unit first-level wakeup source	ADxER[2] and ADxSR[2] (generic_wakeup[0])	
SSP1	SSPSCLK, SSPSFRM, SSPRXD, and SSPEXTCLK	Pin control unit first-level wakeup source	ADxER[3] and ADxSR[3] (generic_wakeup[1])	
SSP2	SSPSCLK2, SSPSFRM2, SSPRXD2, and SSPEXTCLK2	Pin control unit first-level wakeup source	ADxER[4] and ADxSR[4] (generic_wakeup[2])	
UART1	UART1_RXD, UART1_CTS, UART1_RI, UART1_DSR, and UART1_DCD	Pin control unit first-level wakeup source	ADxER[5] and ADxSR[5] (generic_wakeup[3])	
	ULPI_OTG_INTR when GPIO 33 is configured as alternate function #4 (for ULPI Serial mode and ULPI CarKit mode)			(PXA31x Only)
UART2	UART2_RXD and UART2_CTS	Pin control unit first-level wakeup source	ADxER[6] and ADxSR[6] (generic_wakeup[4])	
UART3	UART3_RXD and UART3_CTS	Pin control unit first-level wakeup source	ADxER[7] and ADxSR[7] (generic_wakeup[5])	

Table 59: Processor Wakeup Events (Continued)

Unit	Events or Alternate Functions	Wakeup Detection Type	Slave Power Management Unit Dedicated Register Bits	Notes
Matrix Keypad	KP_MKIN<7:0>	Pin control unit first-level wakeup source	ADxER[8] and ADxSR[8] (generic_wakeup[6])	
SSP3	SSPSCLK3, SSPSFRM3 and SSPRXD3	Pin control unit first-level wakeup source	ADxER[9] and ADxSR[9] (generic_wakeup[7])	
SSP4	SSPSCLK4, SSPSFRM4 and SSPRXD4	Pin control unit first-level wakeup source	ADxER[10] and ADxSR[10] (generic_wakeup[8])	
I^2C	I^2C_SCL and I^2C_SDA	Pin control unit first-level wakeup source	ADxER[11] and ADxSR[11] (generic_wakeup[9])	PXA31x and PXA30x Only
—	Reserved	Reserved		PXA32x Only
MMC1	MM1_DAT0, MM1_DAT1, MM1_DAT2, MM1_DAT3 and MM1_CMD	Pin control unit first-level wakeup source	ADxER[12] and ADxSR[12] (generic_wakeup[10])	
MMC2	MM2_DAT0, MM2_DAT1, MM2_DAT2, MM2_DAT3 and MM2_CMD	Pin control unit first-level wakeup source	ADxER[13] and ADxSR[13] (generic_wakeup[11])	
Timers	EXT_SYNC0	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA31x and PXA30x Only
Timers	EXT_SYNC1	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA31x and PXA30x Only
CLK_EXT	CLK_EXT	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA31x and PXA30x Only
GPIO	GPIO0	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO1	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO2	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO3	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA32x Only
GPIO	GPIO4	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA32x Only
GPIO	GPIO9	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA32x Only
GPIO	GPIO53	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	

Copyright © 2009 Marvell

Table 59: Processor Wakeup Events (Continued)

Unit	Events or Alternate Functions	Wakeup Detection Type	Slave Power Management Unit Dedicated Register Bits	Notes
GPIO	GPIO76	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO83	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO127	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO0_2	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
GPIO	GPIO1_2	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	
MMC3	MM3_DAT0, MM3_DAT1, MM3_DAT2, MM3_DAT3 and MM3_CMD	Pin control unit second-level wakeup source	ADxER[14] and ADxSR[14] (generic_wakeup[12])	PXA31x Only
NAND Controller	ND_RnB	Pin control unit first-level wakeup source	ADxER[15] and ADxSR[15] (generic_wakeup[13])	PXA31x and PXA30x Only
NAND Controller	ND_RnB	Pin control unit second-level wakeup source	ADxER[15] and ADxSR[15] (generic_wakeup[13])	PXA32x Only
I2C	I2C_SCL and I2C_SDA	Pin control unit second-level wakeup source	ADxER[15] and ADxSR[15] (generic_wakeup[13])	PXA32x Only
Timers	EXT_SYNC0	Pin control unit second-level wakeup source	ADxER[15] and ADxSR[15] (generic_wakeup[13])	PXA32x Only
Timers	EXT_SYNC1	Pin control unit second-level wakeup source	ADxER[15] and ADxSR[15] (generic_wakeup[13])	PXA32x Only
CLK_EXT	CLK_EXT	Pin control unit second-level wakeup source	ADxER[15] and ADxSR[15] (generic_wakeup[13])	PXA32x Only
USBOTG	USBOTG_P and USBOTG_N	Pin control unit first-level wakeup source	ADxER[16] and ADxSR[16]	PXA32x and PXA30x Only
Reserved	—	—	ADxER[17] and ADxSR[17]	
Reserved	—	—	ADxER[18] and ADxSR[18]	PXA32x Only
Mini-LCD	Mini-LCD D1->D0 Wakeup Event	Peripheral controller internal first-level wakeup source.	AD1D0ER[18] and AD1D0SR[18]	PXA31x and PXA30x Only
CARD1	USIM_DETECT and USIM_IO	Pin control unit first-level wakeup source	ADxER[19] and ADxSR[19] (usim0_wakeup)	
CARD2	SC_DETECT and SC_IO	Pin control unit first-level wakeup source	ADxER[20] and ADxSR[20] (usim1_wakeup)	

Table 59: Processor Wakeup Events (Continued)

Unit	Events or Alternate Functions	Wakeup Detection Type	Slave Power Management Unit Dedicated Register Bits	Notes
Direct Keypad	KP_DKIN<7:0>	Pin control unit first-level wakeup source	ADxER[21] and ADxSR[21] (kp_wakeup)	
USB Host Port 2 and UDC	USB_P2_1, USB_P2_2, USB_P2_3, USB_P2_4, USB_P2_5, USB_P2_7	Pin control unit first-level wakeup source	ADxER[22] and ADxSR[22] (USB Host Port 2 and USB Device Controller)	PXA32x and PXA30x Only
USB Host Port 2	USB_P2_1, USB_P2_3, USB_P2_4, USB_P2_5, USB_P2_6, and OTG_INTR (On GPIO33 Alternate function 1)	Pin control unit first-level wakeup source	ADxER[22] and ADxSR[22] (USB Host Port 2	PXA31x Only
USB Host Port 3	USB_P3_1, USB_P3_3, USB_P3_4, USB_P3_5 and USB_P3_6	Pin control unit first-level wakeup source	ADxER[23] and ADxSR[23] (otg_port3_wakeup)	PXA32x Only
	USB_P3_1, USB_P3_4, USB_P3_5 and USB_P3_6			PXA30x Only
	USB_P3_1, USB_P3_2, USB_P3_3, USB_P3_4, USB_P3_5 and USB_P3_6			PXA31x Only
Reserved	—	—	ADxER[24] and ADxSR[24]	
Reserved	—	—	ADxER[25] and ADxSR[25]	
ULPI	ULPI_DATA0, ULPI_DATA1, ULPI_DATA3, ULPI_NXT, ULPI_CLK, ULPI_DIR	Pin control unit first-level wakeup source	ADxER[26] and ADxSR[26] (usb2_wakeup)	PXA31x Only
UTMI	UTM_RXVALID, UTM_RXACTIVE, UTM_CLK, UTM_LINESTATE<1:0>			PXA32x and PXA30x Only
Reserved	—	—	ADxER[27] and ADxSR[27]	
USB Host Port 1	USBH1_P and USBH1_N	Pin control unit first-level wakeup source	ADxER[28] and ADxSR[28]	PXA32x and PXA30x Only
TSI	TSI_XP, TSI_XM, TSI_YP, TSI_YM	Pin control unit first-level wakeup source	ADxER[29] and ADxSR[29]	PXA32x Only

Copyright © 2009 Marvell

Table 59: Processor Wakeup Events (Continued)

Unit	Events or Alternate Functions	Wakeup Detection Type	Slave Power Management Unit Dedicated Register Bits	Notes
OS Timers	Five OS timers are generated inside the APMU, and OR'ed to a single wakeup event	Peripheral controller internal first-level wakeup source.	ADxER[30] and ADxSR[30]	
RTC	RTC Alarm: Communicated by Services Unit to APMU.	Services first-level wakeup source.	ADxER[31] and ADxSR[31]	

6.11.1 Services Wakeups

Services wakeups are separate wakeup signals that go directly into the services module. These wakeup signals include the EXT_WAKEUP<1:0> signal and internal RTC alarm wakeups controlled through the following registers:

- Power Manager Wake-Up Enable Register (PWER)
- Power Manager Wake-Up Status Register (PWSR)
- Power Manager EXT_WAKEUP<1:0> Control Register (PECR).

Table 59 summarizes the services wakeups, which are considered first-level wakeup sources because only one register, ADxSR, is read to determine the cause of the wakeup.

Note

If EXT_WAKEUP<1:0> is toggled when the processor enters a low-power mode, the corresponding interrupts are enabled. The system receives interrupts due to the external wakeup toggle regardless of which wakeup source actually caused a wakeup.

6.11.2 Peripheral Controller Wakeups

Two peripheral controller wakeup events including OS Timers and mini-LCD (S0/D1/C2→ S0/D0/C0) are controlled by the following registers:

- Application Subsystem Wakeup from D3 Enable Register (AD3ER)
- Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER)
- Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER)
- Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER)

Table 59, Processor Wakeup Events, on page 201 summarizes the peripheral controller wakeups. They are considered first-level wakeup sources because only one register, ADxSR, is read to determine the cause of the wakeup.

6.11.3 Pin Control Wakeups

There are two types of wakeup signals—first-level and second-level generic—to generate a pin-control wakeup event. These wakeup signals depend on the detection of edges on the multi-function input pins. Each pin has a programmable asynchronous edge detector that can detect rising or falling edges. The edge is detected and latched in the pin and then sent to the power management unit. Multiple signals are combined, so one pin can create a single wakeup event.

Copyright © 2009 Marvell

6.11.3.1 Edge Detect Operation

Some of the multi-function I/O pins are connected to edge-detection logic. This logic is masked unless a pin is configured to a specific function from a list of predefined functions. The edge-detection logic can detect a rising or falling edge on the input path of the pin, and it monitors a pin at all times so that a transition triggers detection. The detection is controlled by the following bits in the Multi-Function Pin Registers (MFPRx):

- EDGE_CLEAR enables and disables the function.
- EDGE_RISE_EN, if enabled, allows detection of rising-edge transactions.
- EDGE_FALL_EN, if enabled, allows detection of falling-edge transactions.

These bits are available only for multi-function pins connected to edge-detection logic.

Note

The output of the edge-detection logic is mainly used for wakeup events in the D1, D2, or D3 power modes and is *not* connected to the edge-detection function of the GPIO controller.

Both rising and falling edge detection can be enabled so that there is an event whenever the signal changes state, even briefly. An event is a level signal that is asserted as long as the edge_clear bit is not set. If the edge-detection logic is to capture an event, all of the following conditions must be met in the order specified:

1. The multi-function I/O pin has at least one function that can wake up the processor. The pin is configured to select this function.

   Table 59 lists the functions that can wake up the processor. In addition, tables in the pins and control chapter show which functions are available on which I/O pin.
2. Either EDGE_RISE_EN, EDGE_FALL_EN or both bits of the relevant MFPR register are set.
3. The value driven on the pin by the external device should change from 0 to 1 or from 1 to 0. The new value should be kept for at least 10 ns. The type of transition (from 0 to 1 or from 1 to 0) must match the type that is configured in the EDGE_RISE_EN and EDGE_FALL_EN bits.

Software clears the edge-detection event by setting the EDGE_CLEAR bit.

6.11.3.2 Generic Wakeups

The generic wakeup signals are controlled by the WE_GENERIC field in the following registers:

- Application Subsystem Wakeup from D3 Enable Register (AD3ER)
- Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER)
- Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER)
- Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER).

6.11.3.3 First-Level Wakeups

Table 59, Processor Wakeup Events, on page 201 summarizes the first-level wakeup sources. They are considered first-level wakeup sources because only one register, ADxSR, is read to determine the source of the wakeup.

For example, suppose that after the processor wakes up from S2/D3/C4 mode, software reads the AD3SR register and finds AD3SR[WE_GENERIC7] set. This situation indicates that the source of the wakeup is activity on a multi-function pin assigned to the SSPSCLK3, SSPSFRM3, or SSPRXD3 alternate functions of the SSP3 controller. While it is evident that the SSP3 controller caused the wakeup, it is difficult to determine exactly which alternate function or combination of these functions is the source of the wakeup.

Copyright © 2009 Marvell

The three alternate functions associated with ADxSR[WE_GENERIC7] wakeup are SSPSCLK3, SSPSFRM3, and SSPRXD3. Table 20 and Table 21 in the pins and control chapter show that these functions are available on several multi-function pins, as follows:

- SSPSCLK3 on GPIO68 and GPIO91.
- SSPSFRM3 on GPIO69 and GPIO92.
- SSPRXD3 on GPIO70, GPIO71, GPIO93, and GPIO94.

If alternate function SSPSCLK3 is assigned to GPIO68, then GPIO68 becomes a wakeup and GPIO91 is *not* a wakeup because SSPSCLK3 is not currently assigned to GPIO91. This same principle applies to the other two alternate functions and their associated multi-function pins.

6.11.3.4 Second-Level Generic Wakeups

Generic wakeup ADxxR[WE_GENERIC12] is considered a second-level wakeup source because multiple wakeup sources are assigned to it. Two registers must be read to determine the cause of the wakeup. This second-level wakeup source uses a combination of WE_GENERIC12 and a Multi-Function Pin Monitor Register (MFP_MONITOR) to determine the cause of the wakeup. MFPRxx[EDGE_CLEAR] must be cleared to enable the MFP_MONITOR register. Table 59, Processor Wakeup Events, on page 201 lists the second-level wakeup sources. Refer to the performance monitoring chapter for details on the MFP_MONITOR register.

For example, assume that after the processor wakes up from S2/D3/C4 mode when EXT_SYNC<1:0> is asserted, software reads the AD3SR register and finds that AD3SR[WE_GENERIC12] set does not provide sufficient information to determine the source of the wakeup. It must also read the MFP_MONITOR register.

For the PXA32x processor, generic wakeup 13 (ADxxR[WE_GENERIC13] is also a second-level wakeup, which uses a combination of ADxxR[WE_GENERIC13] and GPIO Pin Level Register 1 (GPLR1) to determine the cause of the wakeup. Table 60 shows the wakeup sources associated with ADxxR[WE_GENERIC13] and the GPIO pins associated with those wakeup sources.

Table 60: PXA32x Processor Generic Wakeup[13] Sources

Unit	Wakeup Event	GPIO Pin
I2C	I2C_SCL, I2C_SDA	GPIO59
OST	EXT_SYNC0, EXT_SYNC1, CLK_EST	GPIO56
NAND Controller	DF_INT_RNB	GPIO62

6.12 Resets

The processor can be reset by one of three reset sources from external pins (nRESET, nGPIO_RESET, or the operating system timers (OST). The types of reset discussed in this section are as follows:

- Power-on reset (start-of day)
- Hardware reset
- GPIO reset
- Global watchdog reset
- S3/D4/C4 low-power mode exit reset

The Application Subsystem Reset Status Register (ARSR) and Power Management Unit Status Register (PSR) record the type of the reset. Table 61 lists the registers used to determine the source

of each reset. A reset state forces internal registers into a pre-defined state. As seen in Table 62, every type of reset has different effects on the internal register state of the processor hardware unit.

Table 61: Reset Status Registers

Reset	Reset Status Register	Description
GPIO	ARSR[GPR][1]	A GPIO reset has occurred since the last time software or a hardware or power-on reset cleared this bit.
Watchdog	ARSR[WDT][1]	A watchdog reset has occurred since the last time software or a hardware or power-on reset cleared this bit.
Hardware/POR	ARSR[HWR][2]	A hardware or power-on reset has occurred since the last time software cleared this bit.
S3/D4/C4 Exit	ARSR[LPMR][2], PSR[BFS][3], PSR[SS3S][4]	The ARSR[LPMR] bit is set to indicate an S3/D4/C4 exit after a software nBATT_FAULT entry.

NOTE:
1. Write 0b1 to clear. Return to reset states with hardware or power-on resets.
2. Write 0b1 to clear. Not cleared by hardware or power-on resets.
3. If the processor enters S3/D4/C4 due to nBATT_FAULT, the Power Management Unit Status Register (PSR) BFS bit and the Application Subsystem Reset Status Register (ARSR) LPMR bit are both set.
4. If the processor enters S3/D4/C4 due to a software write to PWRMODE[M], then the Power Management Unit Status Register (PSR) SS3S bit and the Application Subsystem Reset Status Register (ARSR) LPMR bit are both set.

Table 62: Effect of Each Type of Reset on Internal Register State

Unit	Deep Sleep-exit Reset	GPIO Reset	Watchdog Reset	Hardware Reset†
Core	Reset	Reset	Reset	Reset
Memory controller	Reset	Reset all registers except configuration registers (memory refresh maintained)	Reset	Reset
Other internal functions	Reset	Reset	Reset	Reset
RTC	Preserved	Preserved	All reset except RTTR	Reset
Keypad interface	All keypad register states are reset but can act as wakeup events	Reset	Reset	Reset
MF pins	Multi-function pins retain state and all pins can act as wakeups	Reset	Reset	Reset

† The power-on reset state is the same as the hardware-reset state.

6.12.1 Power-on Reset

Power-on reset is invoked when a positive transition is detected on the backup battery pin VCC_BBATT after the pin is at a 0 V (completely powered down) level. When VCC_BBATT is

Copyright © 2009 Marvell

asserted, all units reset to their default reset state. Power-on reset cannot be entered from any other mode because the device must be completely powered down first. If power is removed from VCC_BBATT, all other power supplies are disabled.

Note

The low-voltage power supplies (VCC_APPS and VCC_SRAM) are controlled through PWR_EN or I^2C commands, and if these two supplies are not disabled before SYS_EN is de-asserted, they must be disabled by the de-assertion of SYS_EN. All low-voltage power supplies must be removed before any high-voltage power supply is removed.

The power supply ramp times and sequences, as well as the delay between the power supply transitions and nRESET_OUT de-assertion, are described in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Permanent damage to the device may result if any other supply power-up sequence is followed. The sequence for power-on reset is the same as for a hardware reset.

6.12.2 Hardware Reset

Hardware reset is invoked when the nRESET pin is asserted. Hardware reset is for a complete and total reset only. All internal registers are held in their defined reset states. The remaining internal clocks are stopped and the processor is fully static. Additionally, all pins assume their reset states, including SYS_EN, which is de-asserted.

There is no way to mask or disable the propagation of the nRESET pin value into the MPMU. When nRESET is asserted, the processor enters reset state regardless of the previous mode. The nRESET_OUT pin is asserted while the nRESET pin is asserted. To enter hardware reset, nRESET must be held low long enough to allow internal stabilization and propagation of the reset state. For timing specifications, refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.

The sequence for hardware reset is as follows:

1. The nRESET pin is asserted for a period of time described in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.
2. The processor asserts nRESET_OUT and de-asserts SYS_EN.
3. The nRESET pin is de-asserted.

Warning

VCC_BBATT must be stable before the nRESET is de-asserted or operation is not guaranteed.

4. The processor asserts SYS_EN, enabling high-voltage power supplies.
5. nRESET_OUT is asserted.
6. The processor asserts PWR_EN, and the PWR_I^2C module sends I^2C commands to enable the external low-voltage power supplies, VCC_APPS and VCC_SRAM

Note

The delay between the de-assertion of nRESET and nRESET_OUT is described in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.

Normal boot-up sequencing begins with all subsystems, starting with their predefined reset conditions. The core software must examine the Application Subsystem Reset Status Register (ARSR) to determine that the reset source is a hardware reset.

6.12.3 GPIO Reset

GPIO reset is a soft reset in which data can be maintained in external memory devices and the real-time clock. GPIO reset is invoked when the nGPIO_RESET pin is asserted or when the SWGR bit in the Power Management Unit Control Register (PMCR) is set.

During GPIO reset, all units in the processor subsystem except those in the services unit and those listed in Table 62 are held at their defined reset conditions. GPIO reset does not reset the services unit state machines, the system or DMC PLLs, or registers in the real-time clock (RTC). The APMU handshakes with the dynamic memory controller to ensure that external memory is put into self-refresh mode before GPIO reset propagates to the application subsystem. The nRESET_OUT signal is asserted during GPIO reset if the GP_ROD bit in Power Management Unit General Configuration Register (PCFR) is cleared.

If nGPIO_RESET is asserted for less than the amount of time specified in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*, the detection of nGPIO_RESET assertion is indeterminate and the processor may remain in the previous mode or enter GPIO reset.

The sequence for GPIO reset is as follows when the processor is in S0/D0 mode:

1. The APMU handshakes with the dynamic memory controller to stop DDR SDRAM transfers and put external DDR SDRAM into self-refresh mode.
2. The SPMU asserts nRESET_OUT if the PCFR[GP_ROD] bit is cleared.
3. The SPMU asserts system reset to the CPMU and GPIO reset to the pad unit.
4. If the processor oscillator and PLLs are enabled when nGPIO_RESET is asserted, they are not disabled. In addition, if nBATT_FAULT is asserted when nGPIO_RESET is asserted, the SPMU enters S3 state and does not exit until an enabled wakeup event is detected.
5. nGPIO_RESET is de-asserted.
6. The SPMU de-asserts nRESET_OUT.

The sequence for GPIO reset is as follows when the processor is in S2/D3/C4 state:

1. nGPIO_RESET is asserted.
2. The SPMU detects the assertion of nGPIO_RESET and treats it as a wakeup event.
3. The APMU does not handshake with the dynamic memory controller to stop DDR SDRAM transfers and put external memory into self-refresh mode because the dynamic memory controller is already in self-refresh mode because the SPMU is in S2/D3/C4 mode.
4. If the PCFR[GP_ROD] bit is set, the SPMU asserts nRESET_OUT.
5. If the processor oscillator is enabled when nGPIO_RESET is asserted, it is not disabled.
6. nGPIO_RESET is de-asserted.

Copyright © 2009 Marvell

Note

The delay between the de-assertion of nGPIO_RESET and nRESET_OUT is described in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*.

When normal boot-up sequencing begins, all units in the processor subsystems start with their predefined reset conditions. The application core software must examine the Application Subsystem Reset Status Register (ARSR) to determine that the reset source is a GPIO reset.

6.12.4 Global Watchdog Reset

There is one watchdog reset source in the timers module. The watchdog reset is invoked when software fails to prevent a watchdog time-out event for a timer. The internal watchdog reset is output to the entire processor and causes a reset as if a hardware reset is asserted except for the modules listed in Table 62. It resets the services unit, application and communication subsystems, and the pad unit. Additionally, all pins assume their reset states, including SYS_EN, which is de-asserted. The communication subsystem watchdog reset resets the communication subsystem only and is output to the services unit and application subsystem as an interrupt. The communication subsystem watchdog reset operation is not discussed in this manual because it does not reset the services unit or the application subsystem.

Global watchdog reset is invoked when values in the following two registers match that of the OS timer counter:

- WME bit in the OS Timer Watchdog Match Enable register (OWER) in the timer module
- OS Timer Match register 3 (OSMR3)

The OS counter for Channel 3 increments only in S0/D0/C0, S0/D0/C1, and S0/D0CS/C0 modes. A watchdog reset cannot occur in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4. When the match occurs, the processor enters global watchdog reset regardless of the previous mode.

The sequence for global watchdog reset is as follows:

1. The processor OS timer asserts global watchdog reset (OSMR3 matches the OS timer counter.)
2. The processor OS timer global watchdog reset is automatically reset as a result of propagation of the internal application subsystem reset state.
3. The communication subsystem, the services unit, and the pad unit are reset.
4. The reset is de-asserted.

The services unit executes the hardware reset sequence, and normal boot-up sequencing begins for the application subsystem. All units start with their predefined reset state. The application core software must examine the Application Subsystem Reset Status Register (ARSR) to determine that the reset source is a global watchdog reset.

6.12.5 S3/D4/C4 Low-Power Mode Exit Reset

Low-power mode exit reset is invoked when the APMU exits the S3/D4/C4 power state. All units in the application subsystem that were powered off are reset to their predefined reset states. The SPMU controls S3/D4/C4 mode exit reset.

6.13 Registers

Table 63 lists the coprocessor registers.

Table 63: Core Coprocessor Register Summary

Address	Description	Page
Coprocessor 14, Register 6	Core Clock Configuration Register (XCLKCFG) (CP14 Register 6)	page 213
Coprocessor 14, Register 7	Core PWRMODE Register (CP14 Register 7)	page 214

Table 64 lists the registers of the Services Clock Control Unit and the power management unit and the physical addresses to access them.

Table 64: Services Unit Register Summary

Address	Description	Page
Services Power Management Unit		
0x40F5_0000	Power Management Unit Control Register (PMCR)	page 215
0x40F5_0004	Power Management Unit Status Register (PSR)	page 216
0x40F5_0008	Power Management Unit Scratch-Pad Register (PSPR)	page 217
0x40F5_000C	Power Management Unit General Configuration Register (PCFR)	page 218
0x40F5_0010	Power Manager Wake-Up Enable Register (PWER)	page 220
0x40F5_0014	Power Manager Wake-Up Status Register (PWSR)	page 221
0x40F5_0018	Power Manager EXT_WAKEUP<1:0> Control Register (PECR)	page 223
0x40F5_0088	Power Manager Mask Event Register (PMER)	page 225
0x40F5_0094	VCC_APPS Voltage Control Register (AVCR)	page 228
0x40F5_0098	VCC_SRAM Voltage Control Register (SVCR)	page 229
0x40F5_0100	Power Management Unit Voltage Change Control Register (PVCR)	page 231
Services Clock Control Unit		
0x4135_0000	Oscillator Configuration Register (OSCC)	page 232

Table 65 shows the registers of the application subsystem clock control unit and power management unit and the physical addresses to access them.

Table 65: Application Subsystem Unit Register Summary

Address	Description	Page
Application Subsystem Power Management Unit		
0x40F4_0000	Application Subsystem Power Status/Configuration Register (ASCR)	page 233

Copyright © 2009 Marvell

Table 65: Application Subsystem Unit Register Summary

Address	Description	Page
0x40F4_0004	Application Subsystem Reset Status Register (ARSR)	page 234
0x40F4_0008	Application Subsystem Wakeup from D3 Enable Register (AD3ER)	page 235
0x40F4_000C	Application Subsystem Wakeup from D3 Status Register (AD3SR)	page 239
0x40F4_0010	Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER)	page 243
0x40F4_0014	Application Subsystem Wakeup from D2 to D0 Status Register (AD2D0SR)	page 246
0x40F4_0018	Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER)	page 249
0x40F4_001C	Application Subsystem Wakeup from D2 to D1 Status Register (AD2D1SR)	page 249
0x40F4_0020	Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER)	page 250
0x40F4_0024	Application Subsystem Wakeup from D1 to D0 Status Register (AD1D0SR)	page 253
0x40F4_002C	Application Subsystem General Purpose Register (AGENP)	page 259
0x40F4_0030	Application Subsystem D3 Configuration Register (AD3R)	page 259
0x40F4_0034	Application Subsystem D2 Configuration Register (AD2R)	page 258
0x40F4_0038	Application Subsystem D1 Configuration Register (AD1R)	page 259
Application Subsytem Clock Control Unit		
0x4134_0000	Application Subsystem Clock Configuration Register (ACCR)	page 260
0x4134_0004	Application Subsystem Clock Status Register (ACSR)	page 266
0x4134_0008	Application Subsystem Interrupt Control/Status Register (AICSR)	page 269
0x4134_000C	D0 Mode Clock Enable Register A (D0CKEN_A)	page 270
0x4134_0010	D0 Mode Clock Enable Register B (D0CKEN_B)	page 275
0x4134_0014	AC '97 Clock Divisor Value Register (AC97_DIV)	page 277

6.13.1 Core Clock Configuration Register (XCLKCFG) (CP14 Register 6)

Coprocessor 14 contains one register that controls the clock modes on the processor.

- CP14 register 6—XCLKCFG register

The instruction requires the following field assignments:

Opcode1:0

Opcode2:0

CRn:6

CRm:0

Consult the *ARM Reference Manual* for details.

Note

CP14 registers can be accessed in supervisor mode only.

The Core Clock Configuration Register (XCLKCFG) (CP14 Register 6), defined in Table 66, controls the following modes and sequences:

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 66: XCLKCFG Bit Definitions

	Coprocessor 14 Register CR6								XCLKCFG												Core											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																														F	T
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:2	—	—	Reserved
1	R	F	Core Frequency Change 0 = No core frequency change is performed. 1 = Core frequency change sequence begin when written.
0	R	T	Turbo Mode 0 = No core frequency change is performed. 1 = Core frequency change sequence begin when written.

6.13.2 Core PWRMODE Register (CP14 Register 7)

The Core PWRMODE Register (CP14 Register 7), defined in Table 67, initiates the following power modes:

- S0/D0/C0 power mode (run and turbo)
- S0/D0/C1 (core Idle)
- S0/D1/C2 power mode (Standby with LCD refresh (mini-LCD)
- S0/D2/C2 power mode (Standby)
- S0/D0CS/C0 power mode (Ring Oscillator - core running)
- S0/D0CS/C1 power mode (Ring Oscillator - core idle)
- S2/D3/C4 power mode (Sleep)
- S3/D4/C4 power mode (Deep Sleep)

The reset mode is S0/D0/C0, in which the core is fully active. The instruction requires the following field assignments:

- Opcode1:0
- Opcode2:0

Copyright © 2009 Marvell

- CRn:7
- CRm:0

For details, consult the *ARM Processor Reference Manual*.

This is a write-only register. Reads from this register always return 0b0. Write 0b0 to reserved bits.

Table 67: PWRMODE Bit Definitions

	Coprocessor 14 Register CR7												PWRMODE												Core							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																													M		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0

Bits	Access	Name	Description
31:3	—	—	reserved
2:0	W	M	**Mode** 0b000 = S0/D0/C0 power mode 0b001 = S0/D0/C1 (core Idle) 0b010 = S0/D1/C2 power mode (PXA30x and PXA31x Only) 0b011 = S0/D2/C2 power mode (PXA30x and PXA31x Only) 0b100 = Reserved 0b101 = S0/D0CS/C0 power mode (Ring Oscillator) 0b110 = S2/D3/C4 power mode 0b111 = S3/D4/C4 power mode These bits are cleared (return to S0/D0/C0) upon completion of the specified mode

6.13.3 Power Management Unit Control Register (PMCR)

Power Management Unit Control Register (PMCR), defined in Table 68, controls the SPMU interrupt generation and behavior when SPMU-based interrupt conditions occur. Power-on, hardware, and watchdog resets return the PMCR bits to their reset values, as shown in Table 68. GPIO reset does not affect the PMCR bits.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 68: PMCR Bit Definitions

Physical Address 0x40F5_0000	PMCR																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SWGR	Reserved																														
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31	R/Write 1 to Set	SWGR	Software GPIO Reset 0 = GPIO reset not asserted to subsystems 1 = Assert GPIO reset to subsystems
30:0	—	—	Reserved

6.13.4 Power Management Unit Status Register (PSR)

Power Management Unit Status Register (PSR), defined in Table 69, contains the following status flags:

The status flags are cleared by writing 0b1 to them. Writing 0b0 has no effect. Power-on and hardware resets return the PSR bits to their reset values, as shown in Table 69.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 69: PSR Bit Definitions

Physical Address 0x40F5_0004	PSR																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																	TSS			Reserved									BFS	SS3S	SS2S
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]

Bits	Access	Name	Description
31:3	—	—	Reserved
2	R/Write 1 to Clear	BFS	Battery Fault Status 0 = nBATT_FAULT not asserted since it was last cleared by a reset or by software. 1 = nBATT_FAULT asserted and caused the SPMU to enter S3 mode[††].

Copyright © 2009 Marvell

Table 69: PSR Bit Definitions (Continued)

	Physical Address 0x40F5_0004																PSR				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																	TSS			Reserved									BFS	SS3S	SS2S
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]

Bits	Access	Name	Description
1	R/Write 1 to Clear	SS3S	Software S3 Status 0 = SPMU is not placed into S3 state by writing the application core PWRMODE register (CP14 Register 7) since SS3S was last cleared by a reset or by the software. 1 = SPMU placed into S3 state by writing the application core PWRMODE Register (CP14 Register 7).
0	R/Write 1 to Clear	SS2S	Software S2 Status 0 = SPMU not placed into S2 state by writing the application core PWRMODE register (CP14 Register 7) since SS2S was last cleared by a reset or by the software. 1 = The SPMU placed into S2 state by writing the application core PWRMODE register (CP14 Register 7).

6.13.5 Power Management Unit Scratch-Pad Register (PSPR)

Power Management Unit Scratch-Pad Register (PSPR), defined in Table 70, saves processor configuration information in any preferred format. PSPR is a holding register that is powered during S2 and S3 states (if no nBATT_FAULT is asserted) and is cleared by power-on, hardware, and watchdog resets. Any value can be written to it in S0 state. The value can be read in S0 state. Use the register value to retain processor configuration prior to entering low-power states.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 70: PSPR Bit Definitions

	Physical Address 0x40F5_0008																PSPR				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SP31	SP30	SP29	SP28	SP27	SP26	SP25	SP24	SP23	SP22	SP21	SP20	SP19	SP18	SP17	SP16	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Reset	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]

Bits	Access	Name	Description
31:0	R/W	SP[n]†	Scratch Pad Register bit n, where n = [31:0]

1. S3 low-power mode exit does not clear this bit.

6.13.6 Power Management Unit General Configuration Register (PCFR)

Power Management Unit General Configuration Register (PCFR), defined in Table 71, contains bits to configure various functions within the processor.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 71: PCFR Bit Definitions

	Physical Address 0x40F5_000C																															
	PCFR																				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SYS_DEL				PWR_DEL				LPM_DEL				Reserved						L1_DIS	L0_EN	Reserved									PUDH	SL_ROD	GP_ROD
Reset	1[1]	1[1]	0[1]	0[1]	1[1]	1[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	?	?	?	?	?	?	0[1]	0[1]	?	?	?	?	?	?	?	?	?	0[2]	0[1]	0[1]

Bits	Access	Name	Description
31:28	R/W	SYS_DEL	System-Power-Supply Ramp Delay Specifies the number of timekeeping oscillator (~32 kHz) cycles between assertion of SYS_EN (system power supplies enabled) and continuation of the power-up sequence to 2^n (reset value = 0b1100 for 125 ms delay). 0b0000-0b1100: n = SYS_DEL 0b1101-0b1111: n = 12 Example: For SYS_DEL = 0b1100: 0b1100 = 12 = n $2^n = 2^{12} = 4096$ System Power Supply Ramp Delay = 4096 × (1/32.768 kHz) = ~125 ms This is followed in ~5 32-KHz clocks by the assertion of PWR_EN and the output of the I^2C commands to enable low-voltage power supplies.
27:24	R/W	PWR_DEL	Power-Supply (VCC_APPS and VCC_SRAM) Ramp Delay Specifies the number of timekeeping-oscillator (~32 kHz) cycles between the assertion of the PWR_EN signal or the completion of I^2C commands used to enable the low-voltage power supplies and the continuation of the power up sequence to 2^n (reset value = 0b1100 for 125 ms delay). 0b0000-0b1100: n = PWR_DEL 0b1101-0b1111: n = 12 This is followed in ~5 32-Hz clocks by the continuation of the S3 and S2-exit sequence. See the PCFR[SYS_DEL] bit description for a similar example calculation

Copyright © 2009 Marvell

Table 71: PCFR Bit Definitions (Continued)

	Physical Address 0x40F5_000C																															
	PCFR																				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SYS_DEL				PWR_DEL				LPM_DEL				Reserved						L1_DIS	L0_EN	Reserved									PUDH	SL_ROD	GP_ROD
Reset	1[1]	1[1]	0[1]	0[1]	1[1]	1[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	?	?	?	?	?	?	0[1]	0[1]	?	?	?	?	?	?	?	?	?	0[2]	0[1]	0[1]

Bits	Access	Name	Description
23:20	R/W	LPM_DEL	Power-Supply (VCC_APPS and VCC_SRAM) Ramp Delay when exiting D1, D2, and D3 Low-Power Modes (SPMU is in S0 state) and executing an application subsystem frequency change Specifies the number of timekeeping-oscillator (~32 kHz) cycles between completion of I^2C commands used to enable the D1, D2, and D3 disabled power supplies and the continuation of the power up or frequency change sequence. Example: For LPM_DEL = 0b1011: 0b1011 = 11 Power Supply Ramp Delay = 11 x (1/32.768 kHz) = ~ 0.336 ms
19:14	—	—	Reserved
13	R/W	L1_DIS	S2/D3/C4 Mode L1 Regulator Disable 0 = The L1 high current regulator is used in S2/D3/C4 mode. 1 = The L1 high current regulator is turned off in S2/D3/C4 mode. L0_EN determines the alternate power source used for S2/D3/C4 mode. **NOTE:** Disables the use of the high-current linear regulator (L1) in S2/D3/C4 low-power mode. If set, the L0_EN bit determines the replacement for the L1 converter.
12	R/W	L0_EN	S2/D3/C4 and S3/D4/C4 Mode L0 Regulator Enable 0 = The DC-DC converter is used in S3/D4/C4 mode. 1 = L0 regulator is used in S3/D4/C4 mode. **NOTE:** Selects between low-power regulator L0 and DC-DC regulator during S2/D3/C4 and S3/D4/C4 mode. For this bit to be effective in S2/D3/C4 low-power mode, the L1_DIS bit must be set.
11:3	—	—	Reserved
2	R/W	PUDH	Pull-up/Pull-down Disable Enables/disables the pull-up and pull-down resistors in the pads. 1 = Pull-up and pull downs in nGPIO_RESET, EXT_WAKEUP<:1:0>, PWR_SCL, and PWR_SDA are disabled 0 = Pull-up and pull downs in nGPIO_RESET, EXT_WAKEUP<1:0>, PWR_SCL and PWR_SDA are enabled. PUDH may be set by software after the external devices driving these pins are configured.

Table 71: PCFR Bit Definitions (Continued)

	Physical Address 0x40F5_000C																															
	PCFR																		SPMU													
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SYS_DEL				PWR_DEL				LPM_DEL				Reserved						L1_DIS	L0_EN	Reserved									PUDH	SL_ROD	GP_ROD
Reset	1[1]	1[1]	0[1]	0[1]	1[1]	1[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	?	?	?	?	?	?	0[1]	0[1]	?	?	?	?	?	?	?	?	?	0[2]	0[1]	0[1]

Bits	Access	Name	Description
1	R/W	SL_ROD	S2 State nRESET_OUT Disable Prevents the nRESET_OUT pin from asserting upon entry into S2 or S3 states. 0 = nRESET_OUT is asserted upon entry into S2 or S3 states. 1 = nRESET_OUT is not asserted upon entry into S2 or S3 states, but the application subsystems is reset.
0	R/W	GP_ROD	GPIO nRESET_OUT Disable Enables/disables assertion of nRESET_OUT during GPIO reset. 0 = nRESET_OUT is asserted during GPIO reset. 1 = nRESET_OUT is not asserted during GPIO reset. **Note:** GPIO reset does not clear this bit.

1. S3 low-power mode exit reset does not clear or set this bit.
2. S3 low-power mode exit reset does not clear or set this bit and GPIO reset does reset this bit.

6.13.7 Power Manager Wake-Up Enable Register (PWER)

Power Manager Wake-Up Enable Register (PWER), defined in Table 72, selects whether the falling edge or rising edge of the corresponding wakeup source(s) causes a wakeup from S2 or S3 state. Possible wakeup sources that can be programmed in PWER are RTC alarm and EXT_WAKEUP<1:0>. For an external wakeup to serve as a wakeup source from these modes:

- Either or both of the rising-edge or falling-edge bits in PWER must be set.
- The following timing requirements must met (refer to the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)* for information on time durations):
  - If a rising edge is to be detected on a signal, the signal must be held low for a minimum time, then high for a minimum time.
  - If a falling edge is to be detected on a signal, the signal must be held high for a minimum time, then low for a minimum time.
  - When nBATT_FAULT is asserted, PWER assumes its reset value, enabling only EXT_WAKEUP<1:0> as wakeup sources.

A write to the PWER is followed by reads to ensure the register has been updated with the new register values before proceeding.

Copyright © 2009 Marvell

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 72: PWER Bit Definitions

	Physical Address 0x40F5_0010																															
	PWER																SPMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WERTC	Reserved																											WEF1	WEF0	WER1	WER0
PXA31x and PXA30x Only	WERTC	Reserved																												WEF0	Reserved	WER0
Reset	0[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1[1]	?	1[1]

Bits	Access	Name	Description
31	R/W	WERTC	Wakeup Enable for RTC S2 or S3 State 0 = Disable wakeup due to RTC alarm. 1 = Enable wakeup due to RTC alarm.
30:4	—	—	Reserved
3	R/W	WEF[1]†	Wakeup Enable for Falling-Edge EXT_WAKEUP<1> for S2 or S3 State 0 = Disable falling-edge wakeup due to EXT_WAKEUP<1> 1 = Enable falling-edge wakeup due to EXT_WAKEUP<1> **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
2	R/W	WEF[0]†	Wakeup Enable for Falling-Edge EXT_WAKEUP<0> for S2 or S3 State 0 = Disable falling-edge wakeup due to EXT_WAKEUP<0>. 1 = Enable falling-edge wakeup due to EXT_WAKEUP<0>.
1	R/W	WER[1]†	Wakeup Enable for Rising-Edge EXT_WAKEUP<1> for S2 or S3 State 0 = Disable rising-edge wakeup due to EXT_WAKEUP<1> 1 = Enable rising-edge wakeup due to EXT_WAKEUP<1> **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
0	R/W	WER[0]†	Wakeup Enable for Rising-Edge EXT_WAKEUP<0> for S2 or S3 State 0 = Disable rising-edge wakeup due to EXT_WAKEUP<0>. 1 = Enable rising-edge wakeup due to EXT_WAKEUP<0>.

1. S3 low-power mode exit reset does not clear or set this bit.

6.13.8 Power Manager Wake-Up Status Register (PWSR)

Power Manager Wake-Up Status Register (PWSR), defined in Table 73, indicates which sources (enabled through the PWER) caused a wake up from S2 or S3 states. These bits can be set only by a rising edge, falling edge, or either on the given EXT_WAKEUP<1:0> pin, depending on the settings in the PWER. These bits are cleared by writing a 0b1 to them. Writing a 0b0 to any status bit has no effect.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 73: PWSR Bit Definitions

	Physical Address 0x40F5_0014																															
	PWSR																SPMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	EERTC	Reserved																											EDF1	EDF0	EDR1	EDR0
PXA31x and PXA30x Only	EERTC	Reserved																												EDF0	Reserved	EDR0
Reset	0[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	?	0[1]

Bits	Access	Name	Description
31	R/Write 1 to Clear	EERTC	S2 or S3 Wakeup from RTC 0 = Wakeup due to RTC source not detected. 1 = Wakeup due to RTC source is detected.
30:4	—	—	Reserved
3	R/Write 1 to Clear	EDF[1]†	S2 or S3 Wakeup from EXT_WAKEUP<1> 0 = Wakeup due to EXT_WAKEUP<1> falling-edge detect not detected. 1 = Wakeup due to EXT_WAKEUP<1> falling-edge detect is detected. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
2	R/Write 1 to Clear	EDF[0]†	S2 or S3 Wakeup from EXT_WAKEUP<0> 0 = Wakeup due to EXT_WAKEUP<0> falling-edge detect not detected. 1 = Wakeup due to EXT_WAKEUP<0> falling-edge detect is detected.
1	R/Write 1 to Clear	EDR[1]†	S2 or S3 Wakeup from EXT_WAKEUP<1> 0 = Wakeup due to EXT_WAKEUP<1> rising-edge detect not detected. 1 = Wakeup due to EXT_WAKEUP<1> rising-edge detect is detected. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
0	R/Write 1 to Clear	EDR[0]†	S2 or S3 Wakeup from EXT_WAKEUP<0> 0 = Wakeup due to EXT_WAKEUP<0> rising-edge detect not detected. 1 = Wakeup due to EXT_WAKEUP<0> rising-edge detect is detected.

1. S3 low-power mode exit reset does not clear or set this bit.

6.13.9 Power Manager EXT_WAKEUP<1:0> Control Register (PECR)

Power Manager EXT_WAKEUP<1:0> Control Register (PECR), defined in Table 74 indicates the current state of the EXT_WAKEUP<1:0> pin and controls the interrupt generation for EXT_WAKEUP<1:0> based interrupts. The interrupt status bits are cleared by writing a 0b1 to them.

Copyright © 2009 Marvell

Writing a 0b0 to any of the interrupt status bits has no effect. PECR status bits are maintained throughout operation and are independent of power states.

Software must carefully separate two back-to-back writes to this register (for example, due to back-to-back Interrupt Service Routine (ISR) Entry or a combination of a regular Write occurring too close to an ISR Write).

The IN0 and IN1 status bits are updated only when a change is detected on the EXT_WAKEUP<1:0> input pins. The IN0 and IN1 status bits are cleared after reset and are not updated until an edge (signal change) is detected on the corresponding EXT_WAKEUP<1:0> input pin.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 74: PECR Bit Definitions (Continued)

Physical Address 0x40F5_0018 — **PECR** — **SPMU**

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	E1IS	E1IE	E0IS	E0IE	Reserved																DS1			DS0			DIR1	DIR0	OUT1	OUT0	IN1	IN0
PXA31x and PXA30x Only	Reserved		E0IS	E0IE	Reserved																			DS0			Reserved	DIR0	Reserved	OUT0	Reserved	IN0
Reset	?	?	0[1]	0[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	?	0[1]	?	0[1]	?	0[1]

Bits	Access	Name	Description
31	R/Write 1 to Clear	E1IS	EXT_WAKEUP<1> Interrupt Status 0 = Interrupt not due to EXT_WAKEUP<1> pin. 1 = Interrupt was due to EXT_WAKEUP<1> pin change. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
30	R/W	E1IE	EXT_WAKEUP<1> Pin Interrupt Enable 0 = No interrupt when EXT_WAKEUP<1> pin changes. 1 = Generate an interrupt when EXT_WAKEUP<1> pin changes. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
29	R/Write 1 to Clear	E0IS	EXT_WAKEUP<0> Interrupt Status 0 = Interrupt not due to EXT_WAKEUP<0> pin. 1 = Interrupt was due to EXT_WAKEUP<0> pin change.
28	R/W	E0IE	EXT_WAKEUP<0> Pin Interrupt Enable 0 = No interrupt when EXT_WAKUEP<0> pin changes. 1 = Generate an interrupt when EXT_WAKEUP<0> pin changes.
27:12	—	—	Reserved

Table 74: PECR Bit Definitions (Continued)

Physical Address 0x40F5_0018	PECR																				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	E1IS	E1IE	E0IS	E0IE	Reserved																DS1			DS0			DIR1	DIR0	OUT1	OUT0	IN1	IN0
PXA31x and PXA30x Only	Reserved		E0IS	E0IE	Reserved																			DS0			Reserved	DIR0	Reserved	OUT0	Reserved	IN0
Reset	?	?	0[1]	0[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	?	0[1]	?	0[1]	?	0[1]

Bits	Access	Name	Description
11:9	R/W	DS1	Drive Strength for EXT_WAKEUP<1>: Sets drive strength for EXT_WAKEUP<1> output driver from 0b000 (weakest drive) to 0b111 (strongest drive). **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
8:6	R/W	DS0	Drive Strength for EXT_WAKEUP<0>: Sets drive strength for EXT_WAKEUP<0> output driver from 0b000 (weakest drive) to 0b111 (strongest drive).
5	R/W	DIR1	Direction for EXT_WAKEUP<1>: 0 = Configure EXT_WAKEUP<1> as an input. 1 = configure EXT_WAKEUP<1> as an output. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
4	R/W	DIR0	Direction for EXT_WAKEUP<0>: 0 = Configure EXT_WAKEUP<0> as an input. 1 = Configure EXT_WAKEUP<0> as an output.
3	R/W	OUT1	Output Value for EXT_WAKEUP<1> 0 = Drive EXT_WAKEUP<1> low when configured as an output. 1 = Drive EXT_WAKEUP<1> high when configured as an output. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
2	R/W	OUT0	Output Value for EXT_WAKEUP<0> 0 = Drive EXT_WAKEUP<0> low when configured as an output. 1 = Drive EXT_WAKEUP<0> high when configured as an output.

Copyright © 2009 Marvell

Table 74: PECR Bit Definitions (Continued)

	Physical Address 0x40F5_0018																															
	PECR																				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	E1IS	E1IE	E0IS	E0IE	Reserved																DS1			DS0			DIR1	DIR0	OUT1	OUT0	IN1	IN0
PXA31x and PXA30x Only	Reserved		E0IS	E0IE	Reserved																			DS0			Reserved	DIR0	Reserved	OUT0	Reserved	IN0
Reset	?	?	0[1]	0[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	?	0[1]	?	0[1]	?	0[1]

Bits	Access	Name	Description
1	R	IN1	Input Value for EXT_WAKEUP<1> 0 = A logic 0 value read on EXT_WAKEUP<1> when configured as an input. 1 = A logic 1 value read on EXT_WAKEUP<1> when configured as an input. **NOTE:** PXA32x Only
	—	—	Reserved (PXA31x and PXA30x Only)
0	R	IN0	Input Value for EXT_WAKEUP<0> 0 = A logic 0 value read on EXT_WAKEUP<0> when configured as an input. 1 = A logic 1 value read on EXT_WAKEUP<0> when configured as an input.

1. S3 low-power mode exit reset does not clear or set this bit.

6.13.10 Power Manager Mask Event Register (PMER)

PMER, defined in Table 75, is used to provide software control of the power manager event detection and execution. PMER provides control of the following events:

- GPIO reset — Masks the execution of the GPIO reset sequence in the APMU.
  - When PMER[GRME] = 0b0, the GPIO reset event is not masked and when the nGPIO_RESET signal is asserted, the APMU places the external memory into self-refresh and then GPIO reset is propagated to the application subsystem. All units in the application subsystem are reset to their predefined states.
  - When PMER[GRME] = 0b1, the GPIO reset event is masked and when the nGPIO_RESET signal is asserted, the SPMU does not propagate GPIO reset to the APMU. The external memory controller is not placed into self-refresh mode and the application subsystem is not reset.
- Wake-up Enable— Enables the detection of EXT_WAKEUP<1:0> and RTC wake up prior to completing the low-power mode entry sequence.
  - When the external wake-up event is enabled, the SPMU detects wake-up events for S2 and S3 entry even though the low-power mode entry sequence has not completed. If PMER[EWEE] = 0b1, the SPMU detects a transition on EXT_WAKEUP<1:0> and then when

S2 and S3 entry sequence is entered, exits the low-power mode if the detected wake-up event is enabled.

- When PMER[EWEE] = 0b0, the SPMU does not detect transitions on EXT_WAKEUP<1:0> until a low-power mode entry sequence has completed. For S2 entry, the SPMU does not detect wake-up events until PWR_EN is asserted. For S3 entry, the SPMU does not detect wake-up events until SYS_EN is asserted.

When the SPMU is in S0 and the APMU is in D0, and when the SPMU is transitioning from S0 to S2 or S3, or the APMU is transitioning from D0 to D1, D2 or D3, the bits in the PMER are used to mask the respective SPMU events. Once the SPMU has entered S2 or S3 or the APMU has entered D1, D2 or D3, the PMER bits are no longer used to mask any SPMU events. The PMER has no effect on event detection when either the SPMU or APMU are in a low-power mode.

Note

If the event being masked is pending when the software writes PMER, the Write is ignored. For example, if writing a 1 to the GRME, but a GPIO reset has been detected and is pending, users can not write to the PMER. Therefore, when writing to the PMER, the software must write the PMER and then read the register contents to ensure the Write was accepted. Software should not proceed with power mode or frequency changes until the PMER write has completed and the register contents verified.

Warning

PMER is a reserved system register and must not be written to or read from outside the scope of the defined PWR_I2C software accesses.

Table 75: PMER Bit Definitions (Continued)

Physical Address 0x40F5_0088 — **PMER** — **SPMU**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved																												EWEE	Reserved		GRME
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	EWEE	External Wake-up Event Enable: 0 = Do not enable the detection of Wake-up events for S2 and S3 low power modes until after the mode entry sequence has completed 1 = Enable the detection of Wake-up Events during S2 and S3 low power mode entry sequences.
2:1	—	—	reserved

Copyright © 2009 Marvell

Table 75: PMER Bit Definitions (Continued)

Physical Address 0x40F5_0088																																
PMER																																
SPMU																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												EWEE	Reserved		GRME
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0

Bits	Access	Name	Description
0	R/W	GRME	GPIO Reset Mask Event: 0 = Allow the GPIO reset sequence execution and reset assertion when nGPIO_RESET is asserted. 1 = Do not execute the GPIO reset sequence and do not assert an internal reset when nGPIO_RESET is asserted.

NOTE: All bits must be programmed with the same value (0x0F or 0x00). Any other values are not supported and will cause failures.

6.13.11 VCC_APPS Voltage Control Register (AVCR)

AVCR controls the voltage sent to the PMIC for VCC_APPS as required for the frequencies defined through the Services Clock Control Unit (SCCU).

Warning

Do not exceed the voltage levels outlined in Section 5.2 Operating Conditions in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. For details about the required the voltage levels for each frequency range, refer to the Voltage, Temperature, and Frequency Electrical Specification Table.

Note

The default values for this register do not match the typical values as outlined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Power management software must program these registers to match the typical values in the Voltage, Temperature, and Frequency Electrical Specification Table in the EMTS for each operating point.

Table 76: AVCR Bit Definitions

	Physical Address 0x40F5_0094																															
	AVCR																				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			VCCAPPS_4					reserved			VCCAPPS_3					reserved			VCCAPPS_2					reserved			VCCAPPS_1				
Reset	?	?	?	1	1	0	1	1	?	?	?	1	1	0	1	0	?	?	?	0	1	1	1	1	?	?	?	0	1	0	1	1

Bits	Access	Name	Description
31:29	—	—	reserved
28:24	R/W	**VCCAPPS_4**	VCC_APPS voltage output for Level 3 (see **VCCAPPS_1** for encoding)
23:21	—	—	reserved
20:16	R/W	**VCCAPPS_3**	VCC_APPS voltage output for Level 2 (see **VCCAPPS_1** for encoding)
15:13	—	—	reserved
12:8	R/W	**VCCAPPS_2**	VCC_APPS voltage output for Level 1 (see **VCCAPPS_1** for encoding)
7:5	—	—	reserved
4:0	R/W	VCCAPPS_1	VCC_APPS voltage output for Level 0: 0x00 - 0.725V 0x01 - 0.750V 0x02 - 0.775V 0x03 - 0.800V 0x04 - 0.825V 0x05 - 0.850V 0x06 - 0.875V 0x07 - 0.900V 0x08 - 0.925V 0x09 - 0.950V 0x0A - 0.975V 0x0B - 1.000V 0x0C - 1.025V 0x0D - 1.050V 0x0E - 1.075V 0x0F - 1.100V 0x10 - 1.125V 0x11 - 1.150V 0x12 - 1.175V 0x13 - 1.200V 0x14 - 1.225V 0x15 - 1.250V 0x16 - 1.275V 0x17 - 1.300V 0x18 - 1.325V 0x19 - 1.350V 0x1A - 1.375V 0x1B - 1.400V 0x1C - 1.425V 0x1D - 1.450V 0x1E - 1.475V 0x1F - 1.500V

6.13.12 VCC_SRAM Voltage Control Register (SVCR)

AVCR controls the voltage sent to the PMIC for VCC_SRAM as required for the frequencies defined through the Services Clock Control Unit (SCCU).

Copyright © 2009 Marvell

Warning

Do not exceed the voltage levels outlined in Section 5.2 Operating Conditions in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. For details about the required voltage levels for each frequency range, refer to the Voltage, Temperature, and Frequency Electrical Specification Table.

Note

The default values for this register do not match the typical values as outlined in the *PXA3xx Processor Family Processor Electrical, Mechanical, and Thermal Specification (EMTS)*. Power management software must program these registers to match the typical values in the Voltage, Temperature, and Frequency Electrical Specification Table in the EMTS for each operating point.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 77: SVCR Bit Definitions

	Physical Address 0x40F5_0098																															
	SVCR																SPMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			VCCSRAM_4					reserved			VCCSRAM_3					reserved			VCCSRAM_2					reserved			VCCSRAM_1				
Reset	?	?	?	1	1	0	1	1	?	?	?	1	1	0	1	1	?	?	?	1	0	0	1	1	?	?	?	0	1	1	1	1

Bits	Access	Name	Description
31:29	—	—	reserved
28:24	R/W	VCCSRAM_4	VCC_SRAM voltage output for Level 3 (see VCCSRAM_1 for encoding)
23:21	—	—	reserved
20:16	R/W	VCCSRAM_3	VCC_SRAM voltage output for Level 2 (see VCCSRAM_1 for encoding)
15:13	—	—	reserved
12:8	R/W	VCCSRAM_2	VCC_SRAM voltage output for Level 1 (see VCCSRAM_1 for encoding)
7:5	—	—	reserved

Copyright © 2009 Marvell

Table 77: SVCR Bit Definitions (Continued)

	Physical Address 0x40F5_0098																															
	SVCR																SPMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			VCCSRAM_4					reserved			VCCSRAM_3					reserved			VCCSRAM_2					reserved			VCCSRAM_1				
Reset	?	?	?	1	1	0	1	1	?	?	?	1	1	0	1	1	?	?	?	1	0	0	1	1	?	?	?	0	1	1	1	1

Bits	Access	Name	Description
4:0	R/W	VCCSRAM_1	VCC_SRAM voltage output for Level 0 0x00 - 0.725V 0x01 - 0.750V 0x02 - 0.775V 0x03 - 0.800V 0x04 - 0.825V 0x05 - 0.850V 0x06 - 0.875V 0x07 - 0.900V 0x08 - 0.925V 0x09 - 0.950V 0x0A - 0.975V 0x0B - 1.000V 0x0C - 1.025V 0x0D - 1.050V 0x0E - 1.075V 0x0F - 1.100V 0x10 - 1.125V 0x11 - 1.150V 0x12 - 1.175V 0x13 - 1.200V 0x14 - 1.225V 0x15 - 1.250V 0x16 - 1.275V 0x17 - 1.300V 0x18 - 1.325V 0x19 - 1.350V 0x1A - 1.375V 0x1B - 1.400V 0x1C - 1.425V 0x1D - 1.450V 0x1E - 1.475V 0x1F - 1.500V

6.13.13 Power Management Unit Voltage Change Control Register (PVCR)

Power Management Unit Voltage Change Control Register (PVCR), shown in Table 78, contains configuration bits that control the automatic voltage-change sequence. When using PWR_I^{2}C commands to control VCC_APPS and VCC_SRAM power supplies, set FVE and PVE before starting a frequency or power mode change.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 78: PVCR Bit Definitions

	Physical Address 0x40F5_0100																															
	PVCR																				SPMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FVE	PVE	Reserved															VCSA	Reserved							Slave Address						
Reset	0[1]	1[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	?	?	?	?	0	1	1	0	1	0	0

Bits	Access	Name	Description
31	R/W	FVE	Frequency/Voltage Change Enable Enables the PWR_I^{2}C to transmit hardcoded commands during a processor system-frequency change or application subsystem low-power mode entry and exit sequences where power supply voltage level changes are required. Changes include an application core PLL-frequency change, a processor system-clocks change, or an application subsystem D0, D1, D2 or S2 power-mode change. The FVE bit sends hardware-controlled PWR_I^{2}C commands to execute the voltage change. 0 = The voltage-change sequencer is not initiated to change power supply voltage levels. 1 = The voltage-change sequencer is initiated during a processor system frequency change or when the subsystem is entering or exiting low power states that require voltage level changes. **NOTE:** PVE and FVE must be programmed with the same values.
30	R/W	PVE	Power Change Enable Enables the PWR_I^{2}C to transmit commands to enable or disable the VCC_APPS and VCC_SRAM power supply when the application subsystem is entering D3 or when the SPMU is entering S2 or S3 low-power states. 0 = The voltage-change sequencer is not initiated to enable/disable power supplies. 1 = The voltage-change sequencer is initiated when entering or exiting low power modes when power supply enable/disable is required. **NOTE:** PVE and FVE must be programmed with the same values.
29:15	—	—	Reserved
14	R	VCSA	**Voltage-Change Sequencer Active** Indicates when the PWR_I^{2}C is transmitting. 0 = No voltage-change sequence is in progress. 1 = The voltage-change sequencer is actively communicating with the PWR_I^{2}C module and the external regulator. Software must not write any of the PWR_I^{2}C registers. Software can read PVCR register but must not write to it.
13:7	—	—	Reserved
6:0	R/W	Slave Address	Slave Address The seven-bit address of the external regulator's I^{2}C module (default is 0x34). Specifies the address of the external voltage regulator when commands are initiated by the application subsystem.

Table 78: PVCR Bit Definitions (Continued)

Physical Address 0x40F5_0100	PVCR																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FVE	PVE	Reserved															VCSA	Reserved							Slave Address						
Reset	0[1]	1[1]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	?	?	?	?	0	1	1	0	1	0	0

Bits	Access	Name	Description
1. Low power mode exit does not clear or set this bit			

6.13.14 Oscillator Configuration Register (OSCC)

The Oscillator Configuration Register (OSCC), defined in Table 79, controls the processor (13 MHz) oscillator and timekeeping oscillator configuration and outputs as follows:

Note

A Write to a Services Clock Control Unit register must be followed by Reads to the register until the register is updated with the expected values before proceeding

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 79: OSCC Bit Definitions

Physical Address 0x4135_0000	OSCC																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																			ROS	PEN	TENS3	TENS2	TENS0	VCXOST							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0[1]	0[1]	1[1]	1	1	1	0	0	1	0	1

Bits	Access	Name	Description
31:13	—	Reserved	Reserved
12	Read Only	ROS	Ring Oscillator Status 0 = 120 MHz ring oscillator output is not ready for use 1 = 120 MHz ring oscillator output is stabilized and ready for use
11	R/W	PEN	CLK_POUT Enable 0 = Disable CLK_POUT pin 1 = Enable CLK_POUT pin to output processor oscillator clock when in S0/D0/C0, S0/D0CS/C0, S0/D1/C2 and S0/D2/C2 states. All other states CLK_POUT is disabled.

Copyright © 2009 Marvell

Table 79: OSCC Bit Definitions (Continued)

	Physical Address 0x4135_0000																			OSCC												SCCU
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																			ROS	PEN	TENS3	TENS2	TENS0	VCXOST							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0[1]	0[1]	1[1]	1	1	1	0	0	1	0	1

Bits	Access	Name	Description
10	R/W	TENS3	CLK_TOUT S3 Enable 0 = Disable CLK_TOUT pin in S3 power state 1 = Enable CLK_TOUT pin to output timekeeping oscillator clock when in S3 power state
9	R/W	TENS2	CLK_TOUT S2 Enable 0 = Disable CLK_TOUT pin in S2 power state 1 = Enable CLK_TOUT pin to output timekeeping oscillator clock when in S2 power state
8	R/W	TENS0	CLK_TOUT S0 Enable 0 = Disable CLK_TOUT pin in S0 power state 1 = Enable CLK_TOUT pin to output timekeeping oscillator clock when in S0 power state
7:0	R/W	VCXOST	Determines the wait time for processor oscillator stabilization 0x0 = Wait time = one 32 KHz clock cycle 0x1 = Wait time = two 32 KHz clock cycle ... 0xE5 = Wait time = 230 32 KHz clock cycle 0xE6 = Reserved ... 0xFF = Reserved

1. Low-power mode exit does not change this bit.

6.13.15 Application Subsystem Power Status/Configuration Register (ASCR)

The Application Subsystem Power Status/Configuration Register (ASCR) register contains the status flags that are cleared by writing a 0b1. Writing 0b0 has no effect. System and GPIO resets return the ASCR bits to their reset values. RDH is set as a result of D3 entry.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 80: ASCR Bit Definitions

	Physical Address 0x40F4_0000																															
	ASCR																	APMU														
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RDH	Reserved																MTS			Reserved	MTS_S			Reserved					D1S	D2S	D3S
Reset	1	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	0	?	0	1	0	?	?	?	?	?	0	0	0

Bits	Access	Name	Description
31	R/W	RDH	RDH should be cleared as part of the software initialization coming out of any reset and coming out of D3. Refer to the “Low-Power Mode Operation” section in the Pins and Control Chapter for more information on the delayed release functionality.
30:15	—	—	Reserved
14:12	R/W	MTS	Maximum Turbo Setting 0b000 = Reserved 0b001 = VCC_APPS voltage to be set to support XN – 1 0b010 = VCC_APPS voltage to be set to support XN – 2 0b011 — 0b111 = Reserved **NOTE:** Software must change frequency after writing to this field.
11	—	—	Reserved
10:8	R/W	MTS_S	Maximum Turbo Setting_Status 0b000 = Reserved 0b001 = VCC_APPS voltage has been set to support XN – 1 0b010 = VCC_APPS voltage has been set to support XN – 2 0b011–0b111 = Reserved
7:3	—	—	Reserved
2	R/Write 1 to Clear	D1S	Application Subsystem D1 State Status 0 = The application subsystem has not been placed into D1 state since D1S was cleared by a reset or by software. 1 = The application subsystem is in D1 state.
1	R/Write 1 to Clear	D2S	Application Subsystem D2 State Status 0 = The application subsystem has not been placed into D2 state since D2S was cleared by a reset or by software. 1 = The application subsystem is in D2 state.
0	R/Write 1 to Clear	D3S	Application Subsystem D3 State Status 0 = The application subsystem has not been placed in D3 state since D3S was cleared by a reset or by software. 1 = The application subsystem is in D3 state.

6.13.16 Application Subsystem Reset Status Register (ARSR)

Application Subsystem Reset Status Register (ARSR) indicates the last sources of an application subsystem reset. Possible reset sources are: GPIO, watchdog timeout, low-power mode exit,

Copyright © 2009 Marvell

hardware or power-on reset. Each ARSR status bit is set by its source of reset. Write 0b1 to clear a bit.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 81: ARSR Bit Definitions

	Physical Address 0x40F4_0004																															
	ARSR																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												**GPR**	**LPMR**	**WDT**	**HWR**
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	**0**	**0**	**0**	**1**

Bits	Access	Name	Description
31:4	—	—	Reserved
3	R/Write 1 to Clear	GPR	GPIO Reset 0 = No GPIO reset since the last time software or power-on reset cleared this bit 1 = GPIO reset has occurred since the last time software or power-on reset cleared this bit
2	R/Write 1 to Clear	LPMR	S3 Low-Power Mode Exit Reset 0 = No LPMR reset has occurred since the last time software or power-on reset cleared this bit 1 = LPMR reset has occurred since the last time software or power-on reset cleared this bit For details on low-power mode exits, see the description of the Application Subsystem Power Status/Configuration Register (ASCR).
1	R/Write 1 to Clear	WDT	Watchdog Time-out Reset 0 = No watchdog time-out reset has occurred since the last time software or power-on reset cleared this bit 1 = Watchdog time-out reset has occurred since the last time software or power-on reset cleared this bit.
0	R/Write 1 to Clear	HWR	Hardware/Power-On Reset 0 = No hardware/power-on reset has occurred since the last time software cleared this bit 1 = Hardware/power-on reset has occurred since the last time software cleared this bit.

6.13.17 Application Subsystem Wakeup from D3 Enable Register (AD3ER)

Application Subsystem Wakeup from D3 Enable Register (AD3ER) selects whether the corresponding wake-up sources cause an application subsystem wakeup from D3 to D0 state.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 82: AD3ER Bit Definitions

	Physical Address 0x40F4_0008								AD3ER								APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WE_RTC	WE_OST	WE_TSI	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														WE_EXTERNAL[1]	WE_EXTERNAL[0]
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	WE_RTC	Wake-Up Options for RTC 0 = Disable wake-up due to RTC alarm. 1 = Enable wake-up due to RTC alarm.
30	R/W	WE_OST	Wake-Up Options for OS Timer 0 = Disable wake-up due to timer event. 1 = Enable wake-up due to timer event.
29	R/W	WE_TSI	Wake-Up Options for Touch Screen Interface (PXA32x Only) 0 = Disable wake-up due to TSI event. 1 = Enable wake-up due to TSI event. Note: This value needs to be programmed with the same value as AD1D0ER[WE_TSI].
	—	—	Reserved (PXA30x and PXA31x Only)
28	R/W	WE_DMUX1	Wake-up Options for USB Host Port 1 (PXA32x and PXA30x Only) 0 = No wake-up due to USB Host Port 1 1 = Wake-up due to USB Host Port 1
	—	—	Reserved (PXA31x Only)

Copyright © 2009 Marvell

Table 82: AD3ER Bit Definitions (Continued)

	Physical Address 0x40F4_0008								AD3ER								APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WE_RTC	WE_OST	WE_TSI	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														WE_EXTERNAL[1]	WE_EXTERNAL[0]
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
27	—	—	Reserved
26	R/W	WE_USB2	Wake-Up Options for USB Client 2.0 Port 0 = Disable wake-up due to USB Client 2.0 port. 1 = Enable wake-up due to USB Client 2.0 port.
25:24	—	—	Reserved
23	R/W	WE_DMUX3	Wake-up Options for USB Host Port 3 0 = No wake-up due to USB Host Port 3 1 = Wake-up due to USB Host Port 3
22	R/W	WE_DMUX2	Wake-up Options for USB Host Port 2. 0 = No wake-up due to USB Host Port 2 1 = Wake-up due to USB Host Port 2 **NOTE:** Applies to the USB Device Controller (UDC) for the PXA32x and PXA30x processors.
21	R/W	WE_KP	Wake-Up Options for a Keypad Direct Key 0 = Disable wake-up due to an edge detect on a direct keypad Input. 1 = Enable wake-up due to an edge detect on a direct keypad Input.

Table 82: AD3ER Bit Definitions (Continued)

	Physical Address 0x40F4_0008								AD3ER								APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WE_RTC	WE_OST	WE_TSI	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														WE_EXTERNAL[1]	WE_EXTERNAL[0]
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
20	R/W	WE_USIM1	Wake-Up Options for USIM Port 1 0 = Disable wake-up due to USIM port 1event. 1 = Enable wake-up due to USIM port 1 event.
19	R/W	WE_USIM0	Wake-Up Options for USIM Port 0 0 = Disable wake-up due to USIM port 0 event. 1 = Enable wake-up due to USIM port 0 event.
18:17	—	—	Reserved
16	R/W	WE_OTG	Wake-up Options for USBOTG (PXA32x and PXA30x Only) 0 = Disable wake-up due to USBOTG port 0 event. 1 = Enable wake-up due to USBOTG port 0 event.
	—	—	Reserved (PXA31x Only)
15:2	R/W	WE_Generic [n]	Wake-Up Options for generic event inputs, where n = 13-0 0 = Disable wake-up due to generic event[n] edge detect. 1 = Enable wake-up due to generic event[n] edge detect. Refer to Section 6.11.3.2, Generic Wakeups for more information on generic wake-up events.

Copyright © 2009 Marvell

Table 82: AD3ER Bit Definitions (Continued)

	Physical Address 0x40F4_0008								AD3ER								APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WE_RTC	WE_OST	WE_TSI	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														WE_EXTERNAL[1]	WE_EXTERNAL[0]
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	R/W	WE_EXTERNAL[1]	**Wake-Up Options for EXT_WAKEUP<1>** (PXA32x Only) 0 = Disable wake-up due to EXT_WAKEUP<1> edge detect. 1 = Enable wake-up due to EXT_WAKEUP<1> edge detect.
	—	—	Reserved (PXA30x and PXA31x Only)
0	R/W	WE_EXTERNAL[0]	**Wake-Up Options for EXT_WAKEUP<0>** 0 = Disable wake-up due to EXT_WAKEUP<0> edge detect. 1 = Enable wake-up due to EXT_WAKEUP<0> edge detect.

6.13.18 Application Subsystem Wakeup from D3 Status Register (AD3SR)

Application Subsystem Wakeup from D3 Status Register (AD3SR) indicates which sources cause an application subsystem wakeup from D3 to D0 state. These bits are cleared by writing 0b1 to them. Writing 0b0 to any status bit has no effect.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 83: AD3SR Bit Definitions

	Physical Address 0x40F4_000C																															
	AD3SR																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WS_RTC	WS_OST	WS_TSI	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														WS_EXTERNAL[1]	WS_EXTERNAL[0]
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved			WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	0	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/Write 1 to clear	WS_RTC	Wake-Up Status for RTC 0 = No wake-up due to RTC alarm. 1 = Wake-up due to RTC alarm.
30	R/Write 1 to clear	WS_OST	Wake-Up Status for OS Timer 0 = No wake-up due to timer event. 1 = Wake-up due to timer event.
29	R/Write 1 to clear	WS_TSI	Wake-Up Status for Touch Screen Interface (PXA32x Only) 0 = No wake-up due to TSI event. 1 = Wake-up due to TSI event.
	—	—	Reserved (PXA31x and PXA30x Only)
28	R/W	WS_DMUX1	Wake-up Status for USB Host Port 1 (PXA32x and PXA30x Only) 0 = No wake-up due to USB Host Port 1 1 = Wake-up due to USB Host Port 1
	—	—	Reserved (PXA31x Only)
27	—	—	Reserved

Copyright © 2009 Marvell

Table 83: AD3SR Bit Definitions (Continued)

	Physical Address 0x40F4_000C																AD3SR								APMU							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WS_RTC	WS_OST	WS_TSI	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														WS_EXTERNAL[1]	WS_EXTERNAL[0]
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved			WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	0	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
26	R/Write 1 to clear	WS_USB2	Wake-Up Status for USB Client 2.0 Port 0 = No wake-up due to USB Client 2.0 port. 1 = Wake-up due to USB Client 2.0 port.
25:24	—	—	Reserved
23	R/Write 1 to clear	WS_DMUX3	Wake-up Status for USB Host Port 3 0 = No wake-up due to USB Host Port 3 1 = Wake-up due to USB Host Port 3
22	R/Write 1 to clear	WS_DMUX2	Wake-up Status for USB Host Port 2. 0 = No wake-up due to USB Host Port 2 1 = Wake-up due to USB Host Port 2
21	R/Write 1 to clear	WS_KP	Wake-Up Status for a Keypad Direct Key 0 = No wake-up due to an edge detect on a direct keypad Input. 1 = Wake-up due to an edge detect on a direct keypad Input.
20	R/Write 1 to clear	WS_USIM1	Wake-Up Status for USIM Port 1 0 = No wake-up due to USIM port 1event. 1 = Wake-up due to USIM port 1 event.

Copyright © 2009 Marvell

Table 83: AD3SR Bit Definitions (Continued)

	Physical Address 0x40F4_000C																AD3SR								APMU							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WS_RTC	WS_OST	WS_TSI	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														WS_EXTERNAL[1]	WS_EXTERNAL[0]
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved			WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	0	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
19	R/Write 1 to clear	WS_USIM0	Wake-Up Status for USIM Port 0 0 = No wake-up due to USIM port 0 event. 1 = Wake-up due to USIM port 0 event.
18:17	—	—	Reserved
16	R/W	WS_OTG	Wake-up Status for USBOTG (PXA32x and PXA30x Only) 0 = No wake-up due to USBOTG port 0 event. 1 = Wake-up due to USBOTG port 0 event.
	—	—	Reserved (PXA31x Only)
15:2	R/Write 1 to clear	WS_GENERIC[n]	Wake-Up Status for Generic Event Inputs, where n = 1–13 0 = No wake-up due to generic event[n] edge detect. 1 = Wake-up due to generic event[n] edge detect.
1	R/Write 1 to clear	WS_EXTERNAL[1]	Wake-up Status for external event input (EXT_WAKEUP<1>) (PXA32x Only) 0 = No wake-up due to EXT_WAKEUP<1> edge detect. 1 = Wake-up due to EXT_WAKEUP<1> edge detect. **NOTE:** These are communicated through the services unit
	—	—	Reserved (PXA30x and PXA31x Only)

Copyright © 2009 Marvell

Table 83: AD3SR Bit Definitions (Continued)

	Physical Address 0x40F4_000C								AD3SR								APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	WS_RTC	WS_OST	WS_TSI	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														WS_EXTERNAL[1]	WS_EXTERNAL[0]
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved			WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	0	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
0	R/Write 1 to clear	WS_EXTERNALL[0]	Wake-up Status for external event input (EXT_WAKEUP<0>) 0 = No wake-up due to EXT_WAKEUP<0> edge detect. 1 = Wake-up due to EXT_WAKEUP<0> edge detect. **NOTE:**

6.13.19 Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER)

Application Subsystem Wakeup from D2 to D0 State Enable Register (AD2D0ER) selects whether the corresponding wakeup sources cause an application subsystem wake-up from D2 to D0 state.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 84: AD2D0ER Bit Definitions

	Physical Address 0x40F4_0010																AD2D0ER								APMU							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	WE_RTC	Wake-Up Options for RTC 0 = Disable wake-up due to RTC alarm. 1 = Enable wake-up due to RTC alarm.
30	R/W	WE_OST	Wake-Up Options for OS Timer 0 = Disable wake-up due to timer event. 1 = Enable wake-up due to timer event.
29	—	—	Reserved
28	R/W	WE_DMUX1	Wake-up Options for USB Host Port 1 (PXA30x Only) 0 = No wake-up due to USB Host Port 1 1 = Wake-up due to USB Host Port 1
	—	—	Reserved (PXA31x Only)
27	—	—	Reserved
26	R/W	WE_USB2	Wake-Up Options for USB Client 2.0 Port 0 = Disable wake-up due to USB Client 2.0 port. 1 = Enable wake-up due to USB Client 2.0 port.
25:24	—	—	Reserved
23	R/W	WE_DMUX3	Wake-up Options for USB Host Port 3 0 = No wake-up due to USB Host Port 3 1 = Wake-up due to USB Host Port 3

Copyright © 2009 Marvell

Table 84: AD2D0ER Bit Definitions (Continued)

Physical Address 0x40F4_0010 **AD2D0ER** **APMU**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
22	R/W	WE_DMUX2	Wake-up Options for USB Host Port 2. 0 = No wake-up due to USB Host Port 2 1 = Wake-up due to USB Host Port 2 **NOTE:** Applies to the USB Device Controller (UDC) for the PXA30x processor.
21	R/W	WE_KP	Wake-Up Options for a Keypad Direct Key 0 = Disable wake-up due to an edge detect on a direct keypad Input. 1 = Enable wake-up due to an edge detect on a direct keypad Input.
20	R/W	WE_USIM1	Wake-Up Options for USIM Port 1 0 = Disable wake-up due to USIM port 1event. 1 = Enable wake-up due to USIM port 1 event.
19	R/W	WE_USIM0	Wake-Up Options for USIM Port 0 0 = Disable wake-up due to USIM port 0 event. 1 = Enable wake-up due to USIM port 0 event.
18:17	—	—	Reserved
16	R/W	WE_OTG	Wake-up Options for USBOTG (PXA30x Only) 0 = Disable wake-up due to USBOTG port 0 event. 1 = Enable wake-up due to USBOTG port 0 event.
	—	—	Reserved (PXA31x Only)
15:2	R/W	WE_Generic [n]	Wake-Up Options for generic event inputs, where n = 13-0 0 = Disable wake-up due to generic event[n] edge detect. 1 = Enable wake-up due to generic event[n] edge detect. Refer to Section 6.11.3.2, Generic Wakeups for more information on generic wake-up events.
1	—	—	Reserved

Table 84: AD2D0ER Bit Definitions (Continued)

	Physical Address 0x40F4_0010												AD2D0ER				APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved			WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	Reserved		WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
0	R/W	WE_EXTERNAL[0]	**Wake-Up Options for EXT_WAKEUP<0>** 0 = Disable wake-up due to EXT_WAKEUP<0> edge detect. 1 = Enable wake-up due to EXT_WAKEUP<0> edge detect.

6.13.20 Application Subsystem Wakeup from D2 to D0 Status Register (AD2D0SR)

Application Subsystem Wakeup from D2 to D0 Status Register (AD2D0SR) indicates which sources caused an application subsystem wakeup from D2 to D0 state.These bits are cleared by writing 0b1 to them. Writing 0b0 to any status bit has no effect.

Copyright © 2009 Marvell

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 85: AD2D0SR Bit Definitions

	Physical Address 0x40F4_0014								AD2D0SR								APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved			WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0

Bits	Access	Name	Description
31	R/Write 1 to clear	WS_RTC	Wake-up Status for RTC 0 = No wake-up due to RTC alarm. 1 = Wake-up due to RTC alarm.
30	R/Write 1 to clear	WS_OST	Wake-Up Status for OS Timer 0 = No wake-up due to timer event. 1 = Wake-up due to timer event.
29	—	—	Reserved
28	R/W	WS_DMUX1	Wake-up Status for USB Host Port 1 (PXA30x Only) 0 = No wake-up due to USB Host Port 1 1 = Wake-up due to USB Host Port 1
	—	—	Reserved (PXA31x Only)
27	—	—	Reserved
26	R/Write 1 to clear	WS_USB2	Wake-Up Status for USB Device 2.0 Port 0 = No wake-up due to USB Device 2.0 port. 1 = Wake-up due to USB Device 2.0 port.
25:24	—	—	Reserved
23	R/Write 1 to clear	WS_DMUX3	Wake-up Status for USB Host Port 3 0 = No wake-up due to USB Host Port 3 1 = Wake-up due to USB Host Port 3
22	R/Write 1 to clear	WS_DMUX2	Wake-up Status for USB Host Port 2. 0 = No wake-up due to USB Host Port 2 1 = Wake-up due to USB Host Port 2

Table 85: AD2D0SR Bit Definitions (Continued)

Physical Address 0x40F4_0014	AD2D0SR																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved			WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	Reserved		WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0

Bits	Access	Name	Description
21	R/Write 1 to clear	WS_KP	Wake-Up Status for a Keypad Direct Key 0 = No wake-up due to an edge detect on a direct keypad Input. 1 = Wake-up due to an edge detect on a direct keypad Input.
20	R/Write 1 to clear	WS_USIM1	Wake-Up Status for USIM Port 1 0 = No wake-up due to USIM port 1event. 1 = Wake-up due to USIM port 1 event.
19	R/Write 1 to clear	WS_USIM0	Wake-Up Status for USIM Port 0 0 = No wake-up due to USIM port 0 event. 1 = Wake-up due to USIM port 0 event.
18:17	—	—	Reserved
16	R/W	WS_OTG	Wake-up Status for USBOTG (PXA30x Only) 0 = No wake-up due to USBOTG port 0 event. 1 = Wake-up due to USBOTG port 0 event.
	—	—	Reserved (PXA31x Only)
15:2	R/Write 1 to clear	WS_GENERIC[n]	Wake-Up Status for Generic Event Inputs, where n = 1–13 0 = No wake-up due to generic event[n] edge detect. 1 = Wake-up due to generic event[n] edge detect.
1	—	—	Reserved
0	R/Write 1 to clear	WS_EXTERNALL[0]	Wake-up Status for external event input (EXT_WAKEUP<0>) 0 = No wake-up due to EXT_WAKEUP<0> edge detect. 1 = Wake-up due to EXT_WAKEUP<0> edge detect.

Copyright © 2009 Marvell

6.13.21 Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER)

Application Subsystem Wakeup from D2 to D1 State Enable Register (AD2D1ER) selects whether the corresponding wakeup sources cause an application subsystem wakeup from D2 to D1 state. Unlike the other Power Transition Enable registers, the D2-to-D1 transition can happen only as a result of real-time clock alarm.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 86: AD2D1ER Bit Definitions

Physical Address 0x40F4_0018	AD2D1ER	APMU

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA31x and PXA30x Only	WE_RTC	Reserved																														
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31	R/W	WE_RTC	Wake-up options for RTC 0 = Disable wake-up due to RTC alarm. 1 = Enable wake-up due to RTC alarm.
30:0	—	—	Reserved

6.13.22 Application Subsystem Wakeup from D2 to D1 Status Register (AD2D1SR)

Application Subsystem Wakeup from D2 to D1 Status Register (AD2D1SR) indicates which sources caused an application subsystem wakeup from D2 to D1 state. These bits are cleared by writing 0b1 to them. Writing 0b0 to any status bit has no effect.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 87: AD2D1SR Bit Definitions

	Physical Address 0x40F4_001C																															
	AD2D1SR																				APMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x and PXA30x Only	WS_RTC	Reserved																														
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31	R/Write 1 to clear	WS_RTC	Wake-up Status for RTC 0 = No wake-up due to RTC alarm. 1 = Wake-up due to RTC alarm.
30:0	—	—	Reserved

6.13.23 Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER)

Application Subsystem Wakeup from D1 to D0 State Enable Register (AD1D0ER) selects whether the corresponding wakeup sources caused an application subsystem wakeup from the D1 to D0 state.

Copyright © 2009 Marvell

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 88: AD1D0ER Bit Definitions

	Physical Address 0x40F4_0020																															
	AD1D0ER																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	WE_MLCD	Reserved		WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	WE_MLCD	Reserved	WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	WE_RTC	Wake-up options for RTC 0 = Disable wake-up due to RTC alarm. 1 = Enable wake-up due to RTC alarm.
30	R/W	WE_OST	Wake-Up Options for OS Timer 0 = Disable wake-up due to timer event. 1 = Enable wake-up due to timer event.
29	—	—	Reserved
28	R/W	WE_DMUX1	Wake-up Options for USB Host Port 1 (PXA30x Only) 0 = No wake-up due to USB Host Port 1 1 = Wake-up due to USB Host Port 1
	—	—	Reserved (PXA31x Only)
27	—	—	Reserved
26	R/W	WE_USB2	Wake-up options for USB Device 2.0 port 0 = Disable wake-up due to USB Device 2.0 port. 1 = Enable wake-up due to USB Device 2.0 port.
25:24	—	—	Reserved
23	R/W	WE_DMUX3	Wake-up options for USB Host Port 3 0 = No wake-up due to USB Host Port 3 1 = Wake-up due to USB Host Port 3
22	R/W	WE_DMUX2	Wake-up options for USB Host Port 2. 0 = No wake-up due to USB Host Port 2 1 = Wake-up due to USB Host Port 2

Table 88: AD1D0ER Bit Definitions (Continued)

Physical Address 0x40F4_0020 — AD1D0ER — APMU

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA31x Only	WE_RTC	WE_OST	Reserved			WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	WE_MLCD	Reserved		WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
PXA30x Only	WE_RTC	WE_OST	Reserved	WE_MUX1	Reserved	WE_USB2	Reserved		WE_MUX3	WE_MUX2	WE_KP	WE_USIM1	WE_USIM0	WE_MLCD	Reserved	WE_OTG	WE_GENERIC <13:0>														Reserved	WE_EXTERNAL[0]
Reset	0	0	?	0	?	0	?	?	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
21	R/W	WE_KP	Wake-up options for a Keypad Direct Key 0 = Disable wake-up due to an edge detect on a a direct keypad Input. 1 = Enable wake-up due to an edge detect on a a direct keypad Input.
20	R/W	WE_USIM1	Wake-up options for USIM Port 1 0 = Disable wake-up due to USIM Port 1event. 1 = Enable wake-up due to USIM Port 1 event.
19	R/W	WE_USIM0	Wake-up options for USIM Port 0 0 = Disable wake-up due to USIM Port 0 event. 1 = Enable wake-up due to USIM Port 0 event.
18	R/W	WE_MLCD	Wake-Up Options for Mini-LCD 0 = Disable wake-up due to Mini-LCD Frame Count event. 1 = Enable wake-up due to Mini-LCD Frame Count event.
17	—	—	Reserved
16	R/W	WE_OTG	Wake-up Options for USBOTG (PXA30x Only) 0 = Disable wake-up due to USBOTG port 0 event. 1 = Enable wake-up due to USBOTG port 0 event.
	—	—	Reserved (PXA31x Only)
15:2	R/W	WE_GENERIC[n]	Wake-up options for generic event inputs, where n = 0 – 13 0 = Disable wake-up due to generic event[n] edge detect. 1 = Enable wake-up due to generic event[n] edge detect.
1	—	—	Reserved
0	R/W	WE_EXTERNAL[0]	Wake-up options for external event input (EXT_WAKEUP<0>) 0 = Disable wake-up due to EXT_WAKEUP<0> edge detect. 1 = Enable wake-up due to EXT_WAKEUP<0> edge detect.

Copyright © 2009 Marvell

6.13.24 Application Subsystem Wakeup from D1 to D0 Status Register (AD1D0SR)

Application Subsystem Wakeup from D1 to D0 Status Register (AD1D0SR) indicates which sources caused an application subsystem wakeup from D1 to D0 state. These bits are cleared by writing 0b1 to them. Writing 0b0 to any status bit has no effect.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 89: AD1D0SR Bit Definitions

Physical Address 0x40F4_0024																																	
AD1D0SR																																	
APMU																																	
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	WS_MLCD	Reserved		WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	WS_MLCD	Reserved	WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	?	?	?	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0

Bits	Access	Name	Description
31	R/Write 1 to clear	WS_RTC	Wake-up Status for RTC 0 = No wake-up due to RTC alarm. 1 = Wake-up due to RTC alarm.
30	R/Write 1 to clear	WS_OST	Wake-up Status for OS Timer 0 = No wake-up due to timer event. 1 = Wake-up due to timer event.
29	—	—	Reserved
28	R/W	WS_DMUX1	Wake-up Status for USB Host Port 1 (PXA30x Only) 0 = No wake-up due to USB Host Port 1 1 = Wake-up due to USB Host Port 1
	—	—	Reserved (PXA31x Only)
27	—	—	Reserved
26	R/Write 1 to clear	WS_USB2	Wake-up Status for USB Device 2.0 port 0 = No wake-up due to USB Device 2.0 port. 1 = Wake-up due to USB Device 2.0 port.
25:24	—	—	Reserved

Table 89: AD1D0SR Bit Definitions (Continued)

	Physical Address 0x40F4_0024																															
	AD1D0SR																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	WS_MLCD	Reserved		WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	WS_MLCD	Reserved	WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	?	?	?	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0

Bits	Access	Name	Description
23	R/Write 1 to clear	WS_DMUX3	Wake-up Status for a rising edge from USB Host Port 3 0 = No wake-up due to USB Host Port 3 1 = Wake-up due to USB Host Port 3
22	R/Write 1 to clear	WS_DMUX2	Wake-up Status for a rising edge from USB Host Port 2. 0 = No wake-up due to USB Host Port 2 1 = Wake-up due to USB Host Port 2
21	R/Write 1 to clear	WS_KP	Wake-up Status for a Keypad Direct Key 0 = No wake-up due to an edge detect on a direct keypad Input. 0 = Wake-up due to an edge detect on a direct keypad Input.
20	R/Write 1 to clear	WS_USIM1	Wake-up Status for USIM Port 1 0 = No wake-up due to USIM Port 1 event. 1 = Wake-up due to USIM Port 1 event.
19	R/Write 1 to clear	WS_USIM0	Wake-up Status for USIM Port 0 0 = No wake-up due to USIM Port 0 event. 1 = Wake-up due to USIM Port 0 event.
18	R/Write 1 to clear	WS_MLCD	Wake-Up Status for Mini-LCD 0 = No wake-up due to Mini-LCD Frame Count event. 1 = Wake-up due to Mini-LCD Frame Count event.
17	—	—	Reserved
16	R/W	WS_OTG	Wake-up Status for USBOTG (PXA30x Only) 0 = No wake-up due to USBOTG port 0 event. 1 = Wake-up due to USBOTG port 0 event.
	—	—	Reserved (PXA31x Only)

Copyright © 2009 Marvell

Table 89: AD1D0SR Bit Definitions (Continued)

	Physical Address 0x40F4_0024																															
	AD1D0SR																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	WS_RTC	WS_OST	Reserved			WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	WS_MLCD	Reserved		WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
PXA30x Only	WS_RTC	WS_OST	Reserved	WS_MUX1	Reserved	WS_USB2	Reserved		WS_MUX3	WS_MUX2	WS_KP	WS_USIM1	WS_USIM0	WS_MLCD	Reserved	WS_OTG	WS_GENERIC <13:0>														Reserved	WS_EXTERNAL[0]
Reset	0	0	?	?	?	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0

Bits	Access	Name	Description
15:2	R/Write 1 to clear	WS_GENERIC[n]	Wake-up Status for generic event inputs where n = 0 – 13 0 = No wake-up due to generic event[n] edge detect. 1 = Wake-up due to generic event[n] edge detect.
1	—	—	Reserved
0	R/Write 1 to clear	WS_EXTERNAL[0]	Wake-up Status for external event input (EXT_WAKEUP<0>) 0 = No wake-up due to EXT_WAKEUP<0> edge detect. 1 = Wake-up due to EXT_WAKEUP<0> edge detect.

6.13.25 Application Subsystem General Purpose Register (AGENP)

Application Subsystem General Purpose Register (AGENP) is used to determine the status of the 120 MHz ring oscillator by reading the AGENP[RO_ST] bit. The AGENP[RO_CTRL] is used to enable and disable the 120 MHz ring oscillator.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 90: AGENP Bit Definitions

	Physical Address 0x40F4_002C																															
											AGENP										APMU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RO_CTRL	RO_ST	DMCPLL_CTRL	DMCPLL_ST	Reserved																											
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31	R/W	RO_CTRL	A toggle bit to turn off the Ring Oscillator depending on RO_ST bit. 0 = Ignored. 1 = When set, turns off or on depending on the status of RO_ST bit. **NOTE:** This bit is cleared when the Ring Oscillator is turned off/on.
30	R/W	RO_ST	Indicates the status of the Ring Oscillator. 0 = Ring Oscillator is turned off 1 = Ring Oscillator is turned on.
29	R/W	DMCPLL_CTRL	A toggle bit to turn off the DMC PLL depending on DMCPLL_ST bit. 0 = Ignored. 1 = When set, turns off/on DMC PLL depending on the status of the DMCPLL_ST bit. **NOTE:** This bit is cleared after the DMC PLL is turned off/on.
28	R/W	DMCPLL_ST	This bit indicates the status of the DMC PLL. 0 = GBPLL is turned off 1 = GBPLL is turned on.
27:0	—	—	Reserved

6.13.26 Application Subsystem D3 Configuration Register (AD3R)

Application Subsystem D3 Configuration Register (AD3R), defined in Table 91, controls the application subsystem behavior in D3 state. The D3-state unit-operation bits select which internal SRAM arrays (associated non-SRAM array logic is not powered) used only by the application subsystem retain their states when the application subsystem is in D3 state. All units except the PMU are reset. State operation bits in AD3R affects only memories used by the application subsystem.

Copyright © 2009 Marvell

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 91: AD3R Bit Definitions

	Physical Address 0x40F4_0030																															
	AD3R																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved																										AD3_R5	AD3_R4	AD3_R3	AD3_R2	AD3_R1	AD3_R0
PXA31x and PXA30x Only	Reserved																												AD3_R3	Reserved		AD3_R0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0

Bits	Access	Name	Description
31:6	—	—	Reserved
5	R/W	AD3_R5	D3 state unit operation — application subsystem internal SRAM upper bank array 2 (PXA32x Only) 0 = Upper bank array 2 is off when APMU is in D3. 1 = Upper bank array 2 retains state when APMU is (memory array only) in D3.
	—	—	Reserved (PXA31x and PXA30x Only)
4	R/W	AD3_R4	D3 state unit operation — application subsystem internal SRAM upper bank array 1 (PXA32x Only) 0 = Upper bank array 1 is off when APMU is in D3. 1 = Upper bank array 1retains state when APMU is (memory array only) in D3.
	—	—	Reserved (PXA31x and PXA30x Only)
3	R/W	AD3_R3	D3 state unit operation — application subsystem internal SRAM upper bank array 0 0 = Upper bank array 0 is off when APMU is in D3. 1 = Upper bank array 0 retains state when APMU is (memory array only) in D3.
2	R/W	AD3_R2	D3 state unit operation — application subsystem internal SRAM lower bank array 2 (PXA32x Only) 0 = Lower bank array 2 is off when APMU is in D3. 1 = Lower bank array 2retains state when APMU is (memory array only) in D3.
	—	—	Reserved (PXA31x and PXA30x Only)
1	R/W	AD3_R1	D3 state unit operation, application subsystem internal SRAM lower bank array 1 (PXA32x Only) 0 = Lower bank array 1 is off when APMU is in D3. 1 = Lower bank array 1 retains state when APMU is (memory array only) in D3.
	—	—	Reserved (PXA31x and PXA30x Only)

Table 91: AD3R Bit Definitions (Continued)

	Physical Address 0x40F4_0030 AD3R APMU																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved																										AD3_R5	AD3_R4	AD3_R3	AD3_R2	AD3_R1	AD3_R0
PXA31x and PXA30x Only	Reserved																												AD3_R3	Reserved		AD3_R0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0

Bits	Access	Name	Description
0	R/W	AD3_R0	D3 state unit operation, application subsystem internal SRAM lower bank array 0 0 = Lower bank array 0 is off when APMU is in D3. 1 = Lower bank array 0 retains state when APMU is (memory array only) in D3.

6.13.27 Application Subsystem D2 Configuration Register (AD2R)

Application Subsystem D2 Configuration Register (AD2R), defined in Table 92, contains the D2state unit operation bits for memory arrays used by the application subsystem. These bits select which memory arrays retain state and operation when the application subsystem is in D2 state. Disabled unit registers take their reset values at D2 exit. The core and all peripherals retain state in D2 state.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 92: AD2R Bit Definitions

	Physical Address 0x40F4_0034 AD2R APMU																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A31x and PXA30x Only	Reserved																							SETALWAYS	Reserved				AD2_R3	Reserved		AD2_R0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0

Bits	Access	Name	Description
31:9	—	—	Reserved
8	—	—	SETALWAYS -- Must be set to 0b1
7:4	—	—	Reserved

Copyright © 2009 Marvell

Table 92: AD2R Bit Definitions (Continued)

	Physical Address 0x40F4_0034																															
	AD2R																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A31x and PXA30x Only	Reserved																							SETALWAYS	Reserved				AD2_R3	Reserved		AD2_R0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0

Bits	Access	Name	Description
3	R/W	AD2_R3	D2 state unit operation — application subsystem internal SRAM upper bank array 0 0 = Upper bank array 0 is off when APMU is in D2. 1 = Upper bank array 0 retains state when APMU is (memory array only) in D3.
2:1	—	—	Reserved
0	R/W	AD2_R0	D2 state unit operation, application subsystem internal SRAM lower bank array 0 0 = Lower bank array 0 is off when APMU is in D2. 1 = Lower bank array 0 retains state when APMU is (memory array only) in D2.

6.13.28 Application Subsystem D1 Configuration Register (AD1R)

Application Subsystem D1 Configuration Register (AD1R), defined in Table 93, contains the D1-mode unit operational bits. These bits select which units retain full operation when the application subsystem is in D1 state. Disabled units are inaccessible in D1 state but retain state. In D1 state, the core retains state and both PLLs are turned off.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 93: AD1R Bit Definitions

	Physical Address 0x40F4_0038																															
	AD1R																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x and PXA30x Only	Reserved																							SETALWAYS	Reserved				AD1_R3	Reserved		AD1_R0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:9	—	—	Reserved

Table 93: AD1R Bit Definitions (Continued)

	Physical Address 0x40F4_0038																															
	AD1R																APMU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x and PXA30x Only	Reserved																							SETALWAYS	Reserved				AD1_R3	Reserved		AD1_R0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
8	—	—	SETALWAYS - Must be set to 0b1
7:4	—	—	Reserved
3	R/W	AD1_R3	D1 state unit operation — application subsystem internal SRAM upper bank array 0 0 = Upper bank array 0 is off when APMU is in D1. 1 = Upper bank array 0 retains state when APMU is (memory array only) in D1.
2:1	—	—	Reserved
0	R/W	AD1_R0	D1 state unit operation, application subsystem internal SRAM lower bank array 0 0 = Lower bank array 0 is off when APMU is in D1. 1 = Lower bank array 0 retains state when APMU is (memory array only) in D1.

6.13.29 Application Subsystem Clock Configuration Register (ACCR)

The core clock is the base clock from which the core frequency is derived. The Application Subsystem Clock Configuration Register (ACCR), defined in Table 94, controls this frequency.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 94: ACCR Bit Definitions

	Physical Address 4134_0000																															
	ACCR																ACCU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA31x Only	XPDIS	SPDIS	VAUF		Reserved	D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	0	0	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA30x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			Reserved			XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
31	R/W	XPDIS	Core PLL Output Disable: 0 = Core PLL is enabled and may be used after a frequency change 1 = Core PLL is disabled and will not be used after a frequency change **NOTE:** The Core PLL can only be disabled after entering D0CS mode. Software must verify that the PLL is disabled (ACSR[XPDIS_S = 1). The Core PLL must be enabled before exiting D0CS mode. Software must verify that the PLL is locked and ready for use (ACSR[XPCLK] = 1)
30	R/W	SPDIS	System PLL Output Disable: 0 = System PLL is enabled and may be used after a frequency change 1 = System PLL is disabled and will not be used after a frequency change **NOTE:** The System PLL can only be disabled after entering D0CS mode. Software must verify that the PLL is disabled (ACSR[SPDIS_S] = 1). The System PLL must be enabled before exiting D0CS mode. Software must verify that the PLL is locked and ready for use (ACSR[SPCLK] = 1)
29:28	R/W	VAUF	Video Accelerator Unit Frequency Select 0b00 -- 104 MHz 0b01 -- 156MHz 0b10 -- 208 MHz 0b11 -- 78 MHz **NOTE:** These bit fields are only available in the PXA31x Processor
	—	—	Reserved (PXA32x and PXA30x Only)

Table 94: ACCR Bit Definitions (Continued)

	Physical Address 4134_0000																															
	ACCR																				ACCU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA31x Only	XPDIS	SPDIS	VAUF		Reserved	D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	0	0	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA30x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			Reserved			XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
27	—	—	Reserved
26	R/W	D0CS	D0 Mode Clock Select: 0 = Use the core PLL and system PLLs as the processor clock sources in D0 mode. 1 = Use the ring oscillator as the processor clock source in D0 mode. **NOTE:** Must not be cleared until after an interrupt has been generated indicating that the PLLs are locked or status of PLLs are updated.
25:23	R/W	SMCFS	Static Memory Controller Frequency Select Selects the frequency of the static memory controller. 0b000 — 78 MHz 0b010 — 104 MHz 0b101 — 208 MHz All other values are reserved
22:20	—	—	Reserved
19:18	R/W	SFLFS	Frequency select for the Internal Memory Controller (SRAM) Selects the frequency for the Internal Memory Controller (SRAM). 0b00 — 104 MHz 0b01 — 156 MHz 0b10 — 208 MHz 0b11 — 312 MHz

Copyright © 2009 Marvell

Table 94: ACCR Bit Definitions (Continued)

Physical Address 4134_0000 **ACCR** **ACCU**

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA31x Only	XPDIS	SPDIS	VAUF		Reserved	D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	0	0	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA30x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			Reserved			XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
17:16	R/W	XSPCLK	Core Frequency During Frequency Change Selects the nominal system PLL-derived clock frequency used by the core during core PLL frequency-change operations. 0b00 — 156 MHz clock 0b01 — Reserved 0b10 — Reserved 0b11 — No clock to core until core PLL is re-locked
15:14	R/W	HSS	System Bus-Clock Frequency Select: Selects the nominal System bus-clock frequency. 0b00 — 104 MHz 0b01 — 156 MHz 0b10 — 208 MHz 0b11 — 312 MHz

Table 94: ACCR Bit Definitions (Continued)

	Physical Address 4134_0000																															
	ACCR																				ACCU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA31x Only	XPDIS	SPDIS	VAUF		Reserved	D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	0	0	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA30x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			Reserved			XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
13:12	R/W	DMCFS	Dynamic Memory Controller Clock Frequency Select: Selects the frequency of the dynamic memory controller (DMC). 0b00 — 26 MHz 0b01 — 312 MHz (PXA32x and PXA31x Only) 0b01 — Reserved (PXA30x Only) 0b10 — Reserved 0b11 — 260 MHz **NOTE:** The DMC gets two clocks from the clocks unit, one of which is half the frequency of the other. The slower one is sent to the DDR memory chip, while the faster is used by the controller itself. The frequencies controlled by this field refer to the faster clock. **NOTE:** There is a complex protocol for changing the value of this field. Refer to the Dynamic Memory Controller (DMC) chapter in the *PXA3xx Processor Family Processor Vol II: Memory Configuration Developers Manual* for details on changing the DDR SDRAM frequency,
11	R/W	PCCE	Power Mode Change Clock Enable: Enables the processor to use the ring oscillator output clock as a clock source when transitioning from any low-power mode to D0 mode. 0 = Do not use any clock until selected D0 mode clock is locked and stable. 1 = Use ring oscillator output as the processor clock source when transitioning from low-power mode to D0 mode.

Copyright © 2009 Marvell

Table 94: ACCR Bit Definitions (Continued)

	Physical Address 4134_0000																															
	ACCR																ACCU															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA31x Only	XPDIS	SPDIS	VAUF		Reserved	D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	0	0	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA30x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			Reserved			XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
10:8	R/W	XN	**Core PLL Turbo-Mode-to-Run-Mode Ratio:** Creates the nominal core PLL turbo-mode frequency by multiplying the run-mode frequency by XN (core PLL turbo-mode frequency is 13 MHz * XL * XN). 0x1 — Turbo-mode-to-run-mode ratio = 1:1 0x2 — Turbo-mode-to-run-mode ratio = 2:1 All other values reserved **NOTE:** The value of XN must always be less than or equal to the value of the MTS field in the “Application Subsystem Power Status/Configuration Register (ASCR)”.
7	R/W	DDR_D0CS	DDR SDRAM clock frequency selection in D0CS mode: (PXA32x and PXA31x Processors Only) 0 = 15 Mhz 1 = 30 Mhz
	—	—	Reserved (PXA30x Processor Only)
6:5	—	—	Reserved

Table 94: ACCR Bit Definitions (Continued)

	Physical Address 4134_0000																															
	ACCR																				ACCU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA31x Only	XPDIS	SPDIS	VAUF		Reserved	D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			DDR_D0CS	Reserved		XL				
Reset	0	0	0	0	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	0	?	?	0	1	0	0	0
PXA30x Only	XPDIS	SPDIS	Reserved			D0CS	SMCFS			Reserved			SFLFS		XSPCLK		HSS		DMCFS		PCCE	XN			Reserved			XL				
Reset	0	0	?	?	?	0	0	0	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
4:0	R/W	XL	Core PLL Run-Mode-to-Oscillator Ratio: Creates the nominal core PLL run-mode frequency by multiplying the processor (13 MHz) oscillator by XL (core PLL run-mode frequency is 13 MHz * XL). 0x08 — Run-mode-to-oscillator ratio = 8:1 (reset) 0xC — Run-mode-to-oscillator ratio = 12:1 0x10 — Run-mode-to-oscillator ratio = 16:1 0x18 — Run-mode-to-oscillator ratio = 24:1 0x1F— Run-mode-to-oscillator ratio = 31:1 (PXA32x Only) All other values reserved.

6.13.30 Application Subsystem Clock Status Register (ACSR)

The Application Subsystem Clock Status Register (ACSR), defined in Table 95, provides the current status of the processor clock system. This read-only register is loaded whenever the relevant events occur among the following: frequency change, exit from any power mode or core or system PLL status change. The bits map functionally to the corresponding bits in the Application Subsystem Clock Configuration Register (ACCR).

Copyright © 2009 Marvell

This is a read-only register. Ignore reads from reserved bits.

Table 95: ACSR Bit Definitions

Physical Address 4134_0004 **ACSR** **ACCU**

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	XPDIS_S	SPDIS_S	XPLCK	SPLCK	Reserved	RO_S	SMC_S			VAUF_S			SFL_S		XSPCLK_S		HSS_S		DMC_S		Reserved	XN_S			Reserved			XL_S				
Reset	0	0	1	1	?	0	0	0	0	0	0	?	0	0	1	1	0	0	0	0	?	0	0	1	?	?	?	0	1	0	0	0
PXA32x and PXA30x Only	XPDIS_S	SPDIS_S	XPLCK	SPLCK	Reserved	RO_S	SMC_S			Reserved			SFL_S		XSPCLK_S		HSS_S		DMC_S		Reserved	XN_S			Reserved			XL_S				
Reset	0	0	1	1	?	0	0	0	0	?	?	?	0	0	1	1	0	0	0	0	?	0	0	1	?	?	?	0	1	0	0	0

Bits		Name	Description
31	R	XPDIS_S	Core PLL Output Disable (XPDIS) Status 0 = Core PLL is enabled and output can be used. 1 = Core PLL is not enabled for use.
30	R	SPDIS_S	System PLL Output Disable (SPDIS) Status 0 = System PLL is enabled and output can be used. 1 = System PLL is not enabled for use.
29	R	XPLCK	Core PLL Lock 0 = Core PLL is not locked or not enabled for use. 1 = Core PLL is locked and ready to use.
28	R	SPLCK	System PLL Lock: 0 = System PLL is not locked or not enabled for use. 1 = System PLL is locked and ready to use.
27	—	—	Reserved
26	R	RO_S	Ring Oscillator Status 0 = Processor clock is currently not using the ring oscillator output. 1 = Processor clock is currently using the ring oscillator output for its clock.
25:23	R	SMC_S	Static Memory Controller Frequency Selected 0b000 — 78 MHz 0b010 — 104 MHz 0b101 — 208 MHz **All other values are reserved**

Table 95: ACSR Bit Definitions (Continued)

	Physical Address 4134_0004																															
	ACSR																				ACCU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	XPDIS_S	SPDIS_S	XPLCK	SPLCK	Reserved	RO_S	SMC_S			VAUF_S			SFL_S		XSPCLK_S		HSS_S		DMC_S		Reserved	XN_S			Reserved			XL_S				
Reset	0	0	1	1	?	0	0	0	0	0	0	?	0	0	1	1	0	0	0	0	?	0	0	1	?	?	?	0	1	0	0	0
PXA32x and PXA30x Only	XPDIS_S	SPDIS_S	XPLCK	SPLCK	Reserved	RO_S	SMC_S			Reserved			SFL_S		XSPCLK_S		HSS_S		DMC_S		Reserved	XN_S			Reserved			XL_S				
Reset	0	0	1	1	?	0	0	0	0	?	?	?	0	0	1	1	0	0	0	0	?	0	0	1	?	?	?	0	1	0	0	0

Bits		Name	Description
22:21	R	VAUF_S	Selected Video Accelerator Unit Frequency 0b00 -- 104 MHz 0b01 -- 156MHz 0b10 -- 208 MHz 0b11 -- 78MHz **NOTE:** These bit fields are only available in the PXA31x Processor.
	—	—	Reserved (PXA32x and PXA30x Only)
20	—	—	reserved
19:18	R	SFL_S	Frequency for the Internal Memory Controller (SRAM) 0b00 — 104 MHz 0b01 — 156 MHz 0b10 — 208 MHz 0b11 — 312 MHz
17:16	R	XSPCLK_S	Frequency Change Clock Select Status The frequency the core is using during frequency changes. Clock source is provided from the System PLL. 0b00 — 156 MHz 0b11 — No clocks being used. All other values are reserved
15:14	R	HSS_S	System Bus Frequency Select Status: 0b00 — 104 MHz 0b01 —156 MHz 0b10 — 208 MHz 0b11 — 312 MHz
13:12	R	DMC_S	Dynamic Memory Controller SDRAM Clock Frequency status 0b00 — 15 MHz (Reset only) 0b01 — reserved 0b10 — reserved 0b11 — 130 MHz
11	—	—	Reserved

Copyright © 2009 Marvell

Table 95: ACSR Bit Definitions (Continued)

	Physical Address 4134_0004																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA31x Only	XPDIS_S	SPDIS_S	XPLCK	SPLCK	Reserved	RO_S	SMC_S			VAUF_S			SFL_S		XSPCLK_S		HSS_S		DMC_S		Reserved	XN_S			Reserved			XL_S				
Reset	0	0	1	1	?	0	0	0	0	0	0	?	0	0	1	1	0	0	0	0	?	0	0	1	?	?	?	0	1	0	0	0
PXA32x and PXA30x Only	XPDIS_S	SPDIS_S	XPLCK	SPLCK	Reserved	RO_S	SMC_S			Reserved			SFL_S		XSPCLK_S		HSS_S		DMC_S		Reserved	XN_S			Reserved			XL_S				
Reset	0	0	1	1	?	0	0	0	0	?	?	?	0	0	1	1	0	0	0	0	?	0	0	1	?	?	?	0	1	0	0	0

Bits		Name	Description
10:8	R	XN_S	**Turbo-Mode-to-Run-Mode Ratio (XN) Status**
7:5	—	—	Reserved
4:0	R	XL_S	Run-Mode-to-Processor (13 MHz) Oscillator Ratio (XL) Status

6.13.31 Application Subsystem Interrupt Control/Status Register (AICSR)

The Application Subsystem Interrupt Control/Status Register (AICSR) is used to enable/disable the frequency-change interrupt and to provide status to indicate if the core clocks are using the core PLL clocks following a frequency-change operation. Writing a 1 to any of the status bits resets them to 0.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 96: AICSR Bit Definitions (Continued)

	Physical Address 4134_0008																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										PCIS	PCIE	Reserved		FCIS	FCIE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:6	—	—	Reserved

Table 96: AICSR Bit Definitions (Continued)

	Physical Address 4134_0008												AICSR								ACCU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										PCIS	PCIE	Reserved		FCIS	FCIE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
5	R/ Write 1 to Clear	PCIS	Power-Mode Change-Induced Frequency Change Interrupt Status 0 = The core clock is not ready to be switched over to core PLL following a power-mode change, or AICSR[PCIE] is not set. 1 = The core clock is ready to be switched over to the core PLL following a power-mode change, and AICSR[PCIE] is set. **NOTE:** Only set when ACCR[PCCE] is set.
4	R/W	PCIE	Power-Mode Change-Induced Frequency Change Interrupt Enable 0 = Do not send an interrupt after a power mode change sequence. 1 = Send an interrupt when the core clock is ready to be switched over to the core PLL clock output after a power-mode change.
3:2	—	—	Reserved
1	R/ Write 1 to Clear	FCIS	Frequency Change Interrupt Status 0 = The core clock has not switched to using the core PLL following a software requested frequency change, or AICSR[FCIE] is not set. 1 = The core clock has switched to using the core PLL clocks following a software requested frequency change, and AICSR[FCIE] is set.
0	R/W	FCIE	Frequency Change Interrupt Enable 0 = Do not send an interrupt after a software requested frequency change sequence. 1 = Send an interrupt when the core clock switch back to the core PLL clock output after a software requested frequency change.

6.13.32 D0 Mode Clock Enable Register A (D0CKEN_A)

The D0 Mode Clock Enable Register A (D0CKEN_A) enables or disables the clock to most of the processor peripherals when the processor is in D0 mode. For lowest power consumption, any unit that is not being used must have its clock disabled by clearing the appropriate bit. The CKEN bits control only the functional clock inputs to each module. The bus clock typically remains unaffected, unless the cluster is totally turned *off*.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 97: D0CKEN_A Bit Definitions: Clock Enable Mappings for Units

	Physical Address 4134_000C												D0CKEN_A														ACCU							
User Settings																																		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	CKEN25	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA31x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	RESERVED	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	CKEN5	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA30x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
Reset	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	?	1	1	1	1	1	?

Bits	Access	Name	Description
31	—	—	SETALWAYS - Must be programmed with 0b1
30	—	—	Reserved
29	R/W	CKEN29	SSP4 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
28	R/W	CKEN28	SSP3 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
27	R/W	CKEN27	SSP2 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
26	R/W	CKEN26	SSP1 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
25	R/W	CKEN25	Touchscreen Interface Clock Enable (PXA32x only) 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
	—	—	Reserved (PXA31x and PXA30x only)
24	R/W	CKEN24	AC97 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
23	R/W	CKEN23	UART3 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.

Table 97: D0CKEN_A Bit Definitions: Clock Enable Mappings for Units (Continued)

Physical Address 4134_000C — D0CKEN_A — ACCU

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	CKEN25	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA31x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	RESERVED	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	CKEN5	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA30x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
Reset	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	?	1	1	1	1	1	?

Bits	Access	Name	Description
22	R/W	CKEN22	UART1 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
21	R/W	CKEN21	UART2 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
20	R/W	CKEN20	UDC Clock Enable (PXA32x and PXA30x only) 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
			Reserved (PXA31x only)
19	R/W	CKEN19	WTM Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
18	R/W	CKEN18	USIM[1] Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
17	R/W	CKEN17	USIM[0] Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
16	—	—	Reserved
15	R/W	CKEN15	Consumer IR (CIR) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
14	R/W	CKEN14	Keypad Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.

Copyright © 2009 Marvell

Table 97: D0CKEN_A Bit Definitions: Clock Enable Mappings for Units (Continued)

	Physical Address 4134_000C												D0CKEN_A														ACCU							
User Settings																																		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	CKEN25	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA31x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	RESERVED	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	CKEN5	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA30x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
Reset	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	?	1	1	1	1	1	?

Bits	Access	Name	Description
13	R/W	CKEN13	MMC1 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
12	R/W	CKEN12	MMC0 Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
11	R/W	CKEN11	Boot ROM Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
10	R/W	CKEN10	Internal Memory Controller (IM) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
9	R/W	CKEN9	Static Memory Controller (SMC) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
8	R/W	CKEN8	Dynamic Memory Controller (DMC) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
7	R/W	CKEN7	Graphic Controller Unit Clock Enable (PXA32x Only) 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
	—	—	SETALWAYS - Must be programmed with 0b1 (PXA31x and PXA30x Only)
6	R/W	—	USB 2.0 Device Controller (U2DC) Clock Enable 0 = Clock to the unit is disabled 1 = Clock to the unit is enabled

Table 97: D0CKEN_A Bit Definitions: Clock Enable Mappings for Units (Continued)

Physical Address 4134_000C — D0CKEN_A — ACCU

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	CKEN25	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA31x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	RESERVED	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	CKEN5	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
PXA30x Only	SETALWAYS	RESERVED	CKEN29	CKEN28	CKEN27	CKEN26	RESERVED	CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKEN17	RESERVED	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	SETALWAYS	CKEN6	SETALWAYS	CKEN4	CKEN3	CKEN2	CKEN1	SETALWAYS
Reset	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	?	1	1	1	1	1	?

Bits	Access	Name	Description
5	R/W	CKEN5	MMC3 Clock Enable (PXA31x Only) 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
	—	—	SETALWAYS - Must be programmed with 0b1 (PXA32x and PXA30x Only)
4	R/W	CKEN4	NAND Flash Controller (NFC) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
3	R/W	CKEN3	Camera Interface (CI) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
2	R/W	CKEN2	USB Host Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
1	R/W	CKEN1	LCD Controller Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
0	—	—	SETALWAYS - Must be programmed with 0b1

Note

D0CKEN_A[CKEN10] for the internal memory controller clock and D0CKEN_A[CKEN11] for the boot ROM must be set simultaneously. Clearing D0CKEN_A[CKEN1] doesn't guarantee to shut off LCD clock. You also need to Clear the LCD Controller Enable bit (ENB) in the LCD Controller Control Register 0 (LCCR0[ENB) bit to shut off LCD clock.

Copyright © 2009 Marvell

6.13.33 D0 Mode Clock Enable Register B (D0CKEN_B)

Since there are more than 32 peripherals in the subsystem, the overflow bits are allocated in the D0 Mode Clock Enable Register B (D0CKEN_B). The functionality is absolutely identical to its A counterpart.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 98: D0CKEN_B Bit Definitions: Clock Enable Mappings for Units

Physical Address 4134_0010																															
D0CKEN_B																															
ACCU																															
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
PXA31x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
PXA30x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN10	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	?	?	?	?	0	1	1	1	1	?	?	1	?	?	1	1

Bits	Access	Name	Description
31	—	—	SETALWAYS - Must be programmed with 0b1
30	—	—	SETALWAYS - Must be programmed with 0b1
29	—	—	SETALWAYS - Must be programmed with 0b1
28	—	—	SETALWAYS - Must be programmed with 0b1
27	—	—	SETALWAYS - Must be programmed with 0b1
26	—	—	SETALWAYS - Must be programmed with 0b1
25	—	—	SETALWAYS - Must be programmed with 0b1
24	—	—	SETALWAYS - Must be programmed with 0b1.
23	—	—	SETALWAYS - Must be programmed with 0b1.
22	—	—	SETALWAYS - Must be programmed with 0b1
21	—	—	SETALWAYS - Must be programmed with 0b1
20	—	—	SETALWAYS - Must be programmed with 0b1
19	—	—	SETALWAYS - Must be programmed with 0b1
18	—	—	SETALWAYS - Must be programmed with 0b1

Copyright © 2009 Marvell

Table 98: D0CKEN_B Bit Definitions: Clock Enable Mappings for Units (Continued)

Physical Address 4134_0010 — D0CKEN_B — ACCU

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
PXA31x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
PXA30x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN10	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	?	?	?	?	0	1	1	1	1	?	?	1	?	?	1	1

Bits	Access	Name	Description
17	R/W	CKEN17	Mini LCD Controller Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
16	R/W	CKEN16	Mini-Internal Memory Controller Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
15	—	—	SETALWAYS - Must be programmed with 0b1
14	—	—	SETALWAYS - Must be programmed with 0b1
13	—	—	SETALWAYS - Must be programmed with 0b1
12	—	—	SETALWAYS - Must be programmed with 0b1
11	R/W	CKEN11	Video Accelerator Unit Clock Enable (PXA31x Only) 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
			SETALWAYS - Must be programmed with 0b1. (PXA32x and PXA30x Only)
10	R/W	CKEN10	Graphic Controller Unit Clock Enable (PXA31x and PXA30x Only) 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
	—	—	SETALWAYS - Must be programmed with 0b1 (PXA32x Only)
9	R/W	CKEN9	System Bus (2) Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled. **NOTE:** Must be set before any lower power mode or D0CS entry.
8	R/W	CKEN8	1-Wire Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled

Copyright © 2009 Marvell

Table 98: D0CKEN_B Bit Definitions: Clock Enable Mappings for Units (Continued)

	Physical Address 4134_0010												D0CKEN_B														ACCU						
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
PXA31x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
PXA30x Only	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN17	CKEN16	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	SETALWAYS	CKEN10	CKEN9	CKEN8	CKEN7	SETALWAYS	RESERVED	CKEN4	SETALWAYS	SETALWAYS	CKEN1	CKEN0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	?	?	?	?	0	1	1	1	1	?	?	1	?	?	1	1

Bits	Access	Name	Description
7	R/W	CKEN7	GPIO Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled
6	—	—	SETALWAYS - Must be programmed with 0b1
5	—	—	Reserved
4	R/W	CKEN4	Standard I^2C Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
3	—	—	SETALWAYS - Must be programmed with 0b1
2	—	—	SETALWAYS - Must be programmed with 0b1
1	R/W	CKEN1	PWM[3] and PWM[1] Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled.
0	R/W	CKEN0	PWM[2] and PWM[0] Clock Enable 0 = Clock to the unit is disabled. Clock to the unit is enabled.

6.13.34 AC ’97 Clock Divisor Value Register (AC97_DIV)

The AC ’97 unit uses a dithered divider to generate the frequency needed by that unit. Essentially a 15-bit numerator value and a 12-bit denominator value comprise the divisor that is used to divide down the 312-MHz system PLL clock to yield the unit clock. The reset value of the divider is 1625/128**.** Refer to the AC ’97 chapter for details.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 100: AC97_DIV Bit Definitions

	Physical Address 4134_0014																															
	AC97_DIV																				ACCU											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					Numerator															Denominator											
Reset	?	?	?	?	?	0	0	0	0	1	1	0	0	1	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:27	—	—	Reserved
26:12	R/W	Num	Numerator value in the dithering divider used for generating clk_ac97
11:0	R/W	Denom	Denominator value in the dithering divider used for generating clk_ac97

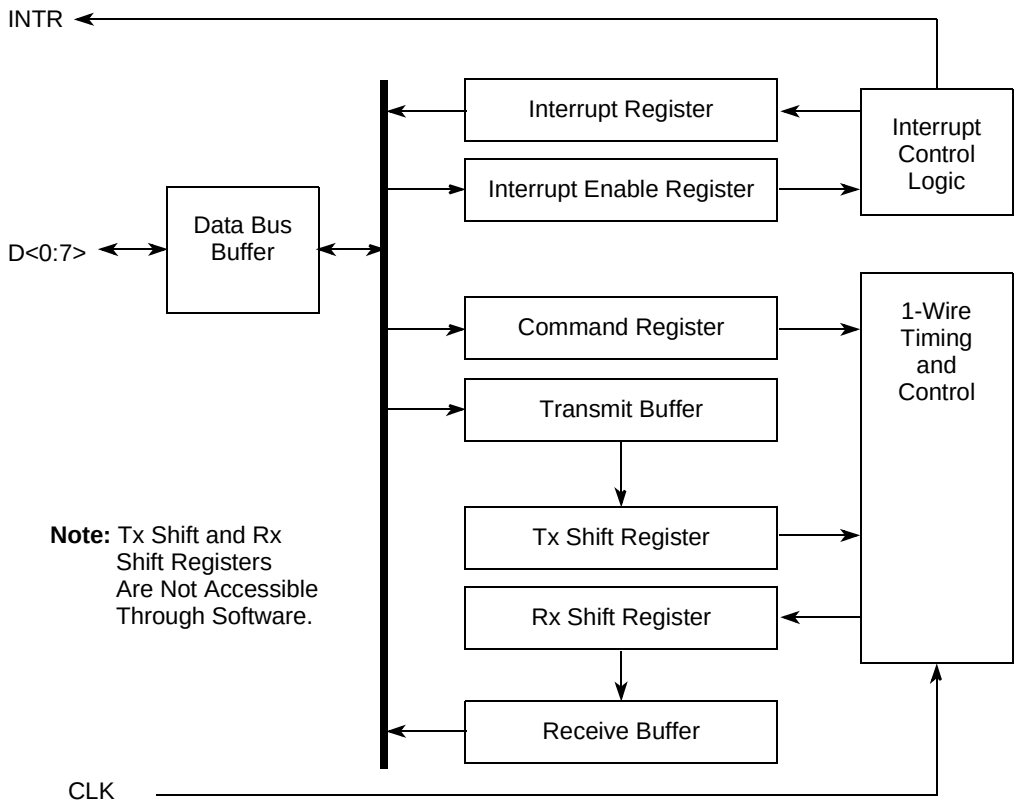
Copyright © 2009 Marvell

7 1-Wire Bus Master Interface

This chapter describes how the processor works with the 1-Wire* bus master interface controller and the related processor-supported registers.

The 1-Wire bus master interface controller receives and transmits 1-Wire bus data and completely controls the 1-Wire bus through eight-bit commands. The processor loads commands, reads and writes data, and sets interrupt control through five registers. All 1-Wire bus timing and control are generated within the 1-Wire bus master interface controller after the host loads a command or data. When bus activity generates a response that the CPU needs to receive, the 1-Wire bus master interface controller sets a status bit and, if enabled, generates an interrupt to the CPU. The operation of the 1-Wire bus is described in detail in the *Book of iButton* Standards*[1]. Refer to this document for details on specific slave implementations. Figure 12 is a block diagram of the 1-Wire bus master.

Figure 12: 1-Wire Bus Master Block Diagram



1. For more information about the *Book of iButton Standards* see http://www.maxim-ic.com/appnotes.cfm/appnote_number/937.

7.1 Signals

Table 97 describes the functionality of the 1-Wire signal.

Table 97: 1-Wire Signal

Name	Direction	Description
ONE_WIRE	In/Out	1-Wire Data Line This open-drain line is the 1-Wire bidirectional data bus signal. 1-Wire slave devices are connected to this pin. This pin must be pulled high by an external resistor, nominally 5 KΩ. 1-Wire spec allows 1KΩ to 5KΩ resistor.

7.2 Operation

This section describes the procedure for reading and writing on the 1-Wire interface. The 1-Wire bus master interface controller is located on Peripheral Bus 2. The 1-Wire protocol requires a reset before any bus communication. Generating a 1-Wire reset on the bus is discussed in Section 7.2.3.1.

The 1-Wire bus master interface controller generates read, write, or reset commands on the 1-Wire bus. The 1-Wire reset is a special command that must precede the command given on the bus.

7.2.1 Writing a Byte

To send a byte on the 1-Wire bus, write the byte to be transferred to the Transmit bufferReceive buffer. The data is then moved to the Tx Shift register (see note in Figure 12) where it is shifted serially onto the bus, LSB first. A new byte of data can then be written to the Transmit bufferTransmit buffer. As soon as the Tx Shift register is empty, the new data is transferred from the Transmit buffer and the process repeats. Each register has a flag for use as an interrupt source. The Transmit buffer empty flag, W1INTR[TBE], is set when the Transmit buffer is empty and ready to accept a new byte. W1INTR[TBE] is cleared as soon as a byte is written into the Transmit buffer. The Tx Shift register empty flag, W1INTR[TEMT], is set when the Tx Shift register has no data in it and is ready to accept a new byte. As soon as a byte of data is transferred from the Transmit buffer, W1INTR[TEMT] is cleared and W1INTR[TBE] is set.

7.2.2 Reading a Byte

The device must be prepared to transmit data before data can be read from a slave device. The slave device is prepared through commands previously received from the CPU. Data is retrieved from the bus in a similar way as a write operation. The host initiates a read by writing to the Transmit buffer. The data that is then shifted into the Rx Shift register (see note in Figure 12) is the wired-AND of the written data and the data from the slave device. Therefore, the host must write 0xFF to read a byte from a slave device. When the Rx Shift register is full, the data is transferred to the Receive buffer where the host can access it.

Additional bytes can now be read by sending 0xFF again. If the slave device is not ready to transmit, the data received is identical to that transmitted. The 1-Wire Transmit/Receive Buffer (W1TRR) register can also generate interrupts. The Receive buffer flag, W1INTR[RBF], is set when data is transferred from the Rx Shift register and cleared when the host reads the Rx Shift register. If RBF is set, there must be no additional transmissions on the 1-Wire bus because the byte in the Receive buffer is overwritten by the next received byte, resulting in lost data. Interrupt flags are explained in detail in Section 7.4.3, 1-Wire Interrupt Register (W1INTR). Write and read operations are detailed in Figure 14 and Figure 15.

Copyright © 2009 Marvell

7.2.3 I/O Signaling

The 1-Wire bus requires strict signaling protocols to ensure data integrity. The four protocols used by the 1-Wire bus master interface controller are as follows:

- Initialization sequence (reset pulse followed by presence pulse)
- Write 0
- Write 1
- Read data

The master initiates all of these types of signaling except the presence pulse.

7.2.3.1 Initialization Sequence

Figure 13 shows the initialization sequence required to begin any communication with the bus slave.

1. The 1-Wire bus master controller transmits a reset pulse for 500 cycles tRSTL.
2. The 1-Wire bus line is then pulled high by the external pullup resistor. Marvell recommends this pullup resistor have less than 5k Ω and the tSLEW timing specification of 1 cycle must be met for this initialization process to work correctly, otherwise it may result in a small presence pulse width that can cause functional failure.

Note

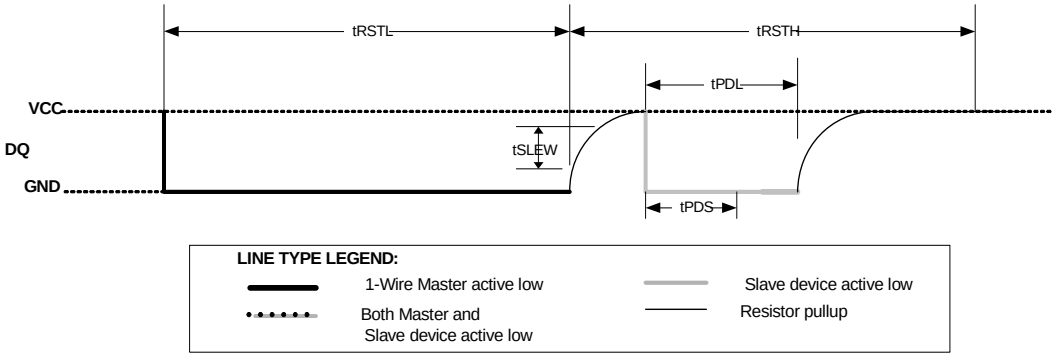
tSLEW timing must meet the 1 cycle specification. VxL and VxH specifications are detailed in the EMTS which details the voltage range for logic 0 and logic 1. The 1-Wire specification allows a 1K-5K pull up resistor range, which allows the bus to be tuned to meet timing specifications.

3. After detecting the rising edge on the ONE_WIRE pin, the 1-Wire master waits for the 1-Wire slave device to pull the bus low or a period of 60 cycles to expire. (NOTE: This is an OR statement).
4. The master waits an additional 30 cycles then samples the bus during the presence pulse after the slave responds to test for a valid-presence pulse (tPDS). The result of this sample is stored in the PDR bit of the 1-Wire Interrupt register, W1INTR[PDR]. PDR = 0b0 means slave present and 0b1 means slave not present.
5. The reset time slot ends a specified amount of time after the master releases the bus.

Note

The timings are based on cycles which are configured in Table 103, W1CDR Bit Definitions, on page 291

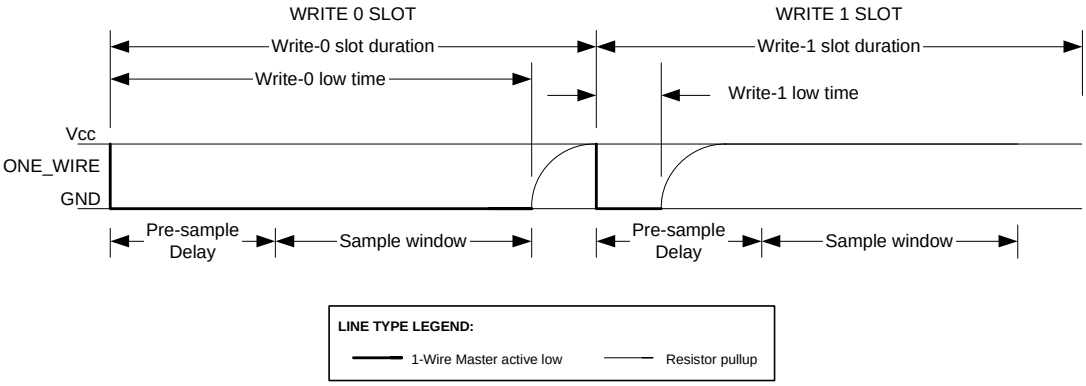
Figure 13: 1-Wire Initialization Sequence (Reset and Presence Pulses)



7.2.3.2 Write Time Slots

A Write time slot is initiated when the 1-Wire bus master drives the 1-Wire bus line from a logic-high (inactive) level to a logic-low level. The 1-Wire bus master generates a Write-1 time slot by releasing the bus line after the required amount of logic-low time for a Write-1 allowing the bus line to pull up to a logic-high level. The 1-Wire bus master generates a Write-0 time slot by holding the bus line low for the required amount of logic-low time. A slave device then samples the bus line a specified amount of time after the bus line transitions from a logic-high to a logic-low state. A Write-1 occurs if the bus line is logic-high when sampled; a Write-0 occurs if the bus line is logic-low when sampled (see Figure 14). Refer to the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification* for detailed timing information.

Figure 14: 1-Wire Write Slots



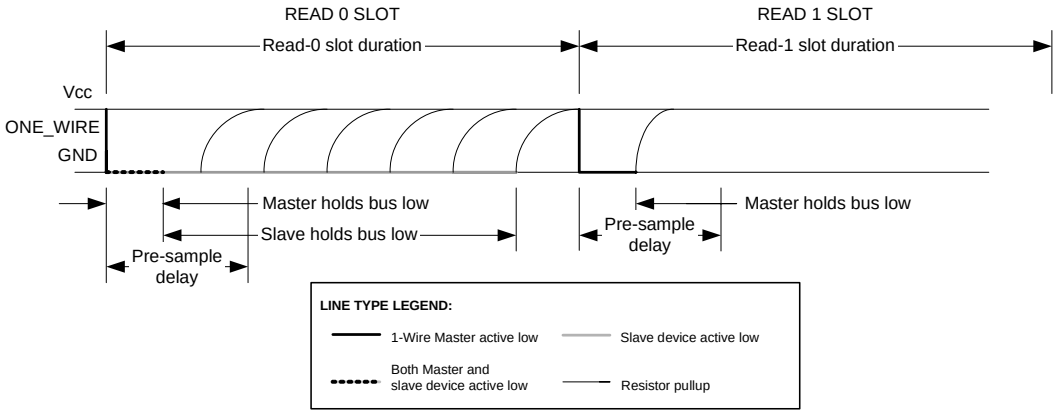
7.2.3.3 Read Time Slots

A Read time slot is initiated when the 1-Wire bus master pulls the bus line low for the minimum required time and then releases it. If the slave device responds with a 0, the slave continues to hold the bus line low for up to a specified amount of time before the slave releases the bus line; the bus line is then pulled high by the external pullup resistor. If the slave device responds with a 1, the slave does not hold the bus line low and the bus line is pulled to a logic-high immediately after the master releases the line. The master samples the data after a specified amount of time after the start of the read time slot. The master ends the read slot after the required amount of time (see Figure 15).

Copyright © 2009 Marvell

Refer to the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification* for detailed timing information.

Figure 15: 1-Wire Read Time Slots



7.3 Interrupt Handling

Three types of interrupts are programmed: the *Receive buffer flag (RBF) interrupt,* the *transmit shift register empty flag (TEMPT) interrupt*, and the *Transmit buffer empty (TBE) interrupt*. This section describes how to program these types of interrupts.

The typical programming flow for an RBF interrupt proceeds as follows:

1. Enable the preferred interrupts and set W1IER[ERBF] and W1IER[EPD].
2. Assert the INTR signal active high. Set W1IER[IAS].
3. Reset the bus. Set W1CMDR[1WR].
4. Unmask interrupts in the ICU.
5. When the PD interrupt occurs, in the service routine, mask interrupts in the ICU, clear the W1IER[EPD] bit, and read the W1INTR[PDR] and W1INTR[PD] bits.

   Ensure that these are correct (that is, PDR = 0 if there is a slave and PD = 1). Do not clear or write over the W1IER[IAS] bit. Leave the interrupt service routine.
6. Send a command. Write to the xmit buffer: W1TRR[DATA]. Unmask interrupts.

   The Receive buffer is filled with the contents written to the xmit buffer.
7. Wait for the RBF interrupt. In a service routine, mask interrupts, and clear the W1IER[ERBF] bit.
8. Read the W1INTR[RBF] bit and ensure that it is set.
9. Read the Receive buffer W1TRR[DATA] to verify that the data is sent.

The typical programming flow for a TEMPT interrupt proceeds as follows:

1. Enable the preferred interrupts. Set W1IER[ETMT] and W1IER[EPD].
2. Set the INTR signal active high. Set W1IER[IAS].
3. Reset the bus. Set W1CMDR[1WR].
4. Unmask interrupts in the ICU.
5. When the PD interrupt occurs, in the service routine, mask interrupts in the ICU, clear the W1IER[EPD] bit, and read the W1INTR[PDR] and W1INTR[PD] bits.

   Ensure that these are correct (that is, PDR = 0 if there is a slave and PD = 1). Do not clear or write over the W1IER[IAS] bit. Leave interrupt service routine.

6. Send data. Write to the xmit buffer: W1TRR[DATA]. Unmask interrupts.
7. Wait for TEMT interrupt. In the service routine, mask interrupts, and clear the W1IER[ETMT] bit.
8. Read the W1INTR[TEMT] bit and ensure that it is set.

The typical programming flow for the TBE interrupt proceeds as follows:

1. Enable the preferred interrupts. Set W1IER[ETBE] and W1IER[EPD].
2. Assert the INTR signal active high. Set W1IER[IAS].
3. Reset the bus. Set W1CMDR[1WR].
4. Unmask interrupts in the ICU.
5. When the PD interrupt occurs in the service routine, mask interrupts in the ICU, clear the W1IER[EPD] bit, and read the W1INTR[PDR] and W1INTR[PD] bits.

   Ensure that these are correct (that is, PDR = 0 if there is a slave and PD = 1). Do not clear or write over the W1IER[IAS] bit. Unmask the interrupts. Leave the interrupt service routine.
6. Send data. Write to the xmit buffer: W1TRR[DATA].
7. Wait for TBE interrupt. In the service routine, mask interrupts, clear the W1IER[ETBE] bit. Read the W1INTR[TBE] bit and ensure that it is set.

   New data, if needed, can be written into the buffer and interrupts should be unmasked.

7.4 Register Descriptions

Five 32-bit user-accessible registers reside within the 1-Wire bus master interface controller: one Command register, one Data register (transmit/Receive buffer), two Interrupt (status and enable) registers, and one Clock Divisor register.

- The Command register controls the 1-Wire bus master interface controller functionality.
- The Data register is a bidirectional register to transmit and receive data from the 1-Wire bus.
- The Interrupt registers hold the controller status and enable interrupts.
- The Clock Divisor register must be configured before 1-Wire communication can occur.

Table 98 summarizes the 1-Wire bus master interface controller register allocations in the memory map. The remainder of this section discusses each of these registers in detail. For easy reference, the summary tables include the page number of the detailed description for each register.

Table 98: 1-Wire Bus Register Summary

Address	Name	Description	Page
0x41B0_0000	W1CMDR	1-Wire Command register	page 286
0x41B0_0004	W1TRR	1-Wire Transmit/Receive buffer	page 288
0x41B0_0008	W1INTR	1-Wire Interrupt register	page 288
0x41B0_000C	W1IER	1-Wire Interrupt Enable register	page 289
0x41B0_0010	W1CDR	1-Wire Clock Divisor register	page 290
0x41B0_0014 – 0x41BF_FFFF	Reserved	—	—

7.4.1 1-Wire Command Register (W1CMDR)

W1CMDR, shown in Table 99, contains four valid bit fields that control the 1-Wire bus master controller functionality. In addition, it contains two bits to bypass the 1-Wire bus master interface controller features and control the 1-Wire bus directly.

Copyright © 2009 Marvell

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 99: W1CMDR Bit Definitions

Physical Address 0x41B0_0000																																
W1CMDR																																
1-Wire Interface																																
User Settings																																
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Reserved																												**DQI**	**DQO**	**SRA**	**1WR**
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	0	0

Bits	Access	Name	Description
31:4	—	Reserved	Reserved
3	R	DQI	ONE_WIRE Input Reflects the present state of the 1-Wire bus. Use it together with the W1CMDR[DQO] bit when controlling the bus directly. The state of this bit does not affect any other functions of the 1-Wire bus master interface controller. Operation of this bit is unaffected by the state of the W1IER [DQOE] bit.
2	W	DQO	ONE_WIRE Output Bypasses the 1-Wire bus master interface controller operations and drives the bus directly, if needed. By controlling the length of time this bit is set and the point when the line is sampled (see W1CMDR[DQI]), the host controller can generate any 1-Wire communication. To prevent accidental writes to the bus, set the W1IER [DQOE] bit in the Interrupt Enable register before the DQO bit functions. 0 = Cleared on power-up reset. Clearing this bit drives the bus high. 1-Wire bus master interface controller operations function only while the 1-Wire bus is held high. 1 = Drives the bus low until it is cleared or the 1-Wire bus master interface controller is reset. While the 1-Wire bus is held low, no other 1-Wire bus master interface controller operations function.
1	R/W	SRA	Search ROM Accelerator 0 = SRA turned off 1 = 1-Wire bus master interface controller switches to SRA mode. See the *Book of iButton Standards* for more information.
0	R/W	1WR	1-Wire Reset Generates a reset on the 1-Wire bus. The 1-Wire bus master interface controller sets the presence detect interrupt flag, W1INTR[PD], when the reset is complete and sufficient time for a presence detect to occur has passed. The result of the presence detect is placed in the Interrupt register bit, W1INTR[PDR]. 0 = Bus is not in reset mode. 1 = Setting this bit automatically clears the SRA bit. The 1WR bit is cleared automatically as soon as the 1-Wire reset completes.

Copyright © 2009 Marvell

7.4.2 1-Wire Transmit/Receive Buffer (W1TRR)

W1TRR, shown in Table 100, provides the location through which data sent and received from the 1-Wire bus master interface controller passes through the transmit/Receive buffer. The 1-Wire bus master interface controller is double-buffered with separate transmit and Receive buffers. Writing to this location connects the Transmit buffer to the data bus, and reading connects the Receive buffer to the data bus.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 100: W1TRR Bit Definitions

Physical Address 0x41B0_0004	W1TRR	1-Wire Interface

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								DATA							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:8	—	Reserved	Reserved
7:0	R/W	Data	Transmitted/Received Data

7.4.3 1-Wire Interrupt Register (W1INTR)

W1INTR, shown in Table 101, contains flags from transmit, receive, and 1-Wire reset operations. Only the presence-detect flag (PD) is cleared when the W1INTR is read; the other flags are automatically cleared when the transmit and Receive buffers are written to or read, respectively. These flags can generate an interrupt if the corresponding enable bit is set in W1IER.

This is a read-only register. Ignore reads from reserved bits.

Table 101: W1INTR Bit Definitions

Physical Address 0x41B0_0008	W1INTR	1-Wire Interface

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																											RBF	TEMT	TBE	PDR	PD
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	0

Bits	Access	Name	Description
31:5	—	Reserved	Reserved
4	R	RBF	Receive Buffer Full 0 = This flag is cleared when the byte is read from the Receive Buffer register. 1 = This flag is set when there is a byte waiting to be read in the Receive buffer.

Copyright © 2009 Marvell

Table 101: W1INTR Bit Definitions (Continued)

Physical Address 0x41B0_0008																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																											RBF	TEMT	TBE	PDR	PD
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	0

W1INTR			1-Wire Interface
Bits	**Access**	**Name**	**Description**
3	R	TEMT	Tx Shift Register Empty 0 = This flag is cleared when data is shifted into the Tx Shift register from the Transmit buffer. 1 = This flag is set after the last bit is transmitted on the 1-Wire bus. **NOTE:** TEMT status bit is valid upon completion of the first data transfer.
2	R	TBE	Transmit Buffer Empty 0 = This flag is cleared when data is written to the Transmit buffer. 1 = This flag is set when the last bit is transferred to the Tx Shift register. **NOTE:** TBE status bit is valid after the first data transfer.
1	R	PDR	Presence Detect Result When a Presence Detect interrupt occurs, this bit reflects the result of the presence detect read. 0 = PDR is 0 if a slave device is found. 1 = No slave device found.
0	R	PD	Presence Detect 0 = The required time after a 1-Wire reset has not elapsed or W1INTR was read since the last 1-Wire reset. 1 = When a 1-Wire reset is issued, this flag is set after the appropriate amount of time for a presence detect pulse to occur. This bit is cleared when the Interrupt register is read.

7.4.4 1-Wire Interrupt Enable Register (W1IER)

W1IER, shown in Table 102, allows system programmers to specify which interrupt source causes an interrupt. When a reset is received, all non-reserved bits in this register are cleared, disabling all interrupt sources.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 102: W1IER Bit Definitions

Physical Address 0x41B0_000C																																
W1IER																																
1-Wire Interface																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								DQOE	Reserved		ERBF	ETMT	ETBE	Reserved	EPD
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0	0	0	1	0

Bits	Access	Name	Description
31:8	—	Reserved	Reserved
7	R/W	DQOE	ONE_WIRE Output Enable A control select for the ONE_WIRE bus. When it is cleared, the bus is controlled by the 1-Wire bus master interface controller as normal. 0 = Defaults to 0 on power-up or reset and should be left 0 unless users want to control the bus manually through the W1CMDR[DQO] bit. 1 = The W1CMDR[DQO] bit controls the state of the bus directly.
6:5	—	Reserved	Reserved
4	R/W	ERBF	Enable Receive Buffer Full Interrupt 0 = Interrupt disabled. 1 = If the Receive buffer full flag is set, an interrupt is generated.
3	R/W	ETMT	Enable Tx Shift Register Empty Interrupt 0 = Interrupt disabled. 1 = If the Tx Shift register empty flag is set, an interrupt is generated.
2	R/W	ETBE	Enable Transmit Buffer Empty Interrupt 0 = Interrupt disabled. 1 = If the Transmit buffer empty flag is set, an interrupt is generated.
1	—	Reserved	Reserved **NOTE:** This bit must always be set.
0	R/W	EPD	Enable Presence Detect Interrupt For timing information, refer to the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification*. 0 = Interrupt disabled. 1 = If the enable presence detect flag is set, an interrupt is generated when a 1-Wire reset is sent and the required amount of time passes for a presence detect pulse to occur.

7.4.5 1-Wire Clock Divisor Register (W1CDR)

W1CDR, shown in Table 103, divides the internal reference clock to generate the 1-Wire clock.This register must be programmed before the 1-Wire bus master interface is used.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 103: W1CDR Bit Definitions

	Physical Address 0x41B0_0010												W1CDR												1-Wire Interface							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																											DIV			PRE	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:5	—	Reserved	Reserved
4:2	R/W	DIV	Divider The 1-Wire bus master interface controller uses the output of the prescaler and divides by the DIV value to produce the 1-Wire clocks. The 1-Wire clock must be approximately 1 MHz for correct operation. The DIV field value raised to the power of 2 is used to divide the clock generated by the prescaler. The DIV value must be set to 0b10
1:0	R/W	PRE	Prescaler The 1-Wire bus master interface controller uses the 26 MHz as the input clock. This 26 MHz clock is initially divided by the prescaler. The output of the prescaler provides the input clock to the Divider. The prescaler is configured as follows: PRE = 11 = divide by 7 PRE = 10 = divide by 5 PRE = 01 = divide by 3 PRE = 00 = divide by 7 The PRE value must be set to 0b11

NOTE: Program this register with 0x0B (0b01011). The internal reference clock frequency is 26 MHz and this setting performs a prescale of 7 and a final divide of 4.
0x0B calculation:
26 MHz / 7 = 3.714 MHz
3.714 MHz / 4 = 0.928 MHz

Copyright © 2009 Marvell

8 DMA Controller

The direct-memory access controller (DMAC) transfers data to and from memory in response to requests generated by peripheral devices. The peripheral devices do not directly supply addresses and commands to the memory controller. Instead, the states required to manage a data stream are maintained in 32 DMA channels in the DMAC. Every DMA request from a peripheral device generates a memory-bus transaction. The processor can access the peripheral bus through the bridge portion of the DMA, thus bypassing the system DMA controller. Therefore, in this document, the DMAC is the DMA bridge. The DMAC supports only flowthrough transfers. The available types of transfers are as follows:

- Internal SRAM to/from internal SRAM, external DRAM, internal peripherals, external peripherals, companion processors.
- External DRAM to/from internal SRAM, external DRAM, internal peripherals, external peripherals, companion processors.

8.1 PXA3xx Processor Differences

Table 104 shows the DMA Controller differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 104: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
External DMA Request signals	None	None	DREQ

8.2 Features

The DMAC provides the following features:

- Memory-to-memory, peripheral-to-memory, and memory-to-peripheral transfers in flowthrough mode.
- Flowthrough mode for transfers between flash memory and DDR SDRAM.
- 32 channels, 92 peripheral-device requests, and one external device request; capability to pre-program any request to any channel.
- Priority mechanism to process active channels (four channels with outstanding DMA requests at any given time).
- Each of the 32 channels can operate for Descriptor-fetch or no-Descriptor-fetch transfers (see Section 8.4.2, “DMA Descriptors” on page 299).
- Special Descriptor modes (Descriptor comparison and Descriptor branching).
- Retrieves trailing bytes in the Receive peripheral-device buffers.
- Programmable data-burst sizes (8, 16, or 32 bytes) and programmable peripheral device data widths (byte, half-word, or word).
- Up to 8191 bytes of data transfer per Descriptor. Larger transfers can be performed by chaining multiple Descriptors.
- Flow-control bits to process requests from peripheral devices. Requests are not processed unless the flow-control bit is set.

Copyright © 2009 Marvell

8.3 Signal Descriptions

The DREQ signal is controlled by the DMAC.

Table 105: DMAC External Signal List

Signal Name	Direction	Description
DREQ (PXA32x Only)	Input	External Companion Chip Request Line The DMAC detects the positive edge of this signal to log a request. The external companion chip asserts the DREQ signal when a DMA transfer request is required. The DREQ signal must remain asserted for four CLK_MEM cycles to allow the DMAC to recognize the low-to-high transition (see Figure 17). When the DREQ signal is de-asserted, it must remain de-asserted for at least four system clock cycles. The DMAC registers the transition from low to high to identify a new request. The external companion chip need not wait until the completion of the data transfer before asserting the next request. It can have up to 31 outstanding requests on the DREQ pin. The number of pending requests are logged in special status registers, DRQSRx. Requests on DREQ can be used for data transfers for Flowthrough mode.

Copyright © 2009 Marvell

8.4 Operation

The DMAC can be configured to transfer data using flowthrough DMA. The DMAC has 32 configurable channels. See Figure 16.

Figure 16: DMAC Block Diagram

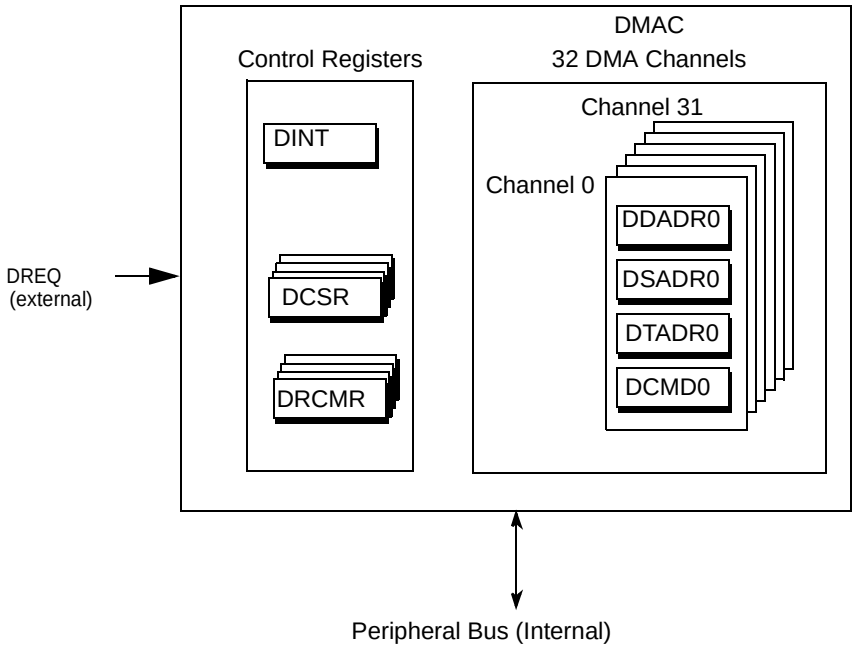
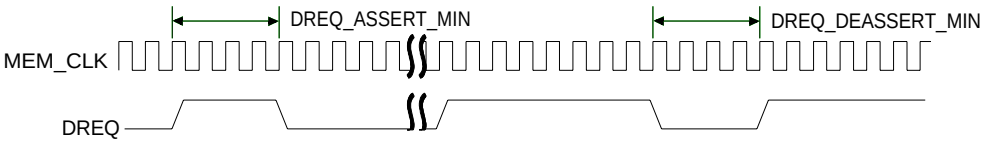


Figure 17: DREQ Timing Requirements



8.4.1 DMA Channels

Each of the 32 DMA channels is controlled through four 32-bit registers:

- DMA Descriptor Address Registers (DDADRx)
- DMA Source Address Register (DSADRx)
- DMA Target Address Registers (DTADRx)
- DMA Command Registers (DCMDx)

Each channel can be configured to service any kind of transfer. Each channel is serviced in increments of the device burst size and delivered in the granularity of the device port width. The burst size and port width for each device are programmed in the channel registers and are based on the device FIFO depth and bandwidth requirements. When multiple channels are actively executing, each channel is serviced with a burst of data. Then the DMA controller performs a context switch to

another active channel. The context switches are based on whether a channel is active, whether its target device is currently requesting service, and the channel priority.

8.4.1.1 DMA Channel-Priority Scheme

The DMA channel-priority scheme helps to ensure that peripherals are serviced according to their bandwidth requirements. Assign a higher priority to peripherals with higher bandwidth requirements and a lower priority to peripherals with lower bandwidth requirements. This priority scheme ensures that higher-bandwidth peripherals are serviced more often than lower-bandwidth peripherals.

The DMA channels are divided internally into four sets of eight channels each. The channels in each set use a round-robin priority. Set 0 has the highest priority and Set 3 has the lowest. Program the modules with the shortest latency requirement into set 0. Program memory-to-memory moves and low-bandwidth peripherals to use Sets 2 and 3. Refer to Table 106 for details.

When all the channels are running concurrently, Set 0 is serviced four out of every eight consecutive channel-servicing instances, Set 1 is serviced twice, and Sets 2 and 3 are each serviced one time.

For example, if all the channels request data transfers, the sets are prioritized in the following order: Set 0, Set 1, Set 0, Set 2, Set 0, Set 1, Set 0, Set 3. After eight channel-servicing instances, the pattern repeats. The channels in each set are given a round-robin priority.

Table 106: Channel Priority

Set	Channels	Priority	Number of Times Served
0	0, 1, 2, 3, 16, 17, 18, 19	Highest	4 / 8
1	4, 5, 6, 7, 20, 21, 22, 23	Higher than 2 and 3. Lower than 0.	2 / 8
2	8, 9, 10, 11, 24, 25, 26, 27	Higher than 3. Lower than 0 and 1.	1 / 8
3	12, 13, 14, 15, 28, 29, 30, 31	Lowest	1 / 8

8.4.1.2 Channel States

The following states apply to the DMA channels:

- Uninitialized. Occurs after a reset. DCSRx[STOPINTR] is set when the channels are uninitialized.
- Not running. Either a valid Descriptor is loaded into the DDADRx register during a Descriptor-fetch transfer or valid DSADRx, DTADRx, and DCMDx registers are programmed during a no-Descriptor-fetch transfer, but the corresponding run bit, DCSRx[RUN], is not set. For a no-Descriptor-fetch transfer, DCSRx[STOPINTR] is not cleared when the registers are programmed. For a Descriptor-fetch transfer, DCSRx[STOPINTR] is cleared when the DMA controller updates the DDADRx register.
- Running. For a Descriptor-fetch transfer, after the DDADRx is programmed and the DCSRx[RUN] bit is set, four words of Descriptors are fetched from the memory and DCSRx[STOPINTR] remains clear. For a no-Descriptor-fetch transfer, after the DSADRx, DTADRx and DCMDx registers are programmed and DCSRx[RUN] is set, the corresponding channel clears the DCSRx[STOPINTR] bit, skips the Descriptor-fetch, running state and enters the Wait for Request or Transfer Data state (see Figure 18).
- Wait for request. The channel waits for a request before it starts to transfer data; DCSRx[STOPINTR] is clear.
- Transfer data. Transferring data between the source and the target; DCSRx[STOPINTR] is clear.

Copyright © 2009 Marvell

- Channel error. The channel with the error remains in the stopped state until software clears the error condition, re-initializes the channel, and sets the DCSRx[RUN] bit and the DCSRx[BUSERRINTR] bit. See Section 8.5.9 and Section 8.5.8.
- Stopped. The channel is stopped. DCSRx[STOPINTR] is set. For a no-Descriptor-fetch transfer, a stopped channel is re-initialized by updating the DSADRx, DTADRx and DCMDx registers and then setting DCSRx[RUN]. For a Descriptor-fetch transfer, a stopped channel is re-initialized by updating the DDADRx register and setting DCSRx[RUN].

Table 107 summarizes the DMAC channel states.

Table 107: Channel States Based on Software Configuration

Descriptor Mode	Software Configuration	DCSRx[RUN]	DCSRx[STOPINTR]	Resulting Channel State
Descriptor-Fetch mode	Power-up	0	1	Uninitialized
	Write to DDADRx before DCSRx[RUN] is set (recommended flow)	0	0	Valid Descriptor, not running
	Set DCSRx[RUN] after writing to DDADRx (recommended flow)	1	0	Running
	Set DCSRx[RUN] before writing to DDADRx (not recommended)	1	1	Invalid
	Write to DDADRx after DCSRx[RUN] is set (not recommended)	1	0	Descriptor fetch, running.
	Stop running channel by clearing DCSRx[RUN] and DCSRx[MASKRUN]	0	$0 \rightarrow 1$	Channel, if not immediately, eventually switches to a stopped state (identified by DCSRx[STOPINTR] toggling from low to high).

Table 107: Channel States Based on Software Configuration (Continued)

Descriptor Mode	Software Configuration	DCSRx[RUN]	DCSRx[STOPINTR]	Resulting Channel State
No-Descriptor-Fetch mode	Power-on	0	1	Uninitialized
	Write to DSADRx, DTADRx and DCMDx before DCSRx[RUN] is set (recommended flow)	0	1	Valid Descriptor, not running
	Set DCSRx[RUN] after configuring DSADRx, DTADRx, and DCMDx (recommended flow)	1	0	Running
	Set DCSRx[RUN] before configuring DSADRx, DTADRx, and DCMDx (not recommended)	1	0	Wait for Request, running. Channel uses current DSADRx, DTADRx, and DCMDx for the transfer and may lead to unpredictable results.
	Stop running channel by clearing DCSRx[RUN] and DCSRx[MASKRUN]	0	$0 \rightarrow 1$	Channel, if not immediately, eventually switches to a stopped state (identified by DCSRx[STOPINTR] toggling from clear to set)

Copyright © 2009 Marvell

8.4.2 DMA Descriptors

The DMA Descriptor consists of four 32-bit words in memory. This structure is loaded into the four control registers. The DMA Descriptor describes the next DMA operation for the particular channel. The DMA Descriptors must be aligned on 16-byte boundaries in memory:

- Word 0 contains a value for the DDADRx register and a single flag bit (STOP).
- Word 1 contains a value for the DSADRx register.
- Word 2 contains a value for the DTADRx register.
- Word 3 contains a value for the DCMDx register.

The DMAC can operate in two distinct modes based on the DCSRx[NODESCFETCH] bit:

- DCSRx[NODESCFETCH] = 0 Descriptor-fetch transfer
- DCSRx[NODESCFETCH] = 1 No Descriptor-fetch transfer

8.4.2.1 Descriptor-Fetch Transfer Operation

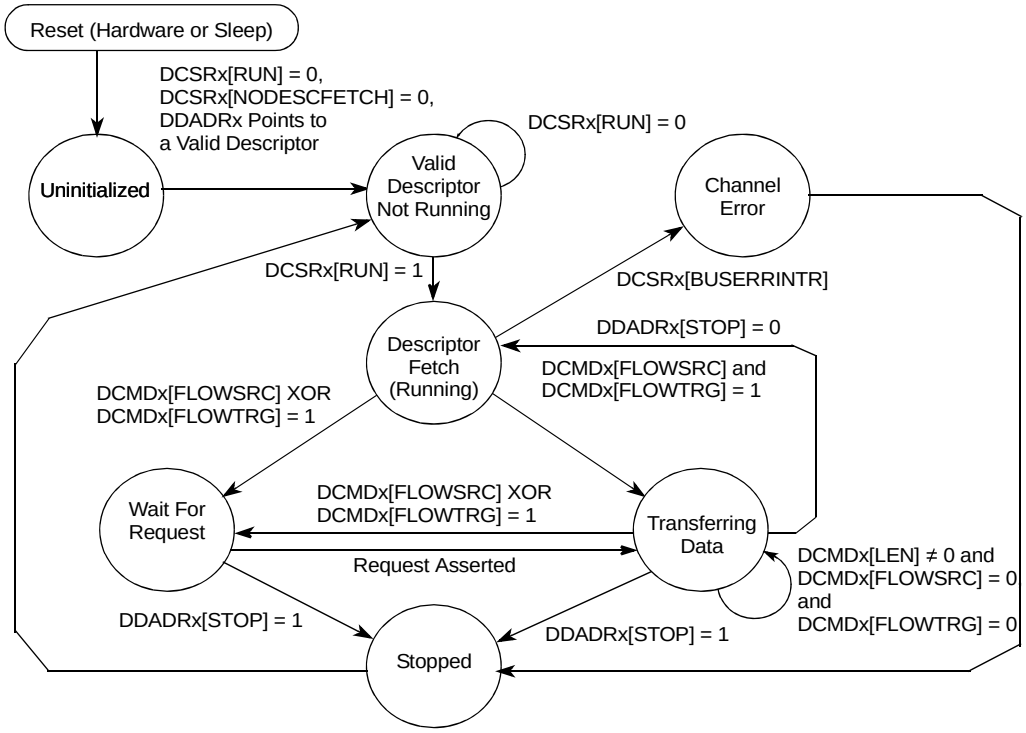
A Descriptor-fetch transfer operation proceeds as follows:

1. Software clears the DCSRx[RUN] bit and then clears the DCSRx[NODESCFETCH] bit.
2. Software writes a valid Descriptor address to the DDADRx register and then sets DCSRx[RUN].

   This sequence enables the DMAC to fetch the four-word Descriptor (if the memory is already set up with the Descriptor chain) from the memory indicated by the DDADRx register.
3. The channel either waits for a request or starts the data transfer, as determined by the DCMDx[FLOW] source and target bits.
4. After the channel transfers a number of bytes equal to the smaller of DCMDx[SIZE] and DCMDx[LEN], it either waits for the next request or continues with the data transfer until the DCMDx[LEN] reaches zero.
5. The channel stops or continues with a new Descriptor fetch from the memory, as determined by the DDADRx[STOP] bit.

Figure 18 summarizes this operation. If an error occurs during the fetch operation, the channel enters the stopped state and remains there unless software clears the error condition, re-initializes the channel, and sets the DCSRx[RUN] bit. The DTADR and DSADR registers are not writable in Descriptor-fetch mode.

Figure 18: Flow of Descriptor Fetch Transfer Operation



Note

When a channel switches between a Descriptor-fetch transfer and a no-Descriptor-fetch transfer, it must be stopped before the mode switch.

For a Descriptor-fetch transfer, the software must load the DDADRx register and set the DCSRx[RUN] bit. The channel-Descriptor fetch does not occur unless the DCSRx[RUN] bit is set.

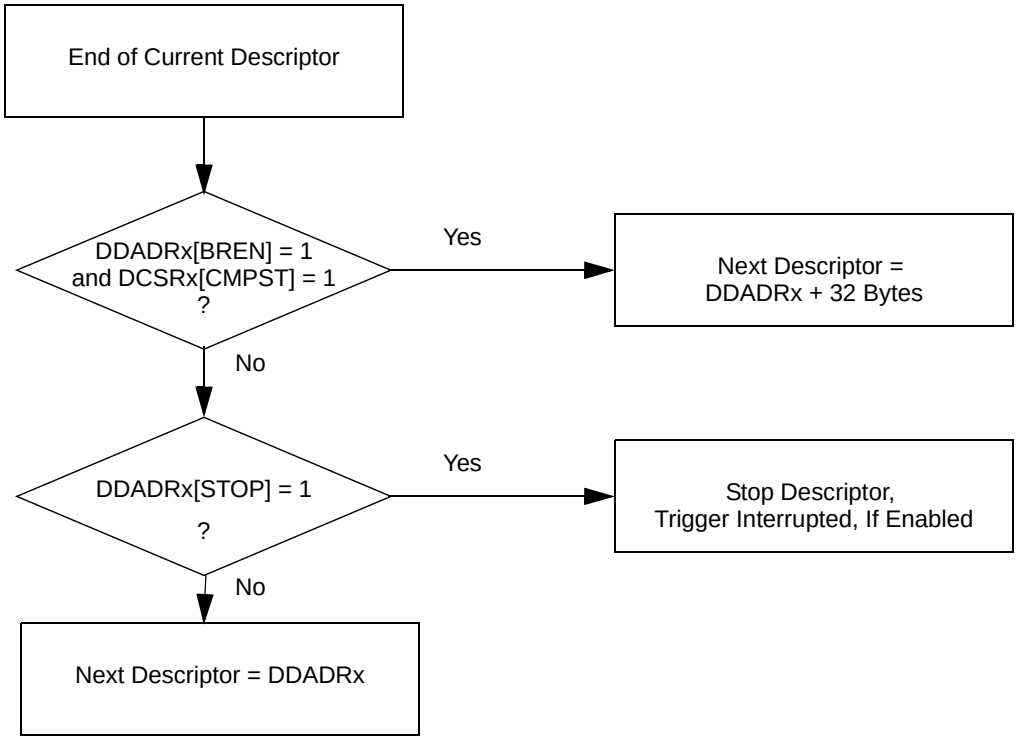
Although software loads the DDADRx register, the DSADRx, DTADRx, and DCMDx registers must be loaded indirectly from the DMA Descriptors. The DMA Descriptors pointed to by the DDADRx register are loaded into the registers of the associated DMA channel when a Write to the DCSRx[RUN] bit switches the channel from stopped to running.

Bit 0 (STOP) of Word 0 of a DMA Descriptor (the low bit of the DDADRx field) is used to mark the special Descriptor, which resides at the end of a Descriptor list. The value of the stop bit does not affect the loading of the fields of a Descriptor into Channel registers in any way; however, if a Descriptor with the stop bit set is loaded into a channel register, the channel stops after completely transferring the data pertaining to that channel.

If DDADRx[BREN] and DCSRx[CMPST] are set, the DMAC fetches the next Descriptor from the address in the DDADRx register + 32 bytes. If either of the bits is cleared, the DMAC fetches the next Descriptor from the address in the DDADRx register. DDADRx[BREN] is relevant only for Descriptor-fetch transfers (DCSRx[NODESCFETCH] is cleared).

Copyright © 2009 Marvell

Figure 19: Flow Chart for Descriptor Branching



8.4.2.2 No-Descriptor-Fetch Transfer Operation

The channel is uninitialized after reset for the typical no-Descriptor-fetch transfer (DCSRx[NODESCFETCH] = 1) operation. A no-Descriptor-fetch data transfer proceeds as follows:

1. Software clears DCSRx[RUN] and then sets DCSRx[NODESCFETCH].
2. Software must write a valid source physical address to the DSADRx register, a target physical address to the DTADRx register, and a command to the DCMDx register.

   The DDADRx register is reserved in this mode and must not be written.
3. Next, the DCSRx[RUN] bit must be set. A Descriptor fetch is not performed.
4. The channel either waits for a request or starts the data transfer, depending on the values of the DCMDx[FLOWSRC] and DCMDx[FLOWTRG] bits.
5. After transferring the number of bytes equal to the smaller of DCMDx[SIZE] and DCMDx[LEN], the channel either waits for the next request or continues with the data transfer until the DCMDx[LEN] reaches zero, depending on the priority of the enabled DMA channels.
6. The channel stops when DCMDx[LEN] reaches zero.

Figure 20 summarizes this operation. Setting DCSRx[STOPIRQEN] to trigger a stop interrupt may cause the DMAC to trigger the stop interrupt prematurely, possibly even before the channel enters active-run mode. See the STOPIRQEN and DCSRx[STOPINTR] descriptions in Table 120 for details on the RUN condition. To detect stoppage of a channel as the result of an End-of-Receive (EOR) from a peripheral, software must validate the stoppage by checking the status of DCSR information on the EORINT bit. Detect normal channel stoppage by using the end interrupt (DCSRx[ENDINTR]), not the stop interrupt (DCSRx[STOPINTR]). See Table 120 for information on the end interrupt.

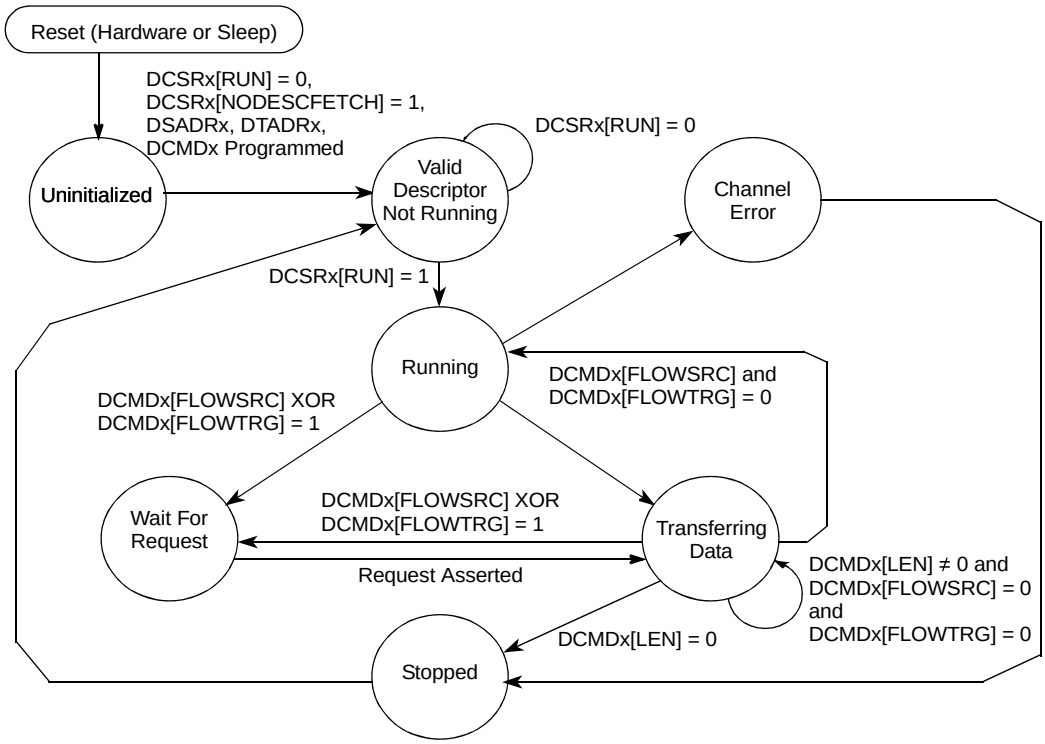
If an error occurs during the fetch operation, the channel stops until software clears the error condition and sets the DCSRx[RUN] bit.

Note

When a channel switches between a Descriptor-fetch transfer and a no-Descriptor-fetch transfer, the channel must be stopped before the switch occurs.

A few DMA channels can be used for a Descriptor-fetch transfer and the remaining channels in a no-Descriptor-fetch transfer simultaneously. Do not program the DDADRx register of the channels during a no-Descriptor-fetch transfer.

Figure 20: No-Descriptor-Fetch Transfer Channel State Diagram



8.4.3 Transferring Data

The internal peripherals connected to the DMA through the peripheral bus operate as flowthrough transfers. Although the source or destination of a DMA transfer is usually a peripheral, the DMAC can also transfer data through memory-to-memory moves.

8.4.3.1 Servicing Internal Peripherals

The processor has two types of internal peripherals: the *peripheral-bus peripherals* and the *system-bus peripherals*. Peripherals such as AC'97, MSL, and UART are examples of peripheral bus peripherals. The DMAC provides DMA requests to Channel Map registers (DRCMRx) that contain five bits of channel number for each possible DMA request. These peripheral requests are mapped to 32 available channels. If the internal peripheral address resides in the DSADRx register, the DCMDx[FLOWSRC] bit must be set to allow the processor to wait for the request before it initiates

Copyright © 2009 Marvell

the transfer. If the internal peripheral address resides in the DTADRx register, the DCMDx[FLOWTRG] bit must be set. If DCMDx[ENDIRQEN] is set, a DMA interrupt is requested at the end of the last cycle associated with the byte that caused DCMDx[LEN] to decrement to zero.

DMA Read Cycles

A DMA Read cycle is a Read from memory and a Write to the peripheral. A DMA Read begins when an internal peripheral sends a request to a channel in the DMAC while the channel is running. The number of bytes to be transferred is specified using DCMDx[LEN]. The following process begins when the request is recognized:

1. The DMAC prompts the memory controller to read from memory the required number of bytes addressed by DSADRx into a 32-byte buffer in the DMAC.
2. The DMAC transfers the data to the peripheral device addressed in DTADRx[31:0]. The DCMDx[WIDTH] specifies the width of the internal peripheral to which the transfer is being made.
3. At the end of the transfer, DSADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

Use the following settings for the DMAC register bits for a DMA read from an internal peripheral:

- DSADRx[SRCADDR] = memory address
- DTADRx[TRGADDR] = internal peripheral address
- DCMDx[INCSRCADDR] = 1
- DCMDx[INCTRGADDR] = 0
- DCMDx[FLOWSRC] = 0
- DCMDx[FLOWTRG] = 1

DMA Write Cycles

A DMA write cycle is a Read from the peripheral and a Write to memory. A DMA Write begins when an internal peripheral sends a request to a channel in the DMAC while the channel is running. The number of bytes to be transferred is specified by DCMDx[SIZE]. When the request is recognized, the following process begins:

1. The DMAC processes the request by transferring the required number of bytes from the peripheral device addressed by DSADRx[31:0] into a DMAC buffer.
2. The DMAC transfers the data to the memory controller. The DCMDx[WIDTH] specifies the width of the internal peripheral that is the source of the transfer.
3. At the end of the transfer, DTADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

Use the following settings for the DMAC register bits for a DMA Write from an internal peripheral:

- DSADRx[SRCADDR] = internal peripheral address
- DTADRx[TRGADDR] = memory address
- DCMDx[INCSRCADDR] = 0
- DCMDx[INCTRGADDR] = 1
- DCMDx[FLOWSRC] = 1
- DCMDx[FLOWTRG] = 0

8.4.3.2 Servicing External Companion Chips (PXA32x Only)

External companion chips can be serviced using DREQ using flowthrough transfers. The DMAC provides DMA requests to Channel Map registers (DRCMRx) that contain five bits of channel number for each of the possible DMA requests. A request can be mapped to any of the eight channels in a priority set. The DMAC detects a request by identifying a low-to-high transition on the DREQ (the timing information is explained in Section 8.4). The DMAC then performs a lookup on the

corresponding bits of the DRCMRx register. The lookup maps the request to one of the channels. If the external companion-chip address resides in the DSADRx register, the DCMDx[FLOWSRC] bit must be set. If the external companion-chip address resides in the DTADRx register, the DCMDx[FLOWTRG] bit must be set to allow the processor to wait for the request before it initiates the transfer.

If DCMDx[ENDIRQEN] is set, a DMA interrupt is requested at the end of the last cycle associated with the byte that caused DCMDx[LEN] to decrement to zero.

Servicing Companion Chips: Flowthrough DMA Read Cycles

A flowthrough-DMA Read begins when an external companion chip sends a DREQ request to a channel in the DMAC while the channel is running and is configured for flowthrough Reads. The number of bytes to be transferred is specified using DCMDx[SIZE]. The following process begins when the request is recognized:

1. The DMAC prompts the static memory controller to read the required number of bytes addressed by DSADRx[31:0] from the companion chip.
2. The bytes are transferred into a 32-byte buffer in the DMAC.
3. The DMAC transfers the bytes to the external or internal device or memory addressed in DTADRx[31:0].
4. At the end of the transfer, DSADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

   Use the following settings for the DMAC register bits for a flowthrough DMA Read from an external companion chip:DSADRx[SRCADDR] = companion chip address

   DTADRx[TRGADDR] = memory address or external or internal device address

   DCMDx[INCSRCADDR] = 1

   DCMDx[INCTRGADDR] = 0 DCMDx[FLOWSRC] = 0

   DCMDx[FLOWTRG] = 1

Servicing Companion Chips: Flowthrough DMA Write Cycles

A flowthrough-DMA Write begins when an external companion chip sends a request via the DREQ bus to a channel in the DMAC while the channel is running and is configured for flowthrough Writes. The number of bytes to be transferred is specified using DCMDx[SIZE]. The following process begins when the request becomes the highest priority request:

1. The DMAC reads the required number of bytes from the external or internal device or memory addressed by DSADRx into a DMAC buffer.
2. The DMAC transfers the bytes to the static memory controller.
3. The static memory controller writes the bytes to the companion chip.
4. At the end of the transfer, DTADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

Use the following settings for the DMAC register bits for a flowthrough-DMA Write to an external companion chip:

DSADRx[SRCADDR] = memory address or external or internal device address

DTADRx[TRGADDR] = companion chip address

DCMDx[INCSRCADDR] = 0 DCMDx[INCTRGADDR] = 1

DCMDx[FLOWSRC] = 1

DCMDx[FLOWTRG] = 0

Copyright © 2009 Marvell

8.4.3.3 Memory-to-Memory Moves

Memory-to-memory moves do not involve request signals. The processor writes to the DCSRx[RUN] bit indicated by the channel that is configured to perform a memory-to-memory move. Software must clear the DCMDx[FLOWSRC] and DCMDx[FLOWTRG] bits as soon as the Descriptor is fetched. The transfer then begins. If DCMDx[ENDIRQEN] is set, a DMA interrupt is requested at the end of the last cycle associated with the byte that caused DCMDx[LEN] to decrement to zero.

Note

Memory-to-memory moves can be performed from internal SRAM to external memory and *vice versa*.

DMA Read/Write Cycles

A DMA memory-to-memory Read or Write begins when the processor sets the DCSRx[RUN] bit. If the channel is in a Descriptor-fetch transfer, it fetches the four-word Descriptor. The DCMDx[FLOWSRC] and DCMDx[FLOWTRG] bits must be cleared for a memory-to-memory move. The channel begins to transfer data without waiting for a PREQ assertion. The number of bytes to be transferred is specified using DCMDx[LEN]. Processing begins as follows:

1. The DMAC prompts the memory controller to read the required number of bytes addressed by DSADRx into a 32-byte buffer in the DMAC.
2. The DMAC generates a Write cycle to the to the location addressed by DTADRx.
3. At the end of the transfer, DSADRx and DTADRx are increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

Use the following settings for the DMAC register bits for memory-to-memory moves:

- DSADRx[SRCADDR]= memory address
- DTADRx[TRGADDR]= memory address
- DCMDx[INCSRCADDR] = 1
- DCMDx[INCTRGADDR] = 1
- DCMDx[FLOWSRC] = 0
- DCMDx[FLOWTRG] = 0

8.4.4 Programming Tips

This section provides information about software requirements, instruction ordering, and misaligned memory accesses.

8.4.4.1 Software Management Requirements

The information that must be maintained on a per-stream basis (for example, the memory address, the peripheral address, the transfer count, and the implied direction of data flow) is maintained in Descriptor registers in the DMAC. These registers are loaded from memory locations specified by the software. Multiple DMA Descriptors can be chained together in a list so that a DMA channel can transfer data to and from a number of separate locations. Descriptors can be dynamically added to an active DMA channel Descriptor chain, which is particularly useful in applications that involve network-transmit lists and network-receiver buffer-free lists.

Each demand for data that a peripheral generates is a memory data Read or Write. A peripheral must not request a DMA transfer unless it is prepared to read or write the full data block (8, 16, or 32 bytes) and can handle trailing bytes that may occur at the end of a DMA transfer.

Copyright © 2009 Marvell

8.4.4.2 Programmed I/O Operations

The processor can read and write the peripheral registers and FIFOs on the peripheral bus. The peripheral internal registers must be accessed using word-access loads and stores. Internal register and FIFO space must be mapped as non-cacheable. Byte and half-word accesses to internal registers are not allowed. The FIFOs of some of the peripherals on the peripheral bus are accessed using byte, half-word, or word-access loads and stores. Refer to the individual peripheral chapters for details.

8.4.4.3 Instruction Ordering

The DMAC executes programmed I/O instructions in the order specified by the software. References to internal addresses generally complete more quickly than those issued to external addresses because of memory accesses that can be sent in one order and completed in a different order.

The DMAC ensures that memory references made by a single DMA channel are presented to memory in order and that the Descriptor fetches occur between the data blocks. The order in which the accesses are completed cannot be guaranteed unless the channels refer to only one type of memory (either to the external memory or to the internal SRAM).

The channel must not reference both internal and external memory in a DMA Descriptor chain for the following operations:

- Self-modifying DMA Descriptor chains.
- Channels that write data blocks followed by status blocks while another channel (generally the processor) polls a field in the status block.

8.4.4.4 Misaligned Memory Accesses

The DMAC is a 64-bit device that can access memory on byte-aligned boundaries. The DMAC may encounter misaligned (not aligned to 64-bit boundary) addresses while it accesses memory. Only the following type of data transfers can access misaligned addresses:

- Memory-to-memory transfers.
- Memory-to-peripheral transfers or peripheral-to-memory transfers. In this case, the peripheral addresses are 32-bit aligned.

Addresses must be 64-bit aligned in the Compare-Descriptor mode.

Note

All companion processor-related transfers must use 64-bit aligned addresses for both source and target locations.

The DMAC employs channel-specific alignment buffers that hold either the leading or the lagging misaligned data. These buffers must be empty when the DMAC performs a context switch to service the next pending channel. When the DMAC completes a Descriptor transfer, it ensures that all data in the alignment buffers is properly flushed to its respective targets.

Because the DMAC encounters overhead while it works with misaligned data, restricting memory addresses to 8-byte boundaries can be helpful. Align the source and target addresses to 32-byte boundaries for highest DMAC and memory controller performance.

By default, during data transfers the DMA controller forces the least significant three bits of all external addresses to zeros and the least significant two bits of all peripheral addresses to zeros. To activate byte-aligned addressing, software must activate the alignment register. See Table 122 for details.

Copyright © 2009 Marvell

8.4.5 How DMA Handles Trailing Bytes

DMA normally transfers bytes equal to the transaction size specified by DCMDx[SIZE]. However, if the Descriptor is reaching its end, the number of trailing bytes in the DCMDx[LEN] field can be smaller than the transfer size. The DMA can transfer the exact number of trailing bytes if the DCMDx[FLOWSRC] and DCMDx[FLOWTRG] bits are both cleared or if it receives a corresponding request from the internal/external peripheral or companion processor. The following examples are possible:

- Memory-to-memory moves. The DMA transfers bytes equal to the smaller of DCMDx[LEN] or DCMDx[SIZE].
- Memory-to-internal peripheral transfers. Most internal peripherals send a request for trailing bytes. Refer to the appropriate chapter of this manual for details on the operation of a specific peripheral. The DMA controller transfers a number of bytes equal to the smaller of DCMDx[LEN] and DCMDx[SIZE].
- Internal peripheral-to-memory transfers. Special handshaking signals and interrupts are employed for transferring trailing bytes from an internal peripheral to memory. The conditions that use the handshaking signals and interrupts are as follows:
  - End of Packet (EOP). The peripheral receives its last data sample from an external CODEC and detects an EOP based on its Receive protocol. Any remaining data samples in the peripheral Receive FIFO are treated as trailing bytes. The peripheral can be programmed to initiate a DMA request, even if it has fewer bytes than its Receive trigger threshold. The DMA responds to this request and reads out the trailing bytes. CPU intervention is not required as long as the Descriptor chain has not ended.
  - Time Out (TO). Peripherals that do not support EOP protocols use a timeout mechanism to determine whether they have received their last data sample. Any remaining data samples in the peripheral Receive FIFO are treated as trailing bytes. The peripheral can be programmed to initiate a DMA request even if it has fewer bytes than its Receive trigger threshold. The DMA controller responds to the DMA request and reads out the trailing bytes. CPU intervention is not required as long as the Descriptor chain has not ended.

Note

When a peripheral signals either an EOP or a TO from an external device, the DMAC sets the End-of-Receive (EOR) status bit in the corresponding channel Control Status register (DCSRx). See Table 120 for details.

  - End-of-Descriptor chain (EOC). Indicates that a DMA channel is at the end of its last Descriptor. After the current transfer, DCMDx[LEN] = 0 and DDADRx[STOP] = 1. The DMAC signals the peripheral on an EOC and the peripheral interrupts the CPU to retrieve any trailing bytes. Refer to the individual peripheral chapters for details on the processor interrupt implementation. EOC is the only trailing-bytes case that requires programmed I/O to retrieve data.
  - Request-after-channel-stops (RAS). A status bit in the DMAC Control Status register (DCSRx) is set when a peripheral asserts a DMA request after the channel to which the peripheral is mapped stops. See Table 120 for details.

The following examples illustrate the handling of various trailing bytes using EOR, EOC, and RAS.

Example 1. The peripheral signals a DMA request to service trailing bytes in its Receive FIFO (RxFIFO). The current Descriptor DCMDx[LEN] is equal to or greater than the trailing-byte count.

1. The peripheral signals a Receive DMA request.
2. The DMAC responds and reads out all trailing bytes including the last byte.
3. The peripheral signals an EOR.

4. The DMAC transfers all trailing bytes to the channel target and then updates DCSRx[EORINT].
5. The DMA channel can be configured to stop, jump, or just wait for another request after receiving EOR, depending on DCSRx[EORSTOPEN] and DCSRx[EORJMPEN]. The EORINT bit must be cleared before a channel is restarted.
6. DCSRx[EORINT] set indicates that all trailing bytes were read and transferred to the required target.

Example 2. The peripheral signals a DMA request to signal an EOR only (RxFIFO is empty). The current Descriptor DCMDx[LEN] is greater than zero.

1. The peripheral signals a Receive DMA request.
2. The peripheral responds to the DMAC that the RxFIFO is empty, and the DMAC does not read any data from the RxFIFO.
3. The peripheral signals an EOR.
4. The DMAC transfers any previously read data (temporarily stored in a local DMA buffer) to the channel target and then updates DCSRx[EORINT].
5. The DMA channel can be configured to stop, jump, or just wait for another request after receiving EOR, depending on DCSRx[EORSTOPEN] and DCSRx[EORJMPEN]. The EORINT bit must be cleared before a channel is restarted.
6. DCSRx[EORINT] set indicates that all trailing bytes were read and transferred to the required target.

Example 3. The peripheral signals a DMA request to service trailing bytes in its RxFIFO. The current Descriptor DCMDx[LEN] is less than the trailing-bytes count. More Descriptors are available in the Descriptor chain.

1. The peripheral signals a Receive DMA request.
2. The DMAC responds and reads only the number of trailing bytes programmed by DCMDx[LEN].
3. The current Descriptor DCMDx[LEN] decrements to zero. This condition forces the DMAC to transfer all the data read from the peripheral RxFIFO to the channel target.
4. The DMAC fetches the next Descriptor.
5. Because the peripheral still has trailing bytes in its RxFIFO, it must generate another request. The peripheral must continue to issue such requests until the DMAC reads out all the trailing bytes and until an EOR is signaled.
6. DCSRx[EORINT] set indicates that all trailing bytes are read and transferred to the required target. The EORINT bit must be cleared before a channel is restarted.

Case 4. The peripheral signals a DMA request to service trailing bytes in its RxFIFO. The current Descriptor DCMDx[LEN] is less than the trailing-bytes count. No more Descriptors are available in the Descriptor chain.

1. The peripheral signals a Receive DMA request.
2. The DMAC responds and reads only the number of trailing bytes programmed by DCMDx[LEN].
3. The current Descriptor DCMDx[LEN] decrements to zero. This condition forces the DMAC to transfer all the data read from the peripheral RxFIFO to the channel target.
4. The DMA channel stops because there are no more Descriptors in the Descriptor chain.
5. The peripheral must set a status bit in a local (peripheral) status register if the DMAC signaled an end-of-chain (EOC) while reading the last byte. This setting indicates that software should handle the remainder of the trailing bytes in the peripheral RxFIFO.
6. The peripheral signals a DMA request if the DMAC fails to signal an EOC. The DMAC sets DCSRx[RAS] because the DMA channel has already stopped. This setting indicates that software should handle the remainder of the trailing bytes in the peripheral RxFIFO.

Copyright © 2009 Marvell

> Note
>
> The DMAC signals an EOC only if the last Descriptor DCMDx[LEN] of the DDescriptorescriptor chain is greater than zero.

Peripherals that require the DMA services for processing trailing bytes must support the preceding signals (TO, EOP, EOR, EOC, and RAS). The following peripherals require DMA services to process trailing bytes:

- UART1, UART2, UART3
- SSP1, SSP2, SSP3, SSP4
- USB device controller (UDC) (PXA32x and PXA30x Only)
- USIM
- MMC1 and SDIO1
- MMC2 and SDIO2
- MMC3 and SDIO3 (PXA31x Only)

8.4.6 Quick Reference to DMA Programming

Table 108, Table 109, Table 110, and Table 111 tabulate width, alignment, and address increment modes for various DMA data transfers.

Table 108: Configuration for Peripheral Bus Peripheral Related Data Transfers

Source	Target	Source Alignment (bytes)	Target Alignment (bytes)	DCMD [INCSRCADDR] (binary)	DCMD [INCTRGADDR] (binary)	DCMD [WIDTH] (binary)
Peripheral Bus Peripheral	Memory	4	1	0 or 1	1	01, 10 or 11
Memory	Peripheral Bus Peripheral	1	4	1	0 or 1	01, 10 or 11
Note: Memory refers to all types of memory explained in the Memory Controller chapter (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).						

If a memory address is byte-aligned, the DALGN register must be programmed. Refer to Section 8.5.10. If DCMDx[INCSRCADDR] or DCMDx[INCTRGADDR] is set, the DMAC increments the source or target address, after each transaction, by a number equal to the width of the peripheral bus peripheral. For example, if the DMAC is transferring 32 bytes of data from memory to a 4-byte wide peripheral on the peripheral bus, the DMAC writes four bytes to the peripheral eight times and increments the target address by four bytes after each of the eight transactions.

Table 109: Configuration for Memory-to-Memory Data Transfers

Source	Target	Source Alignment (bytes)	Target Alignment (bytes)	DCMD [INCSRCADDR] (binary)	DCMD [INCTRGADDR] (binary)	DCMD [WIDTH] (binary)
Memory	Memory	1	1	1	1	00
Expansion memory FIFO	Memory	1	1	0	1	00
Memory	Expansion memory FIFO	1	1	1	0	00
Expansion memory FIFO	Expansion memory FIFO	1	1	0	0	00
Note: Memory refers to all types of memory explained in the Memory Controller chapter (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).						

If a memory address is byte aligned, the DALGN register must be programmed. Refer to Section 8.5.10.

Table 110: System Bus Peripheral Related Data Transfer Configuration

Source	Target	Source Alignment (bytes)	Target Alignment (bytes)	DCMD [INCSRCADDR] (binary)	DCMD [INCTRGADDR] (binary)	DCMD [WIDTH] (binary)
System Bus Peripheral	Memory	8	8	0 or 1	1	00
Memory	System Bus Peripheral	8	8	1	0 or 1	00
System Bus Peripheral	Expansion memory FIFO	8	8	0 or 1	0	00
Expansion memory FIFO	System Bus Peripheral	8	8	0	0 or 1	00
Note: Memory refers to all types of memory explained in the Memory Controller chapter (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).						

If DCMDx[INCSRCADDR] or DCMDx[INCTRGADDR] is set, the DMAC increments the source or target address after each bursting transaction by a number equal to the transaction burst size (8, 16, or 32 bytes) or DCMDx[LEN]. The latter is used if LEN is less than the burst size. For example, if the DMAC transfers 48 bytes of data from memory to the system bus peripheral in bursts of 32 bytes, it increments the target address by 32 after the first burst and then by 16 after the second burst.

Copyright © 2009 Marvell

Table 111: Configuration for Companion Processor (CC)-Related Data Transfers

Source	Target	Source Alignment (bytes)	Target Alignment (bytes)	DCMD [INCSRCADDR] (binary)	DCMD [INCTRGADDR] (binary)	DCMD [WIDTH] (binary)
CC or External Peripheral	Memory	8	8	0 or 1	1	00
Memory	CC or External Peripheral	8	8	1	0 or 1	00
CC or External Peripheral	Expansion memory FIFO	8	8	0 or 1	0	00
Expansion memory FIFO	CC or External Peripheral	8	8	0	0 or 1	00
Note: Memory refers to all types of memory explained in the memory controller chapter (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).						

Any memory can be used for data transfer mode. The companion processor or external peripheral must be connected as a variable-latency I/O memory. If DCMDx[INCSRCADDR] or DCMDx[INCTRGADDR] is set, the DMAC increments the source or target address after each bursting transaction by a number equal to the transaction burst size (8, 16, or 32 bytes) or DCMDx[LEN]. The latter is used if LEN is less than the burst size. For example, if the DMAC transfers 48 bytes of data from memory to the companion processor in bursts of 32 bytes, it increments the target address by 32 after the first burst and then by 16 after the second burst. Table 112 is a quick reference for programming the DMAC for the internal peripherals.

Table 112: DMA Quick Reference for Internal Peripherals

Unit	Function	FIFO Address	Width (bytes)	DCMDx Width (binary)	Burst Size (bytes)	Source or Target	DRCMRx
SSP4	receive	0x41A0_0010	1, 2, or 4	01, 10, or 11	8, 16, 32, or trailing	Source	0x4000_0108
	transmit	0x41A0_0010	1, 2, or 4	01, 10, or 11	8, 16, 32, or trailing	Target	0x4000_010C
UART2	receive	0x4020_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_0110
	transmit	0x4020_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_0114

Copyright © 2009 Marvell

Table 112: DMA Quick Reference for Internal Peripherals (Continued)

Unit	Function	FIFO Address	Width (bytes)	DCMDx Width (binary)	Burst Size (bytes)	Source or Target	DRCMRx
UART1	receive	0x4010_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_0118
	transmit	0x4010_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_011C
AC '97	microphone	0x4050_0060	4	11	8, 16, 32	Source	0x4000_0120
	modem receive	0x4050_0140	4	11	8, 16, 32	Source	0x4000_ 0124
	modem transmit	0x4050_0140	4	11	8, 16, 32	Target	0x4000_0128
	audio receive	0x4050_0040	4	11	8, 16, 32	Source	0x4000_012C
	audio transmit	0x4050_0040	4	11	8, 16, 32	Target	0x4000_0130
	surround transmit	0x4050_002C	4	11	8, 16, 32	Source	0x4000_117C
	Centre/LFE transmit	0x4050_0038	4	11	8, 16, 32	Source	0x4000_ 1180
SSP1	receive	0x4100_0010	1, 2, or 4	01, 10, or 11	8, 16, 32, or trailing	Source	0x4000_0134
	transmit	0x4100_0010	1, 2, or 4	01, 10, or 11	8, 16, 32, or trailing	Target	0x4000_0138
UART3	receive	0x4070_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_014C
	transmit	0x4070_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_0150
MMC/ SDIO 1	receive	0x4110_0040 (width = 1 byte) or 0x4110_0140 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Source	0x4000_0154
	transmit	0x4110_0044 (width = 1 byte) or 0x4110_0144 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Target	0x4000_0158

Copyright © 2009 Marvell

Table 112: DMA Quick Reference for Internal Peripherals (Continued)

Unit	Function	FIFO Address	Width (bytes)	DCMDx Width (binary)	Burst Size (bytes)	Source or Target	DRCMRx
MMC/ SDIO 2	receive	0x4200_0040 (width = 1 byte) or 0x4200_0140 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Source	0x4000_1174
	transmit	0x4200_0044 (width = 1 byte) or 0x4200_0144 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Target	0x4000_1178
MMC/ SDIO 3 (PXA31x Only)	receive	0x4250_0040 (width = 1 byte) or 0x4200_0140 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Source	0x4000_1190
	transmit	0x4200_0044 (width = 1 byte) or 0x4200_0144 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Target	0x4000_1194
USB 1.1 Client	endpoint 0	0x4060_0300	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0160
	endpoint A	0x4060_0304	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0164
	endpoint B	0x4060_0308	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0168
	endpoint C	0x4060_030C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_016C
	endpoint D	0x4060_0310	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0170
	endpoint E	0x4060_0314	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0174
	endpoint F	0x4060_0318	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0178
	endpoint G	0x4060_031C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_017C

Copyright © 2009 Marvell

Table 112: DMA Quick Reference for Internal Peripherals (Continued)

Unit	Function	FIFO Address	Width (bytes)	DCMDx Width (binary)	Burst Size (bytes)	Source or Target	DRCMRx
	endpoint H	0x4060_0320	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0180
	endpoint I	0x4060_0324	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0184
	endpoint J	0x4060_0328	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0188
	endpoint K	0x4060_032C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_018C
	endpoint L	0x4060_0330	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0190
	endpoint M	0x4060_0334	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0194
	endpoint N	0x4060_0338	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0198
	endpoint P	0x4060_033C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_019C
	endpoint Q	0x4060_0340	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01A0
	endpoint R	0x4060_0344	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01A4
	endpoint S	0x4060_0348	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01A8
	endpoint T	0x4060_034C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01AC
	endpoint U	0x4060_0350	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01B0
	endpoint V	0x4060_0354	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01B4
	endpoint W	0x4060_0358	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01B8
	endpoint X	0x4060_035C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01BC
USIM 1	Receive	0x4160_0000	1	01	8 or trailing	Source	0x4000_01F8
	Transmit	0x4160_0004	1	01	8 or trailing	Target	0x4000_01FC

Copyright © 2009 Marvell

Table 112: DMA Quick Reference for Internal Peripherals (Continued)

Unit	Function	FIFO Address	Width (bytes)	DCMDx Width (binary)	Burst Size (bytes)	Source or Target	DRCMRx
USIM 2	Receive	0x4210_0000	1	01	8 or trailing	Source	0x4000_116C
	Transmit	0x4210_0004	1	01	8 or trailing	Target	0x4000_1170
SSP2	Receive	0x4170_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Source	0x4000_013C
	Transmit	0x4170_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Target	0x4000_0140
SSP3	Receive	0x4190_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Source	0x4000_1108
	Transmit	0x4190_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Target	0x4000_110C
Wireless Trust Module (WTM)	Receive	0x43000008	4	11	8, 16, 32	Source	0x4000 111C
	Transmit 1	0x43000000	4	11	8, 16, 32	Target	0x4000 1120
	Transmit 2	0x43000004	4	11	8, 16, 32	Target	0x4000 1124
NAND IF	Receive and transmit data	0x4310_0040	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Both	0x4000_1184
	Command	0x4310_0030	4	11	16	Target	0x4000_118C

8.4.7 Examples

Example 4. Setting Up and Starting a Channel

The following example code shows how to set up and start a channel to transfer LEN words that start at the address in the DSADRx register to peripheral address in the DTADRx register. The stop bit in the DDADRx register is set so that the DMA channel stops after it completely transfers LEN bytes of data associated with this Descriptor.

```
// build real Descriptor
desc[0].ddadr = STOP;
desc[0].dsadr = DSADR;
desc[0].dtadr = DTADR;
desc[0].dcmd = DCMD;
// start the channel
DMANEXT[CHAN] = &desc[0];
DRUN = 1;
```

Example 5. Creating a Zero-Length Descriptor

The following example code shows how to initialize a Descriptor list for a channel that is running:

```
// Allocate a new Descriptor, and make it an End-
// Descriptor whose "ddadr" field points back at itself
```

```
    newDesc = New Desc();
    newDesc->ddadr = newDesc | STOP;
    // make it a zero length Descriptor
    newDesc->dcmd = ZERO;
    // Start the channel
    DMANEXT[CHAN] = newDesc;
    DRUN = 1;
```

The channel starts up, loads the Descriptor into its registers, detects that the transfer length is zero and the stop bit is set, and stops the channel. No data transfer occurs in this case. To restart the channel, write to its DDADRx register and then set the DCSRx[RUN] bit.

Example 6. Initializing a Channel for Use by a Direct DMA Master

The most effective way to move data between a peripheral device and memory is to use the built-in Descriptor-based DMA system. A free DMA channel is readily available in most applications. Some applications demand true DMA. Each application has different requirements, so a Descriptor-based DMA may be best for some applications while a non-Descriptor-based DMA is best for others. Choose the non-Descriptor-based DMA method for applications that cannot tolerate the time required to fetch a Descriptor before each DMA transfer. A Descriptor-based DMA method can reduce the amount of CPU intervention for applications that can tolerate it. True DMA can be offered to devices that require it by programming the Descriptor-based DMA system.

In the following example, the companion processor requires the DMAC to read four Descriptor words, program a channel as indicated by Descriptor, and continue the transfer based on the new Descriptor. Such a companion processor makes the following requirements:

1. The companion processor must connect to a GPIO or external interrupt pin through software.
2. Software then fetches four Descriptor words from one of the ports.
3. Based on the information contained in the Descriptor, transfer data from the source address to the destination address without waiting for another request from the companion processor.
4. Transfer the number of bytes mentioned in DCMDx[LEN] and then return to Step 1.

Such an external device can use a constant Descriptor in memory. A channel used by a direct DMA master must be initialized in a special way. The following example code shows how to initialize such a channel:

```
struct {long ddadr;
    long dsadr;
    long dtadr;
    short len;
    short dcmd;
} desc[2];
desc[0].ddadr = &desc[1];
desc[0].dsadr = I_ADR + I_DESC_OFFS;
desc[0].dtadr = &desc[1].dsadr;
desc[0].len = 8;
desc[0].dcmd = CMD_IncTrgAdr | CMD_FlowThru;
desc[1].ddadr = &desc[0];
desc[1].dtadr = I_ADR + I_DATA_OFFS;
desc[1].dsadr = 0;
desc[1].len = 0;
desc[1].dcmd = 0;
```

The external device sends a standard DMA request when it has data to transfer. The DMAC wakes up and reads four words from the device I_DESC_OFFS address. The DMAC transfers only four 32-bit words because the first Descriptor has an eight-byte count. The four words supplied by the external device are written over the DSADRx, DTADRx, and DCMDx registers of the next Descriptor. The DMAC then steps into the next (dynamically modified) Descriptor and uses the devices

Copyright © 2009 Marvell

I_DATA_OFFS address to process the requested transfer. When the data transfer is complete, the DMAC steps back to the first Descriptor and the process repeats.

Example 7. Adding a Descriptor to the End-of-Descriptor List (Channel Running)

Note

The following example assumes that a Descriptor-fetch transfer is active.

DMA Descriptor lists are used as queues of full buffers for network transmitters and as queues of empty buffers for network receivers. On-the-fly manipulation of DMA Descriptor lists must be as efficient as possible because each buffer can be small.

To add a Descriptor to the end of a Descriptor list for a running channel:

1. Clear the DCSRx[RUN] bit.
2. Wait for the channel to stop. The stop status is indicated in DCSRx[STOPINTR].
3. Create the end Descriptor in the memory with the stop bit set.
4. Manipulate the DDADRx register of the last Descriptor of the current chain in the memory to ensure that the register points to the newly-created end Descriptor in Step 3.
5. Create a new Descriptor with values in the DDADRx, DSADRx, DTADRx, and DCMDx registers that match those in the stopped DMA channel. The new Descriptor is the next Descriptor for this Descriptor list.
6. Examine the DMA channel registers and determine whether the channel stopped in the last Descriptor of the chain. If it did, manipulate the DDADRx register of this Descriptor so that it points to the newly created end Descriptor.
7. Program the channel DDADRx register with the next Descriptor created in Step 5.
8. Set DCSRx[RUN] = 1.

Example 8. Using Software Implementation of Full and Empty Bits

This example shows how to use the compare Descriptor and branch Descriptor modes so that software can implement full and empty bits. As shown in Figure 21, this implementation can be used to create a circular Descriptor chain to transfer a large amount of data. The full and empty bits are maintained in a memory table and are cleared prior to a DMA transfer. The DMAC uses the compare Descriptor mode to verify that the full and empty bits are cleared before it transfers data. If either bit is set, the DMAC branches to a Descriptor in which the DDADRx[STOP] bit is set, which terminates the transfer. The DMAC continues to perform a regular data transfer if the full and empty bits are both cleared. When the data transfer is complete, the DMAC updates the full and empty memory table by setting the full and empty bits, which indicates a successful transfer.

Note

The setting of the DALGNx bit is relevant even when DTADRx contains immediate data rather than an address. If DALGNx is not set, the lowest three bits of the fetched data are forced to 0, as they would be in an address case, forcing data to be 8-byte aligned. Thus, for any comparisons with DALGNx not set, the least significant three bits must always be 0 for a valid comparison with immediate data. This restriction does not apply when DALGN is set.

This example assumes the following memory addresses:

- SRC0, SRC1: source-address locations

- TRG0, TRG1: target-address locations
- FEUPDT: address pointing to data 0x0000_0003. Bit]1:0] = [Full:Empty].

In the example, the memory table contains the following full and empty bit information before the DMA transfers. Only the two least significant bits are valid. Bit[1:0] = [Full:Empty].

- FETBL0: 0x0000_0000
- FETBL1: 0x0000_0000

After successful DMA transfers, the same memory table contains the following information:

- FETBL0: 0x0000_0003
- FETBL1: 0x0000_0003

The following code is a chain of Descriptors for full- and empty-bit implementation:

First Descriptor Set

```
//First Descriptor
//Compare and Branch Descriptor modes enabled.

//No data transferred by this Descriptor.
//Source is indirectly addressed and target is directly addressed
//On a successful compare of &FETBL0 with 0x0000,
// Descriptor chain branches to desc[1] + 4*32bits, i.e desc[2].
//If Compare fails, then Descriptor chain jumps to desc[1].
//Desc[1] stops the channel as Full and Empty bits were not both 0.
desc[0].ddadr = &desc[1], BrEn = 1;
desc[0].dsadr = FETBL0;
desc[0].dtadr = 0x0000;
desc[0].dcmd = CmpEn=1, AddrMode = b01;
//Second Descriptor
//Error setting Descriptor, which stops the channel as
//(Full:Empty) != 0b00
//No data transferred. Stop interrupt triggered.
desc[1].ddadr = (Stop = 1);
desc[1].dsadr = ignored;
desc[1].dtadr = ignored;
desc[1].dcmd = Len=0;
//Third Descriptor
//Data transferring Descriptor
desc[2].ddadr = &desc[3];
desc[2].dsadr = SRC0;
desc[2].dtadr = TRG0;
desc[2].dcmd = Len=4K bytes;
//Fourth Descriptor
//FullEmpty table updating Descriptor
desc[3].ddadr = &desc[4];
desc[3].dsadr = FEUPDT;
desc[3].dtadr = FETBL0;
desc[3].dcmd = Len=4 bytes;
```

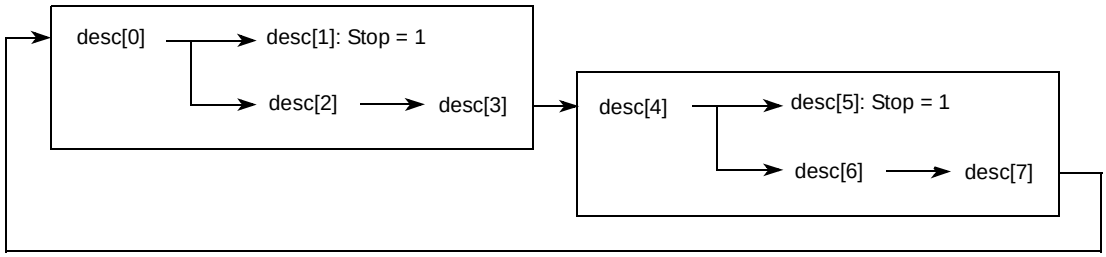
Second Descriptor Set

```
//First Descriptor
//Compare and Branch Descriptor modes enabled.
//No data transferred by this Descriptor.
//Source is indirectly addressed and target is directly addressed
//On a successful compare of &FETBL1 with 0x0000,
```


Copyright © 2009 Marvell

```
// Descriptor chain branches to desc[5] + 4*32bits, i.e desc[6].
//If Compare fails, then Descriptor chain jumps to desc[5].
//Desc[5] stops the channel as Full and Empty bits were not both 0.
desc[4].ddadr = &desc[5], BrEn = 1;
desc[4].dsadr = FETBL1;
desc[4].dtadr = 0x0000;
desc[4].dcmd = CmpEn=1, AddrMode = b01;
//Second Descriptor
//Error setting Descriptor, which stops the channel as
//(Full:Empty) != 0b00
//No data transferred. Stop interrupt triggered.
desc[5].ddadr = (Stop = 1);
desc[5].dsadr = ignored;
desc[5].dtadr = ignored;
desc[5].dcmd = Len=0;
//Third Descriptor
//Data transferring Descriptor
desc[6].ddadr = &desc[7];
desc[6].dsadr = SRC1;
desc[6].dtadr = TRG1;
desc[6].dcmd = Len=4K bytes;
//Fourth Descriptor
//FullEmpty table updating Descriptor
//Notice that the chain jumps back to desc[0].
//All ok, if software has loaded new data at the address pointed to by
//desc[0] and has also updated FETBL0
desc[7].ddadr = &desc[0];
desc[7].dsadr = FEUPDT;
desc[7].dtadr = FETBL1;
desc[7].dcmd = Len=4 bytes;
```

Figure 21: Descriptor Chain for Software Implementation of Full and Empty Bits



8.5 Register Descriptions

8.5.1 Register Summary

Table 113 summarizes the DMAC registers. The summary table includes the page number of the detailed description for each register for easy reference. The remainder of this section discusses each of these registers in detail.

Table 113: DMA Controller Registers

Address	Name	Description	Page
0x4000_0000	DCSR0	DMA Control / Status register for channel 0	337
0x4000_0004	DCSR1	DMA Control / Status register for channel 1	337
0x4000_0008	DCSR2	DMA Control / Status register for channel 2	337
0x4000_000C	DCSR3	DMA Control / Status register for channel 3	337
0x4000_0010	DCSR4	DMA Control / Status register for channel 4	337
0x4000_0014	DCSR5	DMA Control / Status register for channel 5	337
0x4000_0018	DCSR6	DMA Control / Status register for channel 6	337
0x4000_001C	DCSR7	DMA Control / Status register for channel 7	337
0x4000_0020	DCSR8	DMA Control / Status register for channel 8	337
0x4000_0024	DCSR9	DMA Control / Status register for channel 9	337
0x4000_0028	DCSR10	DMA Control / Status register for channel 10	337
0x4000_002C	DCSR11	DMA Control / Status register for channel 11	337
0x4000_0030	DCSR12	DMA Control / Status register for channel 12	337
0x4000_0034	DCSR13	DMA Control / Status register for channel 13	337
0x4000_0038	DCSR14	DMA Control / Status register for channel 14	337
0x4000_003C	DCSR15	DMA Control / Status register for channel 15	337
0x4000_0040	DCSR16	DMA Control / Status register for channel 16	337
0x4000_0044	DCSR17	DMA Control / Status register for channel 17	337
0x4000_0048	DCSR18	DMA Control / Status register for channel 18	337
0x4000_004C	DCSR19	DMA Control / Status register for channel 19	337
0x4000_0050	DCSR20	DMA Control / Status register for channel 20	337
0x4000_0054	DCSR21	DMA Control / Status register for channel 21	337
0x4000_0058	DCSR22	DMA Control / Status register for channel 22	337
0x4000_005C	DCSR23	DMA Control / Status register for channel 23	337
0x4000_0060	DCSR24	DMA Control / Status register for channel 24	337
0x4000_0064	DCSR25	DMA Control / Status register for channel 25	337
0x4000_0068	DCSR26	DMA Control / Status register for channel 26	337
0x4000_006C	DCSR27	DMA Control / Status register for channel 27	337

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_0070	DCSR28	DMA Control / Status register for channel 28	337
0x4000_0074	DCSR29	DMA Control / Status register for channel 29	337
0x4000_0078	DCSR30	DMA Control / Status register for channel 30	337
0x4000_007C	DCSR31	DMA Control / Status register for channel 31	337
0x4000_00A0	DALGN	DMA alignment register	345
0x4000_00A4	DPCSR	DMA Programmed IO control status register	346
0x4000_00E0	DRQSR0	DMA DREQStatus register (PXA32x Only)	336
		Reserved (PXA30x and PXA31x Only)	—
0x4000_00E4	—	Reserved	—
0x4000_00E8	—	Reserved	—
0x4000_00F0	DINT	DMA Interrupt register	344
0x4000_0100	DRCMR0	Request to Channel Map register for DREQ	328
0x4000_0104	—	Reserved	—
0x4000_0108	DRCMR2	Request to Channel Map register for SSP4 Receive request	328
0x4000_010C	DRCMR3	Request to Channel Map register for SSP4 Transmit request	328
0x4000_0110	DRCMR4	Request to Channel Map register for UART2 Receive request	328
0x4000_0114	DRCMR5	Request to Channel Map register for UART2 Transmit request.	328
0x4000_0118	DRCMR6	Request to Channel Map register for UART1 Receive request	328
0x4000_011C	DRCMR7	Request to Channel Map register for UART1 Transmit request	328
0x4000_0120	DRCMR8	Request to Channel Map register for AC97 microphone request	328
0x4000_0124	DRCMR9	Request to Channel Map register for AC97 modem Receive request	328
0x4000_0128	DRCMR10	Request to Channel Map register for AC97 modem Transmit request	328
0x4000_012C	DRCMR11	Request to Channel Map register for AC97 audio Receive request	328
0x4000_0130	DRCMR12	Request to Channel Map register for AC97 audio Transmit request	328
0x4000_0134	DRCMR13	Request to Channel Map register for SSP1 Receive request	328
0x4000_0138	DRCMR14	Request to Channel Map register for SSP1 Transmit request	328
0x4000_013C	DRCMR15	Request to Channel Map register for SSP2 Receive request	328
0x4000_0140	DRCMR16	Request to Channel Map register for SSP2 Transmit request	328
0x4000_0144	—	Reserved	—

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_0148	—	Reserved	—
0x4000_014C	DRCMR19	Request to Channel Map register for UART3 Receive request	328
0x4000_0150	DRCMR20	Request to Channel Map register for UART3 Transmit request	328
0x4000_0154	DRCMR21	Request to Channel Map register for MMC/SDIO 1 Receive request	328
0x4000_0158	DRCMR22	Request to Channel Map register for MMC/SDIO 1 Transmit request	328
0x4000_015C	—	Reserved	—
0x4000_0160	DRCMR24	Request to Channel Map register for USB endpoint 0 request	328
0x4000_0164	DRCMR25	Request to Channel Map register for USB endpoint A request	328
0x4000_0168	DRCMR26	Request to Channel Map register for USB endpoint B request	328
0x4000_016C	DRCMR27	Request to Channel Map register for USB endpoint C request	328
0x4000_0170	DRCMR28	Request to Channel Map register for USB endpoint D request	328
0x4000_0174	DRCMR29	Request to Channel Map register for USB endpoint E request	328
0x4000_0178	DRCMR30	Request to Channel Map register for USB endpoint F request	328
0x4000_017C	DRCMR31	Request to Channel Map register for USB endpoint G request	328
0x4000_0180	DRCMR32	Request to Channel Map register for USB endpoint H request	328
0x4000_0184	DRCMR33	Request to Channel Map register for USB endpoint I request	328
0x4000_0188	DRCMR34	Request to Channel Map register for USB endpoint J request	328
0x4000_018C	DRCMR35	Request to Channel Map register for USB endpoint K request	328
0x4000_0190	DRCMR36	Request to Channel Map register for USB endpoint L request	328
0x4000_0194	DRCMR37	Request to Channel Map register for USB endpoint M request	328
0x4000_0198	DRCMR38	Request to Channel Map register for USB endpoint N request	328
0x4000_019C	DRCMR39	Request to Channel Map register for USB endpoint P request	328
0x4000_01A0	DRCMR40	Request to Channel Map register for USB endpoint Q request	328
0x4000_01A4	DRCMR41	Request to Channel Map register for USB endpoint R request	328
0x4000_01A8	DRCMR42	Request to Channel Map register for USB endpoint S request	328
0x4000_01AC	DRCMR43	Request to Channel Map register for USB endpoint T request	328
0x4000_01B0	DRCMR44	Request to Channel Map register for USB endpoint U request	328
0x4000_01B4	DRCMR45	Request to Channel Map register for USB endpoint V request	328
0x4000_01B8	DRCMR46	Request to Channel Map register for USB endpoint W request	328

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_01BC	DRCMR47	Request to Channel Map register for USB endpoint X request	328
0x4000_01F8	DRCMR62	Request to Channel Map register for USIM 1 Receive request	328
0x4000_01FC	DRCMR63	Request to Channel Map register for USIM 1 Transmit request	328
0x4000_0200	DDADR0	DMA Descriptor Address register channel 0	329
0x4000_0204	DSADR0	DMA Source Address register channel 0	330
0x4000_0208	DTADR0	DMA Target Address register channel 0	331
0x4000_020C	DCMD0	DMA Command Address register channel 0	332
0x4000_0210	DDADR1	DMA Descriptor Address register channel 1	329
0x4000_0214	DSADR1	DMA Source Address register channel 1	330
0x4000_0218	DTADR1	DMA Target Address register channel 1	331
0x4000_021C	DCMD1	DMA Command Address register channel 1	332
0x4000_0220	DDADR2	DMA Descriptor Address register channel 2	329
0x4000_0224	DSADR2	DMA Source Address register channel 2	330
0x4000_0228	DTADR2	DMA Target Address register channel 2	331
0x4000_022C	DCMD2	DMA Command Address register channel 2	332
0x4000_0230	DDADR3	DMA Descriptor Address register channel 3	329
0x4000_0234	DSADR3	DMA Source Address register channel 3	330
0x4000_0238	DTADR3	DMA Target Address register channel 3	331
0x4000_023C	DCMD3	DMA Command Address register channel 3	332
0x4000_0240	DDADR4	DMA Descriptor Address register channel 4	329
0x4000_0244	DSADR4	DMA Source Address register channel 4	330
0x4000_0248	DTADR4	DMA Target Address register channel 4	331
0x4000_024C	DCMD4	DMA Command Address register channel 4	332
0x4000_0250	DDADR5	DMA Descriptor Address register channel 5	329
0x4000_0254	DSADR5	DMA Source Address register channel 5	330
0x4000_0258	DTADR5	DMA Target Address register channel 5	331
0x4000_025C	DCMD5	DMA Command Address register channel 5	332
0x4000_0260	DDADR6	DMA Descriptor Address register channel 6	329
0x4000_0264	DSADR6	DMA Source Address register channel 6	330

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_0268	DTADR6	DMA Target Address register channel 6	331
0x4000_026C	DCMD6	DMA Command Address register channel 6	332
0x4000_0270	DDADR7	DMA Descriptor Address register channel 7	329
0x4000_0274	DSADR7	DMA Source Address register channel 7	330
0x4000_0278	DTADR7	DMA Target Address register channel 7	331
0x4000_027C	DCMD7	DMA Command Address register channel 7	332
0x4000_0280	DDADR8	DMA Descriptor Address register channel 8	329
0x4000_0284	DSADR8	DMA Source Address register channel 8	330
0x4000_0288	DTADR8	DMA Target Address register channel 8	331
0x4000_028C	DCMD8	DMA Command Address register channel 8	332
0x4000_0290	DDADR9	DMA Descriptor Address register channel 9	329
0x4000_0294	DSADR9	DMA Source Address register channel 9	330
0x4000_0298	DTADR9	DMA Target Address register channel 9	331
0x4000_029C	DCMD9	DMA Command Address register channel 9	332
0x4000_02A0	DDADR10	DMA Descriptor Address register channel 10	329
0x4000_02A4	DSADR10	DMA Source Address register channel 10	330
0x4000_02A8	DTADR10	DMA Target Address register channel 10	331
0x4000_02AC	DCMD10	DMA Command Address register channel 10	332
0x4000_02B0	DDADR11	DMA Descriptor Address register channel 11	329
0x4000_02B4	DSADR11	DMA Source Address register channel 11	330
0x4000_02B8	DTADR11	DMA Target Address register channel 11	331
0x4000_02BC	DCMD11	DMA Command Address register channel 11	332
0x4000_02C0	DDADR12	DMA Descriptor Address register channel 12	329
0x4000_02C4	DSADR12	DMA Source Address register channel 12	330
0x4000_02C8	DTADR12	DMA Target Address register channel 12	331
0x4000_02CC	DCMD12	DMA Command Address register channel 12	332
0x4000_02D0	DDADR13	DMA Descriptor Address register channel 13	329
0x4000_02D4	DSADR13	DMA Source Address register channel 13	330
0x4000_02D8	DTADR13	DMA Target Address register channel 13	331

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_02DC	DCMD13	DMA Command Address register channel 13	332
0x4000_02E0	DDADR14	DMA Descriptor Address register channel 14	329
0x4000_02E4	DSADR14	DMA Source Address register channel 14	330
0x4000_02E8	DTADR14	DMA Target Address register channel 14	331
0x4000_02EC	DCMD14	DMA Command Address register channel 14	332
0x4000_02F0	DDADR15	DMA Descriptor Address register channel 15	329
0x4000_02F4	DSADR15	DMA Source Address register channel 15	330
0x4000_02F8	DTADR15	DMA Target Address register channel 15	331
0x4000_02FC	DCMD15	DMA Command Address register channel 15	332
0x4000_0300	DDADR16	DMA Descriptor Address register channel 16	329
0x4000_0304	DSADR16	DMA Source Address register channel 16	330
0x4000_0308	DTADR16	DMA Target Address register channel 16	331
0x4000_030C	DCMD16	DMA Command Address register channel 16	332
0x4000_0310	DDADR17	DMA Descriptor Address register channel 17	329
0x4000_0314	DSADR17	DMA Source Address register channel 17	330
0x4000_0318	DTADR17	DMA Target Address register channel 17	331
0x4000_031C	DCMD17	DMA Command Address register channel 17	332
0x4000_0320	DDADR18	DMA Descriptor Address register channel 18	329
0x4000_0324	DSADR18	DMA Source Address register channel 18	330
0x4000_0328	DTADR18	DMA Target Address register channel 18	331
0x4000_032C	DCMD18	DMA Command Address register channel 18	332
0x4000_0330	DDADR19	DMA Descriptor Address register channel 19	329
0x4000_0334	DSADR19	DMA Source Address register channel 19	330
0x4000_0338	DTADR19	DMA Target Address register channel 19	331
0x4000_033C	DCMD19	DMA Command Address register channel 19	332
0x4000_0340	DDADR20	DMA Descriptor Address register channel 20	329
0x4000_0344	DSADR20	DMA Source Address register channel 20	330
0x4000_0348	DTADR20	DMA Target Address register channel 20	331
0x4000_034C	DCMD20	DMA Command Address register channel 20	332

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_0350	DDADR21	DMA Descriptor Address register channel 21	329
0x4000_0354	DSADR21	DMA Source Address register channel 21	330
0x4000_0358	DTADR21	DMA Target Address register channel 21	331
0x4000_035C	DCMD21	DMA Command Address register channel 21	332
0x4000_0360	DDADR22	DMA Descriptor Address register channel 22	329
0x4000_0364	DSADR22	DMA Source Address register channel 22	330
0x4000_0368	DTADR22	DMA Target Address register channel 22	331
0x4000_036C	DCMD22	DMA Command Address register channel 22	332
0x4000_0370	DDADR23	DMA Descriptor Address register channel 23	329
0x4000_0374	DSADR23	DMA Source Address register channel 23	330
0x4000_0378	DTADR23	DMA Target Address register channel 23	331
0x4000_037C	DCMD23	DMA Command Address register channel 23	332
0x4000_0380	DDADR24	DMA Descriptor Address register channel 24	329
0x4000_034	DSADR24	DMA Source Address register channel 24	330
0x4000_0388	DTADR24	DMA Target Address register channel 24	331
0x4000_038C	DCMD24	DMA Command Address register channel 24	332
0x4000_0390	DDADR25	DMA Descriptor Address register channel 25	329
0x4000_0394	DSADR25	DMA Source Address register channel 25	330
0x4000_0398	DTADR25	DMA Target Address register channel 25	331
0x4000_039C	DCMD25	DMA Command Address register channel 25	332
0x4000_03A0	DDADR26	DMA Descriptor Address register channel 26	329
0x4000_03A4	DSADR26	DMA Source Address register channel 26	330
0x4000_03A8	DTADR26	DMA Target Address register channel 26	331
0x4000_03AC	DCMD26	DMA Command Address register channel 26	332
0x4000_03B0	DDADR27	DMA Descriptor Address register channel 27	329
0x4000_03B4	DSADR27	DMA Source Address register channel 27	330
0x4000_03B8	DTADR27	DMA Target Address register channel 27	331
0x4000_03BC	DCMD27	DMA Command Address register channel 27	332
0x4000_03C0	DDADR28	DMA Descriptor Address register channel 28	329

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_03C4	DSADR28	DMA Source Address register channel 28	330
0x4000_03C8	DTADR28	DMA Target Address register channel 28	331
0x4000_03CC	DCMD28	DMA Command Address register channel 28	332
0x4000_03D0	DDADR29	DMA Descriptor Address register channel 29	329
0x4000_03D4	DSADR29	DMA Source Address register channel 29	330
0x4000_03D8	DTADR29	DMA Target Address register channel 29	331
0x4000_03DC	DCMD29	DMA Command Address register channel 29	332
0x4000_03E0	DDADR30	DMA Descriptor Address register channel 30	329
0x4000_03E4	DSADR30	DMA Source Address register channel 30	330
0x4000_03E8	DTADR30	DMA Target Address register channel 30	331
0x4000_03EC	DCMD30	DMA Command Address register channel 30	332
0x4000_03F0	DDADR31	DMA Descriptor Address register channel 31	329
0x4000_03F4	DSADR31	DMA Source Address register channel 31	330
0x4000_03F8	DTADR31	DMA Target Address register channel 31	331
0x4000_03FC	DCMD31	DMA Command Address register channel 31	332
0x4000_0400 – 0x4000_1104	—	Reserved	—
0x4000_1108	DRCMR66	Request to Channel Map register for SSP3 Receive request	328
0x4000_110C	DRCMR67	Request to Channel Map register for SSP3 Transmit request	328
0x4000_1110 - 0x4000_1168	DRCMR68 – DRCMR90	Reserved	—
0x4000_116C	DRCMR91	Request to Channel Map register for USIM 2 Receive request	328
0x4000_1170	DRCMR92	Request to Channel Map register for USIM 2 Transmit request	328
0x4000_1174	DRCMR93	Request to Channel Map register for MMC/SDIO 2 Receive request	328
0x4000_1178	DRCMR94	Request to Channel Map register for MMC/SDIO 2 Transmit request	328
0x4000_117C	DRCMR95	Request to Channel Map register for AC97 surround Transmit request	328
0x4000_1180	DRCMR96	Request to Channel Map register for AC97 centre/LFE Transmit request	328
0x4000_1184	DRCMR97	Request to Channel Map register for NAND interface data Transmit & Receive request	328

Copyright © 2009 Marvell

Table 113: DMA Controller Registers (Continued)

Address	Name	Description	Page
0x4000_118C	DRCMR99	Request to Channel Map register for NAND interface command Transmit request	—
0x4000_1190	DRCMR46	Request to Channel Map register for MMC/SDIO 3 Receive request	328
0x4000_1194	DRCMR47	Request to Channel Map register for MMC/SDIO 3 Transmit request	328

8.5.2 DMA Request to Channel Map Register (DRCMRx)

DRCMRx, shown in Table 114, map the DMA request to a channel. They are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 114: DRCMR[0:63], DRCMR[64:99] Bit Definitions

Physical Address
0x4000_0100–0x4000_01FC
0x4000_1100–0x4000_118C

DRCMR[0:63]
DRCMR[64:99]

DMA Controller

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								MAPVLD	Reserved		CHLNUM				
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	Reserved	Reserved
7	R/W	MAPVLD	Map Valid Channel Defines whether the request is mapped to a valid channel. If the bit is set, the request is mapped to a valid channel indicated by CHLNUM. If the bit is cleared, the request is unmapped. This bit can also be used to mask the request. 0 = Request is unmapped 1 = Request is mapped to a valid channel indicated by CHLNUM.
6:5	—	Reserved	Reserved
4:0	R/W	CHLNUM	Channel Number Indicates the valid channel number if DRCMRx[MAPVLD] is set. Do not map two active requests to the same channel since it produces unpredictable results.

Copyright © 2009 Marvell

8.5.3 DMA Descriptor Address Registers (DDADRx)

DDADRx, shown in Table 115, contain the memory address of the next Descriptor for a channel. The bits in this register, except STOP, are undefined on power-up. STOP is cleared on power-up. The address must be aligned to a 128-bit (4-word) boundary. DDADRx must not contain the address of any other internal peripheral register or DMA register because this causes a bus error. The DDADRx register is reserved if the channel is performing a no-Descriptor-fetch transaction.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 115: DDADR[0:31] Bit Definitions

	Physical Address 0x4000_02x0–0x4000_03x0												DDADR[0:31]																DMA Controller			
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Descriptor Address																												Reserved		BREN	STOP
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:4	R/W	Descriptor Address	Descriptor address Contains address of next Descriptor.
3:2	—	Reserved	**Reserved**
1	R/W	BREN	Enable Descriptor Branch Works with DCSRx[CMPST] to determine which Descriptor is fetched next. If both BREN and CMPST are set, the DMAC fetches the next Descriptor from (DDADRx + 32 bytes). If either of the bits is cleared, DMAC fetches the next Descriptor from the DDADRx register. BREN is relevant only for Descriptor-fetch transactions (DCSRx[NODESCFETCH] = 0). See Figure 19 for details about the operational flow. 0 = Disable Descriptor branching. Fetch the next Descriptor from register DDADRx. 1 = Enable Descriptor branching.
0	R/W	STOP	Stop If STOP is cleared, a new Descriptor fetch based on register DDADRx and other control information is initiated after the current Descriptor is processed. 0 = Run channel 1 = Stop channel after completely processing this Descriptor and before fetching the next Descriptor (DCMDx[LEN] = 0)

8.5.4 DMA Source Address Register (DSADRx)

DSADRxs, shown in Table 116, are read-only for Descriptor-fetch transactions and read/write for no-Descriptor-fetch transactions. The registers contain the source address of the current Descriptor for a channel. The source address is the address of the internal peripheral, external peripheral, companion processor, or the address of a memory location. DSADRx cannot contain addresses of any other internal DMA registers because they cause a bus error.

If the source address is the address of a memory location and if the Alignment register is properly configured, the address can be aligned to a byte boundary (see Section 8.5.10, “DMA Alignment Register (DALGN)” (page 344)). Improper configuration of the Alignment register defaults the source address to an 8-byte boundary. Other restrictions on byte boundary alignment can apply for special DMA operations (see Section 8.4.4.4).

DSADRx are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 116: DSADR[0:31] Bit Definitions

Physical Address 0x4000_02x4–0x4000_03x4																																
DSADR[0:31]																																
DMA Controller																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SRCADDR																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:3	R/W	SRCADDR	Source address Source address of the internal peripheral, external peripheral, companion processor, or address of a memory location.
2	R/W	SRCADDR or Reserved	If SRCADDR is a memory location, an alignment register is configured. SRCADDR2 if SRCADDR is an internal peripheral. Reserved for all companion-processor or external peripheral transfers. Reserved for special DMA modes, such as compare modes.
1:0	R/W	SRCADDR or Reserved	Configured if SRCADDR is a memory location and alignment register. Reserved if SRCADDR is an internal peripheral. Reserved for all companion processor or external peripheral related transfers. Reserved for special DMA modes, such as compare modes.

Copyright © 2009 Marvell

8.5.5 DMA Target Address Registers (DTADRx)

DTADRx (Table 117) are read-only for Descriptor-fetch transfers and read/write for no-Descriptor-fetch transfers. The registers contain the target address of the current Descriptor for a channel. The target address is the address of the internal peripheral or the address of a memory location. DTADR cannot contain addresses of any other internal DMA registers because they cause a bus error.

The address can be aligned to a byte boundary (see Section 8.5.10, “DMA Alignment Register (DALGN)” (page 344)) if the target address is the address of a memory location and if the Alignment register is properly configured. Improper configuration of the Alignment register defaults the target address to an 8-byte boundary. Other restrictions on byte boundary alignment can apply for special DMA operations (see Section 8.4.4.4).

DTADRx are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 117: DTADR0–31 Bit Definitions

Physical Address 0x4000_02x8–0x4000_03x8	DTADR[0:31]	DMA Controller

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

Bits	Access	Name	Description
31:3	R/W	TRGADDR	Target Address Target address of the internal peripheral, external peripheral, companion processor, or address of a memory location.
2	R/W	TRGADDR or Reserved	Configured if TRGADDR is a memory location and alignment register. TRGADDR2 if TRGADDR is an internal peripheral. Reserved for all companion processor or external peripheral transfers. Reserved for special DMA modes, such as compare modes.
1:0	R/W	TRGADDR or Reserved	Configured if the target address is a memory location and alignment register. Reserved if target address is an internal peripheral. Reserved for all companion processor or external peripheral transfers. Reserved for special DMA modes, such as compare modes.

Copyright © 2009 Marvell

8.5.6 DMA Command Registers (DCMDx)

DCMx, shown in Table 118, are read-only registers for Descriptor-fetch transfers and read/write for no-Descriptor-fetch transfers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 118: DCMD[0:31] Bit Definitions

Physical Address 0x4000_02xC–0x4000_03xC | DCMD[0:31] | DMA Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	Reserved		CMPEN	Reserved	ADDRMODE	STARTIRQEN	ENDIRQEN	Reserved			SIZE		WIDTH		Reserved	LEN												
Reset	0	0	0	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	INCSRCADDR	Source Address Increment If the source address is an internal peripheral FIFO address or external I/O address, it is not incremented on each successive access. In these cases, INCSRCADDR must be cleared. 0 = Do not increment source address 1 = Increment source address
30	R/W	INCTRGADDR	Target Address Increment If the target address is an internal peripheral FIFO address or external I/O address, it is not incremented on each successive access. In these cases, INCTRGADDR must be cleared. 0 = Do not increment target address 1 = Increment target address
29	R/W	FLOWSRC	Source Flow Control The flow control of the source bit must be set if the source is an internal peripheral or external companion processor. Setting both the FLOWSRC and FLOWTRG bits causes unpredictable behavior. 0 = Do not wait for request signals associated with this channel. 1 = Wait for a request signal before initiating the data transfer.
28	R/W	FLOWTRG	Target Flow Control The flow control of the target bit must be set if the target is an internal peripheral or external companion processor. Setting both the FLOWSRC and FLOWTRG bits causes unpredictable behavior. 0 = Do not wait for request signals associated with this channel. 1 = Wait for a request signal before initiating the data transfer.
27:26	—	Reserved	Reserved

Copyright © 2009 Marvell

Table 118: DCMD[0:31] Bit Definitions (Continued)

Physical Address 0x4000_02xC–0x4000_03xC												DCMD[0:31]																			DMA Controller
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	Reserved		CMPEN	Reserved	ADDRMODE	STARTIRQEN	ENDIRQEN	Reserved			SIZE		WIDTH		Reserved	LEN											
Reset	0	0	0	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
25	R/W	CMPEN	Descriptor Compare Enable Must be cleared for normal DMA operations. Setting CMPEN enables the Descriptor-compare mode, in which the DMAC treats the current Descriptor as a special case and compares data that corresponds to the source and target fields. ADDRMODE is used to determine the addressing mode before the compare operation. 0 = DMA controller does not perform any address-compare operations. 1 = DMA controller recognizes the current Descriptor as a special case and compares data based on the source address and target address fields. If the compare is true, the channel DCSRx[CMPST] bit is set. If the compare is false, CMPST is cleared.
24	—	Reserved	Reserved
23	R/W	ADDRMODE or Reserved	Addressing Mode Controls the addressing mode for Descriptor comparison and is valid only in the Descriptor compare mode (CMPEN = 1). Reserved if CMPEN = 0. If CMPEN is set, the bits specify the addressing modes of the source address and target address fields. If either field contains an address, the DMAC fetches the data at that address and uses it for the compare operation. 0 = Source address field contains address, and target address field contains address 1 = Source address field contains address, and target address field contains data If DALGNx is clear, the lowest three bits of immediate data are forced to be 0 before comparison. If DALGNx is set, then the lowest three bits of immediate data are not forced to be 0 before comparison.
22	R/W	STARTIRQEN or Reserved	Start Interrupt Enable Indicates that the interrupt is enabled as soon as the Descriptor is loaded. In no-Descriptor-fetch transfers, this bit is reserved. 0 = Interrupt not triggered after Descriptor is loaded 1 = Set interrupt bit for that channel in the DINT[CHLINTR] when the Descriptor (4 words) for the channel is loaded

Table 118: DCMD[0:31] Bit Definitions (Continued)

Physical Address 0x4000_02xC–0x4000_03xC												DCMD[0:31]																			DMA Controller
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	Reserved		CMPEN	Reserved	ADDRMODE	STARTIRQEN	ENDIRQEN	Reserved			SIZE		WIDTH		Reserved	LEN											
Reset	0	0	0	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
21	R/W	ENDIRQEN	End Interrupt Enable 0 = Interrupt is not triggered when LENGTH decrements to zero. 1 = Set the DINT interrupt bit for the channel when LENGTH decrements to zero.
20:18	—	—	**Reserved**
17:16	R/W	SIZE	Maximum Burst Size Maximum burst size of each data transfer 0b00 = Reserved 0b01 = 8 bytes 0b10 = 16 bytes 0b11 = 32 bytes The size *must* be less than or equal to the serviced peripheral FIFO trigger threshold to handle the respective FIFO trailing bytes properly.
15:14	R/W	WIDTH or Reserved	Width of Internal Peripheral A reserved field for operations that do not involve internal peripherals, such as memory-to-memory moves and companion processor-related operations. WIDTH must be 0b00 for memory-to-memory moves or companion processor-related operations. 0b00 = Reserved for internal peripheral-related transactions 0b01 = 1 byte 0b10 = half -word (2 bytes) 0b11 = word (4 Bytes)
13	—	Reserved	Reserved

Copyright © 2009 Marvell

Table 118: DCMD[0:31] Bit Definitions (Continued)

Physical Address 0x4000_02xC–0x4000_03xC																															
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	Reserved		CMPEN	Reserved	ADDRMODE	STARTIRQEN	ENDIRQEN	Reserved			SIZE		WIDTH		Reserved	LEN											
Reset	0	0	0	0	?	?	0	0	0	0	0	?	?	?	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
12:0	R/W	LEN	Length of Transfer in Bytes When this bit is cleared, there are zero bytes for Descriptor-fetch transactions. LEN = 0 is an invalid setting for no Descriptor-fetch transactions. Programming LEN = 0 in the Descriptor-fetch mode when CMPEN is clear (normal data transfer mode) causes the channel immediately to discard the Descriptor after it is fetched from memory. If the Descriptor chain has more Descriptors, the channel fetches the next valid Descriptor. The channel stops if the Descriptor chain has no more Descriptors. The maximum transfer length is (8K–1) bytes. If the transfer is of the memory-to-memory type, the length of the transfer can be any value (except for the LEN = 0 restriction in no-Descriptor-fetch mode) up to a maximum of (8K –1) bytes. If the transfer involves an external peripheral (or a companion processor), the length of the transfer must be an integer multiple of the peripheral FIFO threshold (or water-mark). If the transfer involves any internal peripheral, the length of the transfer must be as follows: • For AC'97 and MMC/SDIO, LEN must be an integer multiple of 32 bytes. • For all other internal peripherals, LEN must be an integer multiple of the peripheral's sample width (DCMDx[WIDTH]). **Note:** LEN is ignored in the compare Descriptor mode (when CMPEN is set).

8.5.7 DREQ Status Register 0 (DRQSR0)

DRQSR0/, shown in Table 119, logs the number of pending requests made by an external companion processor on the corresponding DREQ<0> pin. The register reflects the status of a 5-bit counter that is controlled by the DMAC as follows:

- The DMAC increments the counter each time the external companion processor toggles the DREQ pin from low to high (positive edge trigger). The external companion processor must follow the rules outlined in Section 8.4.1.2.
- For a Write to an external peripheral, the DMAC decreases the counter after it completes the Write.
- For a Read from an external peripheral, the DMAC decreases the counter after it sends the corresponding Read request to the memory controller.

The external companion processor must not have more than 31 pending requests at a given time.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 119: DRQSR0 Bit Definitions

Physical Address 0x4000_00E0																																	
DRQSR0																																	
DMA Controller																																	
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																							CLR	Reserved			REQPEND				
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:9	—	Reserved	Reserved
8	W	CLR	Clearing Pending Request Setting this bit clears REQPEND and thereby clears all pending requests made by the external DMA request pin DREQ. Clearing this bit has no effect. CLR can be used for clearing the requests if software prematurely stops the channel mapped to DREQ. CLR must be set only after the mapped channel stops (DCSRx[STOPINTR] set). Clearing the requests of a running channel results in unpredictable behavior. 0 = No effect on REQPEND. 1 = Clear all pending requests registered in DRQSRx[REQPEND]
7:5	—	Reserved	Reserved
4:0	R	REQPEND	Request Pending Indicates the number of pending requests on DREQ.

Copyright © 2009 Marvell

8.5.8 DMA Channel Control/Status Registers (DCSRx)

DCSRx, shown in Table 120, are read/write registers containing the control and status bits for the channels. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 120: DCSR[0:31] Bit Definitions

Physical Address 0x4000_0000 –0x4000_007C	DCSR[0:31]	DMA Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIRQEN	MASKRUN	Reserved											CMPST	EORINTR	REQPEND	Reserved			RASINTR	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
31	R/W	RUN	Run Allows software to start or stop the channel. If RUN is cleared in the middle of the burst, the burst completes before the channel stops. Setting RUN starts the stopped channel. If the channel is in a Descriptor-fetch transfer and RUN is set before a valid Descriptor address is written to register DDADRx, no-Descriptor fetch occurs. RUN is reset as soon as it is cleared and when the channel stops normally. After the channel stops, STOPINTR is set. STOPINTR must be polled to read the channel status or to set STOPIRQEN and expect an interrupt after the channel stops. 0 = Stops the channel 1 = Starts the channel
30	R/W	NODESCFETCH	No-Descriptor Fetch Controls whether a channel has a Descriptor. If NODESCFETCH is set, the channel is considered a simple channel with no Descriptors. In this case, the DMA does not initiate Descriptor fetches when software sets RUN or when the byte count for the current transfer reaches zero. To program the channel for a no-Descriptor-fetch transfer, software must set this bit, then write to the individual DSADRx, DTADRx, and DCMDx registers for that channel. The DDADRx register is not used in this mode and must not be written. The RUN bit must be set to allow the channel to start the transfer. If DCSRx[NODESCFETCH] is cleared, the DMAC initiates Descriptor-fetches when software writes to the DDADRx register, when the byte count for the current transfer reaches zero. 0 = Descriptor-fetch transfer 1 = No Descriptor-fetch transfer

Table 120: DCSR[0:31] Bit Definitions (Continued)

Physical Address 0x4000_0000 –0x4000_007C	DCSR[0:31]	DMA Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIRQEN	MASKRUN	Reserved											CMPST	EORINTR	REQPEND	Reserved			RASINTR	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
29	R/W	STOPIRQEN	Stop Interrupt Enabled Enables the interrupt when STOPINTR is set. If STOPIRQEN is cleared, no interrupt is generated after the channel stops. STOPINTR is set after system reset de-assertion. Thus, if STOPINTR is set before the channel is started, an interrupt is generated. 0 = No interrupt if the channel is in uninitialized or stopped state. 1 = Interrupt enabled if the channel is in uninitialized or stopped state.
28	R/W	EORIRQEN	Setting the End-of-Receive Interrupt Enable Triggers an interrupt on an EOR condition. Clearing this bit does not generate an EOR-related interrupt. 0 = Interrupt not triggered even if EORINT is set. 1 = Enable Interrupt if EORINT is set.
27	R/W	EORJMPEN	Jump to the Next Descriptor on EOR Controls the Descriptor flow when the mapped peripheral signals an EOR to the DMAC. See Figure 22 for the behavior of the Descriptor during this condition. This control bit has no effect on the channel for no Descriptor-fetch transfers (NODESCFETCH is set). The DMAC completes the peripheral-to-memory data transfer on an EOR, regardless of EORJMPEN. 0 = DMAC continues to hold the current Descriptor and waits until the mapped peripheral makes another Receive request. 1 = DMAC jumps to the channel's next Descriptor on receiving an EOR from the mapped peripheral.
26	R/W	EORSTOPEN	Stop Channel on EOR Has no effect on the channel for no Descriptor-fetch transfers (NODESCFETCH is set). The DMAC completes the peripheral-to-memory data transfer on an EOR, regardless of the value of EORSTOPEN. Setting EORSTOPEN causes the DMAC to stop the channel on an EOR and set the corresponding STOPINTR bit. If STOPIRQEN is set when EORSTOPEN is set, an interrupt occurs. See Figure 22 for information on the behavior of the Descriptor. 0 = DMAC continues to hold the current Descriptor and waits until the mapped peripheral makes another Receive request. 1 = DMAC stops the channel that received an EOR from the mapped peripheral.

Copyright © 2009 Marvell

Table 120: DCSR[0:31] Bit Definitions (Continued)

Physical Address 0x4000_0000 –0x4000_007C	DCSR[0:31]	DMA Controller

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIRQEN	MASKRUN	Reserved											CMPST	EORINTR	REQPEND	Reserved			RASINTR	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
25	W	SETCMPST	Set Descriptor Compare Status Partially controls DCSRx[CMPST]. Setting this bit sets CMPST. Clearing SETCMPST has no effect on CMPST. Software can set CMPST even if the Descriptor is not configured in the compare mode (DCMDx[CMPEN] = 0). 0 = No effect on CMPST 1 = Set CMPST
24	W	CLRCMPST	Clear Descriptor Compare Status Partially controls CMPST. Setting CLRCMPST clears CMPST. Clearing CLRCMPST has no effect on CMPST. Software can set CMPST even if the Descriptor is not configured in the compare mode. 0 = No effect on CMPST 1 = Clear CMPST
23	R/W	RASIRQEN	Request after Channel Stopped Interrupt Enable 0 = Interrupt not triggered when a peripheral asserts a DMA request after the channel stops. 1 = Set interrupt bit for that channel in the DINT[CHLINTR] when a peripheral asserts a DMA request after the channel stops.
22	W	MASKRUN	Mask DCSR[RUN] during Programmed IO write to DCSR 0 = Software (programmed IO write) can modify RUN during a write transaction. 1 = Software (programmed IO write) cannot modify RUN during a write transaction.
21:11	—	Reserved	Reserved

Table 120: DCSR[0:31] Bit Definitions (Continued)

Physical Address 0x4000_0000 –0x4000_007C																																
DCSR[0:31]																																
DMA Controller																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIRQEN	MASKRUN	Reserved											CMPST	EORINTR	REQPEND	Reserved			RASINTR	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
10	R	CMPST	Descriptor Compare Status Indicates the most recent status of the source and target compare operation. CMPST is set on a successful compare of the source and target fields. An unsuccessful comparison clears CMPST. Refer to the description of DCMDx[ADDRMODE] for the various addressing modes used for this comparison. For details on the Descriptor compare mode, refer to DCMDx[CMPEN] in Table 118. The DMAC updates CMPST only in Descriptor compare mode (DCMDx[CMPEN] = 1). CMPST can be set and cleared by setting SETCMPST and CLRCMPST, respectively. If software attempts to concurrently set and clear CMPST by setting both SETCMPST and CLRCMPST, SETCMPST has higher precedence. Modifying this bit after RUN is set and the channel is actively running leads to faulty behavior of the Descriptor chain. The channel must be stopped before CMPST is set or cleared. 0 = Unsuccessful address compare in Descriptor-compare mode. 1 = Successful compare of the current Descriptor source and target addresses in Descriptor-compare mode.
9	Read/ Write 1 to clear	EORINT	End of Receive Interrupt Pertains only to internal peripherals and indicates the status of the mapped peripheral Receive data. EORINT is set after the DMAC reads out the last trailing sample from the peripheral Receive FIFO. Figure 22 illustrates the behavior of the Descriptor during this condition. 0 = DMA continues with the current Descriptor because the internal peripheral is still actively receiving data. 1 = Channel mapped internal peripheral has no data remaining in its Receive FIFO and has completed all Receive transactions. Refer to the description of EORJMPEN for the behavior of the DMAC during this condition. **Note:** EORINT must be cleared before a channel is restarted.

Copyright © 2009 Marvell

Table 120: DCSR[0:31] Bit Definitions (Continued)

Physical Address 0x4000_0000 –0x4000_007C	DCSR[0:31]	DMA Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIRQEN	MASKRUN	Reserved											CMPST	EORINTR	REQPEND	Reserved			RASINTR	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
8	R	REQPEND	Request Pending Indicates a pending request for the DMA channel. REQPEND is cleared for a channel if that channel has no pending request or the request has just been issued to the memory interface for a read or write from the external companion processor to memory. If DREQ assertion sets REQPEND and RUN is cleared to stop that channel, REQPEND and the internal registers that hold the DREQ assertion information do not remain set. If the channel is restarted, REQPEND must be reset by a Descriptor that transfers dummy data (for example, a memory-to-memory transfer from a temporary location to another temporary location). 0 = No request is pending for the channel. 1 = A request is pending for the channel.
7:5	—	Reserved	Reserved
4	R/W	RASINTR	Request after Channel Stops 0 = No interrupt 1 = Interrupt caused due to a request made by the peripheral after the respective channel stopped. This bit is reset by writing a value of 0b1 to it.
3	R	STOPINTR	Stop Interrupt Reflects the channel state. Software must clear STOPIRQEN to reset the interrupt. Reprogramming DDADRx and setting RUN restarts the channel. 0 = Channel is running. 1 = Channel is in uninitialized or stopped state. If STOPIRQEN is set, the DMAC generates an interrupt.
2	Read/ Write 1 to clear	ENDINTR	End Interrupt Indicates that the current Descriptor finished successfully and that DCMDx[ENDIRQEN] is set. 0 = No interrupt. 1 = Interrupt due to successful completion of the current transaction and DCMDx[LEN] = 0.

Table 120: DCSR[0:31] Bit Definitions (Continued)

Physical Address 0x4000_0000 –0x4000_007C	DCSR[0:31]	DMA Controller

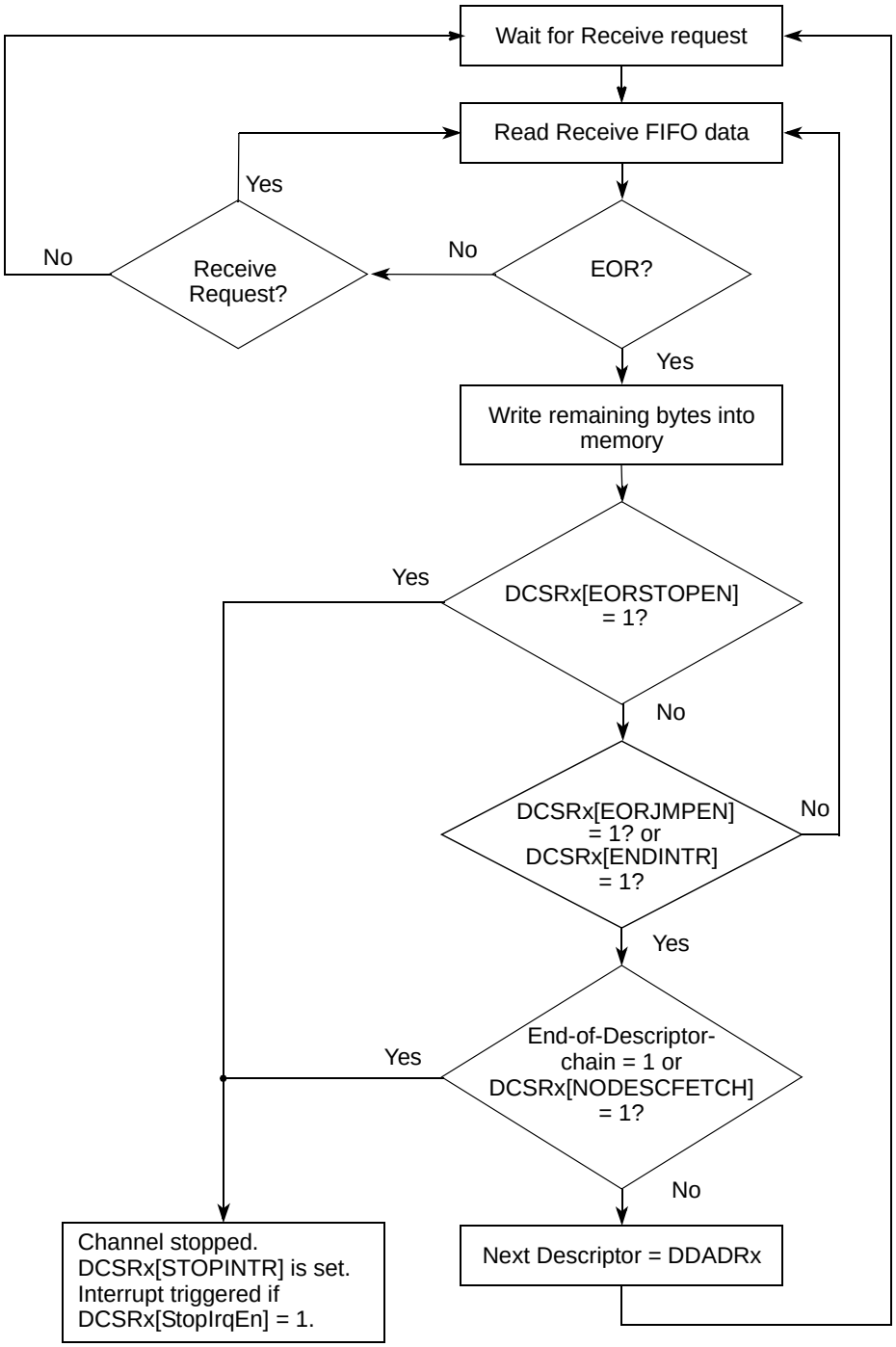
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIRQEN	MASKRUN	Reserved											CMPST	EORINTR	REQPEND	Reserved			RASINTR	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
1	Read/ Write 1 to clear	STARTINTR	Start Interrupt Indicates that the current Descriptor loaded successfully and that DCMDx[STARTIRQEN] is set. 0 = No interrupt. 1 = Interrupt due to successful Descriptor fetch.
0	Read/ Write 1 to clear	BUSERRINTR	Bus Error Interrupt Indicates an error during a data transfer on the internal bus. The error may be due to a bad Descriptor source or target address (any address that is in the non-burstable or reserved space can cause a bus error on the system bus).Only one error per channel is logged. The channel that caused the error is not updated at the end of the transfer and is not accessible until it is reprogrammed and the corresponding RUN bit is set. 0 = No interrupt. 1 = Interrupt caused by bus error.

Figure 22 shows the Descriptor behavior on End-of-Receive (EOR).

Copyright © 2009 Marvell

Figure 22: Descriptor Behavior on End-of-Receive (EOR)



Note: Fetching the NextDescriptor can be different if branching mode is enabled. Refer to the DDADRx[BREN] description for details.

8.5.9 DMA Interrupt Register (DINT)

DINT, shown in Table 121, logs the interrupt information for each channel. An interrupt is generated if one of the following conditions occurs:

- Any transaction error on the internal bus associated with the relevant channel.
- The current transfer finishes successfully and the DCMDx[ENDIRQEN] bit is set.
- The current Descriptor is loaded successfully and the DCMDx[STARTIRQEN] bit is set.
- DCSRx[STOPIRQEN] is set and the channel is in an uninitialized or stopped state.
- DCSRx[EORIRQEN] is set and DCSRx[EORINT] is set (EOR signaled by a peripheral).
- DCSRx[RASIRQEN] is set and the peripheral makes a DMA request after the channel stops.

All DMAC interrupts, except the one that corresponds to DCSRx[STOPINTR], are cleared by setting the corresponding interrupt bit in the DCSRx register.

Table 121: DINT Bit Definitions

Physical Address 0x4000_00F0																																	
DINT																																	
DMA Controller																																	
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHLINTR31	CHLINTR30	CHLINTR29	CHLINTR28	CHLINTR27	CHLINTR26	CHLINTR25	CHLINTR24	CHLINTR23	CHLINTR22	CHLINTR21	CHLINTR20	CHLINTR19	CHLINTR18	CHLINTR17	CHLINTR16	CHLINTR15	CHLINTR14	CHLINTR13	CHLINTR12	CHLINTR11	CHLINTR10	CHLINTR9	CHLINTR8	CHLINTR7	CHLINTR6	CHLINTR5	CHLINTR4	CHLINTR3	CHLINTR2	CHLINTR1	CHLINTR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:0	R	CHLINTRx	Channel Interrupt Indicates that DMA channel *x* has been interrupted. 0 = No interrupt 1 = Interrupt

8.5.10 DMA Alignment Register (DALGN)

DALGN, shown in Table 122, activates byte alignment for source and target addresses. Each bit in the register corresponds to a DMA channel. By default, during data transfers, the DMAC forces the least-significant three bits for all external addresses to zeros and the least significant two bits of all peripheral addresses to zeros.

Setting a channel-specific bit in DALGN causes the corresponding channel to access the complete user-specified address (none of the LSB bits of the address are forced to zeros). For example, if Channel 31 is programmed to transfer data involving a misaligned address, software must write 0b1 to Bit 31 of DALGN. Clearing a bit position in the DALGN register causes the DMAC to treat the corresponding channel as the default, a 64-bit aligned channel; the source and target addresses are forced to zeros as explained earlier. DALGN must be updated before setting DCSR[Run] and then must not be altered until the channel stops.

Copyright © 2009 Marvell

Table 122: DALGN Bit Definitions

Physical Address 0x4000_00A0																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DALGN31	DALGN30	DALGN29	DALGN28	DALGN27	DALGN26	DALGN25	DALGN24	DALGN23	DALGN22	DALGN21	DALGN20	DALGN19	DALGN18	DALGN17	DALGN16	DALGN15	DALGN14	DALGN13	DALGN12	DALGN11	DALGN10	DALGN9	DALGN8	DALGN7	DALGN6	DALGN5	DALGN4	DALGN3	DALGN2	DALGN1	DALGN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:0	R/W	DALGNx	Alignment Control for Channel x 0 = Source and target addresses of channel x are default aligned (internal peripherals default to 4-byte alignment; external bus addresses default to 8-byte alignment). 1 = Source and target addresses of channel x are as defined by the user (byte aligned).

8.5.11 DMA Programmed I/O Control Status Register (DPCSR)

DPCSR, shown in Table 123, activates and monitors posted Writes and split Reads on the system bus when software uses programmed I/O (PIO) instructions to access the peripheral address domain through the DMA bridge. Setting BRGSPLIT activates the following DMA behavior:

- The DMA split responds to the Read instruction if the PIO transaction is a Read from a peripheral address domain. The DMA bridge releases the system bus and uses microcoded instructions to read data from the peripheral bus. When the Read completes across the peripheral bus, the DMAC completes the split transaction by recapturing the system bus and completing the PIO Read transaction. The core stalls until the Read is returned because this is programmed IO. PIO transactions (Reads or Writes) that occur while the current PIO Read transaction is between the split response and the split completion are retried.
- The DMAC posts the Write instruction if the PIO transaction is a Write instruction to a peripheral address domain. The DMA bridge indicates to the system bus that the PIO Write is complete and then releases the system bus. The actual Write transaction is sent across the peripheral bus using microcoded instructions. If software requires a Write to complete on the peripheral bus before continuing, then software must write the address and immediately read the same address, which guarantees that the address updates before the core continues execution.

Clearing the BRGSPLIT bit deactivates the posted Write and split response behavior. A Write transaction on the system bus completes only after the Write is sent across the peripheral bus. The targeted address location is guaranteed to be updated by the time the transaction completes on the system bus. A Read transaction on the system bus completes only after the DMA bridge receives the data from across the peripheral bus. There are no split responses, split completions, or retries in this mode.

If software requires a Write to complete on the peripheral bus before continuing, software must write the address and then immediately read the same address to guarantee that the address updates before the core continues execution. Perform this Read-after-Write transaction to ensure that the processor is in a correct state before the core continues execution.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 123: DPCSR Bit Definitions

Physical Address 0x4000_00A4																																
DPCSR																																
DMA Controller																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BRGSPLIT	Reserved																														BRGBUSY
Reset	1	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Access	Name	Description
31	R/W	BRGSPLIT	Activate Posted Writes and Split Reads Must be modified only when BRGBUSY is cleared (no pending peripheral PIO transactions). Modifying this control bit when a PIO transaction is still pending can lead to unpredictable results and is therefore not recommended. The PIO transactions always complete in the order they are issued, regardless of the value of BRGSPLIT. 0 = Deactivate posted writes, split responses, split completions and retries. 1 = Activate posted writes, split responses, split completions and retries. BRGSPLIT is set by default (reset value).
30:1	—	Reserved	Reserved
0	R	BRGBUSY	Bridge Busy Status A status bit which, when set, indicates a pending PIO transaction across the peripheral bus. Any further PIO transactions on the system bus are retried when BRGBUSY is set. Software cannot modify this bit. When BRGBUSY is clear, there are no pending PIO transactions across the peripheral bus. A new PIO transaction is not retried in this case. 0 = No pending PIO transactions across peripheral bus. A new PIO transaction is not retried. 1 = Pending PIO transaction across peripheral bus. Any further PIO transactions on the system bus are retried.

Copyright © 2009 Marvell

9 Interrupt Controller

This chapter describes the interrupt controller, its modes of operation, and its associated registers. The interrupt controller controls the interrupt sources available to the processor and identifies the location of the interrupt source so that software can determine the first-level source of all interrupts. It also determines whether the interrupts cause an IRQ or an FIQ to occur and masks the interrupts.

Multiple secondary sources are usually mapped to a single primary source. For example, the DMA controller is a primary source of interrupts to the interrupt controller, with 32 possible secondary sources.

Each interrupt source generates either an IRQ or an FIQ. The setting that determines whether to generate an IRQ or an FIQ is called the *level* of the interrupt. The interrupt controller can be programmed to mask interrupts individually from the different sources. If an interrupt is masked, the controller does not cause the interrupt. The software can read registers in the interrupt controller, which identifies all the active IRQ and FIQ interrupts.

The controller assigns a unique priority to each primary source. It uses the assigned priority values to determine the highest priority peripheral when more than one interrupt is pending. Software can determine the peripheral ID with the highest active priority by reading a register in the controller.

The registers can be accessed either as coprocessor registers or as memory-mapped I/O registers. Coprocessor register mode has less access latency than memory-mapped I/O register-mode access.

There are no interrupt controller feature differences among the PXA32x processor, PXA31x processor, and PXA30x processor. Refer to each individual register for other operating differences.

9.1 Features

The interrupt controller unit has these features:

- Peripheral interrupt sources can be mapped to FIQ or IRQ.
- Each interrupt source can be independently enabled.
- Priority mechanism to indicate highest priority interrupt.
- Accessible through the coprocessor interface.
- Accessible as a memory-mapped peripheral for backwards compatibility.

9.2 Signal Descriptions

No external I/O signals are associated with the interrupt controller.

9.3 Operation

The Interrupt Controller Pending register (ICPR) has a bit for each of the peripherals (primary interrupt sources). Each active interrupt sets the corresponding bit. The Interrupt Controller IRQ Pending register (ICIP) and the Interrupt Controller FIQ Pending register (ICFP) identify the active, unmasked, pending interrupt sources that are causing IRQ- and FIQ-level interrupts, respectively. Interrupts are set at their source and masked or unmasked in the interrupt controller. The programmable Interrupt Controller Mask register (ICMR) is used to choose which interrupt source is masked. Masked interrupt sources do not cause an IRQ or FIQ interrupt to the core and only update the ICPR. The Interrupt Control Level register (ICLR) is used to choose whether an interrupt causes an IRQ or an FIQ.

Copyright © 2009 Marvell

Because more than one unmasked interrupt may be active at the same time, the processor must select the one with the highest priority. Software can determine which has the highest priority through the ICIP and ICFP or by reading the Interrupt Controller Highest Priority (ICHP) register at the controller that contains the peripheral ID with the highest priority value among the active unmasked interrupts.

The second level of the interrupt structure is represented by registers contained in the source device (the device that generated the first-level interrupt bit). Second-level interrupt status provides additional information about the interrupt and is used inside the interrupt-service routine. After it reads the first-level registers, software reads the registers in the device to determine the function that is causing the interrupt. In general, multiple second-level interrupts are ORed to produce a first-level interrupt bit. Interrupts are enabled inside the source device.

Interrupt-Priority registers (IPRs) specify the mapping between the different priority levels and the peripheral IDs. The interrupt controller uses these set values to prioritize the active unmasked interrupts and to update the ICHP register. The core can read the highest priority peripheral ID for both IRQ- and FIQ-level from the ICHP. The highest priority register is automatically updated with the peripheral ID, software is not required to examine each of the pending interrupts to determine which peripheral ID has the highest priority. ICHP is updated with both the highest priority IRQ and FIQ peripherals.

Most interrupt controller registers can be accessed using coprocessor-register access mode or memory-mapped-register access mode. The software can use either access method. Anomalies that can occur due to the varying access latencies in each mode must be considered. For example, a coprocessor-mapped read of a register that immediately follows a memory-mapped register write to the register may not yield the correct value. To ensure that the memory-mapped write has been performed properly, the memory-mapped register I/Os must be read immediately after the write.

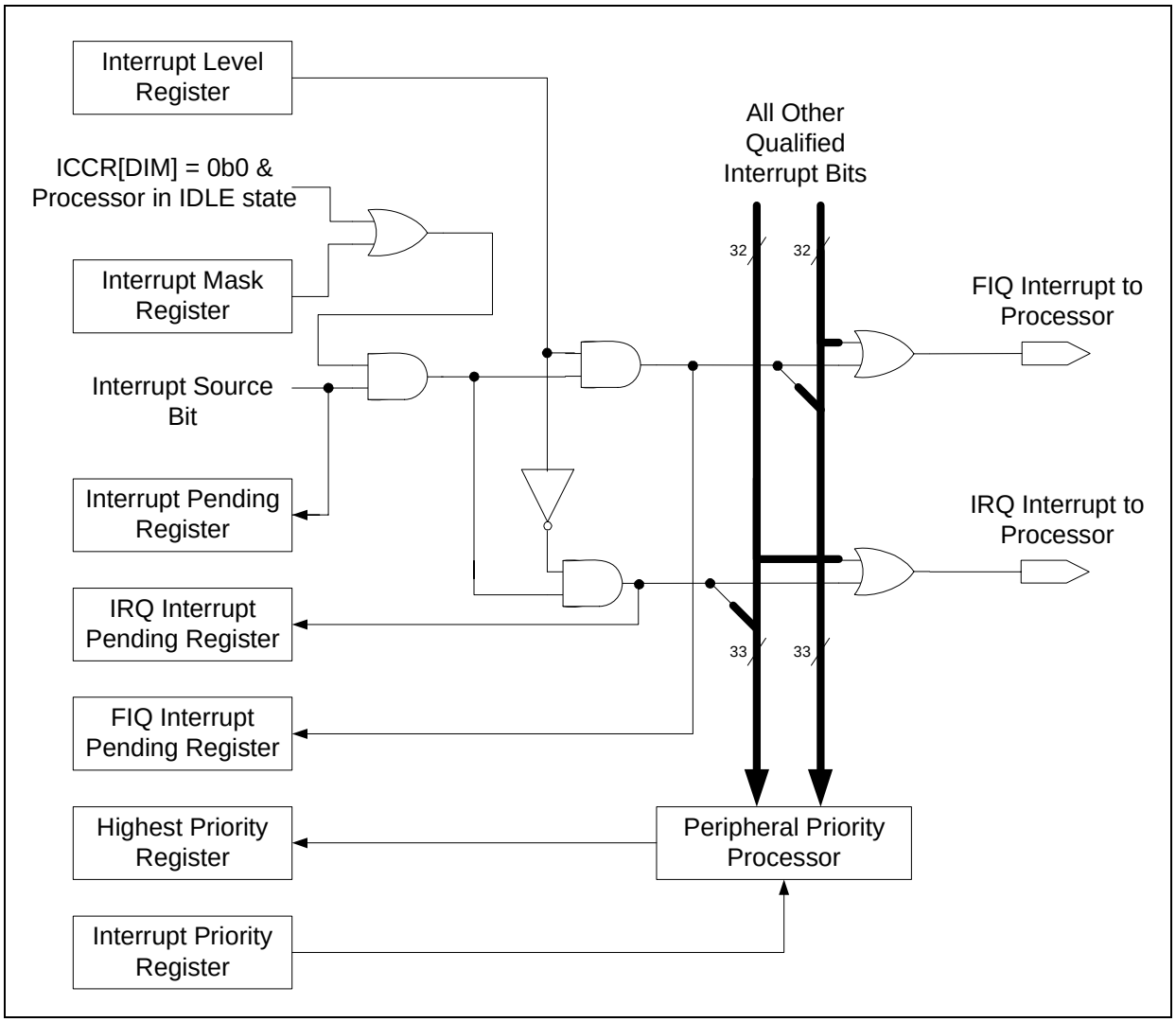
For coprocessor-register access mode, coprocessor space number six is used. Interrupt controller registers can only be accessed in supervisory mode. If any software process attempts to access the registers in the coprocessor mode, it receives an undefined access error. A software process that attempts to access the registers in memory-mapped-register access mode triggers a data-abort error. Software sets the enabling and access options for the coprocessor. Detailed information on setting access and enable options can be found in the *Intel XScale® Microarchitecture Programmer's Reference Manual.*

When the processor is in S0/D0/C1 state, the occurrence of any enabled interrupt causes the processor to resume operation. When ICCR disable-idle mode (DIM) is cleared (its default state), the interrupt controller ignores interrupt masks when the processor is in S0/D0/C1, and the occurrence of any interrupt triggers the processor to resume operation.

Figure 23 is a block diagram of the interrupt controller operation. The registers depicted in the figure are described in detail in this chapter.

Copyright © 2009 Marvell

Figure 23: Interrupt Controller Block Diagram



9.3.1 Accessing Interrupt Controller Registers

Most of the interrupt controller registers can be accessed through coprocessor registers. Accessing the interrupt controller registers through the coprocessor registers significantly reduces access times. The coprocessor-mapped registers must be accessed in supervisor mode only. Attempts to access the coprocessor-mapped interrupt coprocessor in user mode results in an undefined instruction exception.

Table 124 shows the Interrupt Controller registers and the coprocessor register numbers that correspond to them. The registers are mapped to the register space of Coprocessor 6.

Table 124: Interrupt Controller Register Mapping

Coprocessor 6 Register Number	Name	Access Mode	Description
CR0	ICIP	R	Interrupt Controller IRQ Pending register
CR1	ICMR	R/W	Interrupt Controller Mask register
CR2	ICLR	R/W	Interrupt Controller Level register
CR3	ICFP	R	Interrupt Controller FIQ Pending register
CR4	ICPR	R	Interrupt Controller Pending register
CR5	ICHP	R	Interrupt Controller Highest Priority register
CR6	ICIP2	R	Interrupt Controller IRQ Pending register 2
CR7	ICMR2	R/W	Interrupt Controller Mask register 2
CR8	ICLR2	R/W	Interrupt Controller Level register 2
CR9	ICFP2	R	Interrupt Controller FIQ Pending register 2
CR10	ICPR2	R	Interrupt Controller Pending register 2
Not Mapped to coprocessor	ICCR	R/W	Interrupt Controller Control register
Not Mapped to coprocessor	IPR0–IPR55	R/W	Interrupt Priority registers for interrupts 0 to 55

9.3.2 Enabling Coprocessor Access

To access the interrupt controller as Coprocessor 6, the corresponding coprocessor must be enabled in the core CPAR register. See the *Intel XScale® Core Users Guide* for details on enabling the coprocessors.

9.3.3 Accessing the Coprocessor

The example code sequences that follow assume that Coprocessor 6 is enabled:

To read from the coprocessor register, use an MRC instruction, as follows:

```
MRC P6, 0, Rd, CRn, C0, 0 ; Rd:Arm register and CRn:Coprocessor register
```

To write the interrupt controller register, use an MCR instruction, as follows:

```
LDR Rd, =0xdesired_val    ; Loading core register with desired value
MCR P6, 0, Rd, CRn, C0, 0 ; write the Arm register to CP6 register CRn
```

Note

All registers except IPR and ICCR are readable in coprocessor-access mode. Only ICMR, ICMR2, ICLR, and ICLR2 can be written in coprocessor-access mode.

Copyright © 2009 Marvell

9.3.4 Bit Positions and Peripheral IDs

Table 125 lists the primary sources for the interrupts. Each source can be associated with a number of second-level sources. Refer to the appropriate chapters for more detailed information on the second-level interrupt sources in each device. Each interrupt source has a peripheral ID. For the sake of simplicity, the peripheral ID is assigned according to bit positions. The interrupt priority registers (IPRs) are configured using these values as the peripheral ID.

Table 125: Summary of Bit Positions for Primary Sources

Bit Position	Source Module	Bit Field Description	Peripheral ID
IP[55]	MMC3 (PXA31X only)	MMC3 interrupt	55
	Not used (PXA32x and PXA30x Only)	—	
IP[54]	MVED (PXA31X only)	MVED Interrupt from Island AXI	54
	Not used (PXA32x and PXA30x Only)	—	
IP[53]	BPB2IMG (PXA31X only)	BPB2IMG Bridge interrupt from Island AXI	53
	Not used (PXA32x and PXA30x Only)	—	
IP[52]	Clock Controller	Processor CCU interrupts	52
IP[51]	DMC Interrupt	Dynamic memory controller	51
IP[50]	EXT_WAKEUP[1] (PXA32x Only)	External Wake-up source 1, refer to the Services Power Management Unit PECR register for more information on this interrupt.	50
	Not used (PXA31x and PXA30x Only)		
IP[49]	EXT_WAKEUP[0]	External Wake-up source 0, refer to the Services Power Management Unit PECR register for more information on this interrupt.	49
IP[48]	DMA_MVED (PXA31X only)	MVED DMA Interrupt Request	48
	Not used (PXA32x and PXA30x Only)	—	
IP[47]	SGP - Master PMU	SGP - Master PMU (services) interface module, refer to the Services Power Management Unit PMCR register for more information on this interrupt.	47
IP[46]	USB 2	USB 2.0 device	46
IP[45]	NAND Interface	NAND	45
IP[44]	1-Wire	1-Wire	44

Table 125: Summary of Bit Positions for Primary Sources (Continued)

Bit Position	Source Module	Bit Field Description	Peripheral ID
IP[43]	Not used	—	43
IP[42]	Not Used	—	42
IP[41]	MultiMediaCard 2	Multi-Media Card 2	41
IP[40]	Not used	—	40
IP[39]	Graphics controller	Graphics controller	39
IP[38]	USIM 2	USIM 2	38
IP[37]	Not used		37
IP[36]	TSI (PXA32x Only)	Touch screen	36
	Not used (PXA31x and PXA30x Only)	—	
IP[35]	Not used	—	35
IP[34]	Consumer Infrared	Consumer Infrared	34
IP[33]	Quick Capture Interface	Quick Capture Interface Interrupt	33
IP[32]	Not used	—	32
IP[31]	Real-Time Clock	RTC equals alarm register (from the Services RTC controller RTSR).	31
IP[30]		One-Hz clock occurred (from the Services RTC controller RTSR).	30
IP[29]	Operating System Timer	OS timer equals Match register 3	29
IP[28]		OS timer equals Match register 2	28
IP[27]		OS timer equals Match register 1	27
IP[26]		OS timer equals Match register 0	26
IP[25]	DMA Controller	DMA Channel service request	25
IP[24]	Synchronous Serial Port 1	SSP_1 service request	24
IP[23]	Flash Card Interface/MMC1	MMC/SDIO/Flash Card 1 status/Error detection	23
IP[22]	UART1	Transmit or receive error in UART1	22
IP[21]	UART2	Transmit or receive error in UART2	21
IP[20]	UART3	Transmit or receive error in UART3	20
IP[19]	Not Used	—	19

Copyright © 2009 Marvell

Table 125: Summary of Bit Positions for Primary Sources (Continued)

Bit Position	Source Module	Bit Field Description	Peripheral ID
IP[18]	I^2C	I^2C service request	18
IP[17]	LCD Controller	LCD controller service request	17
IP[16]	Synchronous Serial Port 2	SSP_2 service request	16
IP[15]	USIM Interface One	Smart Card Interface 1 status/error	15
IP[14]	AC '97	AC97 interrupt	14
IP[13]	Synchronous Serial Port 4	SSP_4 service request	13
IP[12]	PML	PML (Performance Monitor) interrupt	12
IP[11]	USB Device Controller (PXA32x and PXA30x Only)	USB Device Controller interrupt	11
	Not Used (PXA31x Only)	—	
IP[10]	GPIO	GPIO_x "OR" of the GPIO edge detect (except 0 and 1)	10
IP[9]		GPIO[1] detects an edge	9
IP[8]		GPIO[0] detects an edge	8
IP[7]	Operating System Timer	OS timer match register number 4–11	7
IP[6]	Power I^2C	Power I^2C interrupt	6
IP[5]	Not used	—	5
IP[4]	Keypad Controller	Keypad Controller interrupt	4
IP[3]	USB Host Controller	USB Host interrupt 1 (OHCI)	3
IP[2]		USB Host interrupt 2 (Non-OHCI)	2
IP[1]	Not used	—	1
IP[0]	Synchronous Serial Port 3	SSP_3 service request	0

9.4 Register Descriptions

Most interrupt controller registers are accessible through either their memory-mapped address or a coprocessor address. Table 126 lists the memory-mapped interrupt controller registers by physical address, and Table 127 lists the registers by coprocessor address. For easy reference, the summary tables include the page number of the detailed description for each register. The remainder of this section discusses each of these registers in detail.

Table 126: Interrupt Controller Register Summary by Physical Addresses

Physical Address	Name	Description	Page
0x40D0_0000[1]	ICIP	Interrupt Controller IRQ Pending register	364
0x40D0_0004[1]	ICMR	Interrupt Controller Mask register	382
0x40D0_0008[1]	ICLR	Interrupt Controller Level register	391
0x40D0_000C[1]	ICFP	Interrupt Controller FIQ Pending register	373
0x40D0_0010[1]	ICPR	Interrupt Controller Pending register	355
0x40D0_0014	ICCR	Interrupt Controller Control register	399
0x40D0_0018[1]	ICHP	Interrupt Controller Highest Priority register	401
0x40D0_001C–0x40D0_0098	IPR0 – IPR55	Interrupt Priority registers for interrupts 0 – 55	400
0x40D0_009C[1]	ICIP2	Interrupt Controller IRQ Pending register 2	363
0x40D0_00A0[1]	ICMR2	Interrupt Controller Mask register 2	382
0x40D0_00A4[1]	ICLR2	Interrupt Controller Level register 2	395
0x40D0_00A8[1]	ICFP2	Interrupt Controller FIQ Pending register 2	373
0x40D0_00AC[1]	ICPR2	Interrupt Controller Pending register 2	355
0x40D0_0108–0x40DF_FFFF	—	Reserved	
1 These registers are also mapped to coprocessors. Refer to Table 127 for the coprocessor address.			

Table 127: Interrupt Controller Register Summary by Coprocessor Addresses

Coprocessor Address	Name	Description	Page
CP6, CR0	ICIP	Interrupt Controller IRQ Pending register	364
CP6, CR1	ICMR	Interrupt Controller Mask register	382
CP6, CR2	ICLR	Interrupt Controller Level register	391
CP6, CR3	ICFP	Interrupt Controller FIQ Pending register	373
CP6, CR4	ICPR	Interrupt Controller Pending register	355
CP6, CR5	ICHP	Interrupt Controller Highest Priority register	401
CP6, CR6	ICIP2	Interrupt Controller IRQ Pending register 2	363
CP6, CR7	ICMR2	Interrupt Controller Mask register 2	382

Copyright © 2009 Marvell

Table 127: Interrupt Controller Register Summary by Coprocessor Addresses (Continued)

Coprocessor Address	Name	Description	Page
CP6, CR8	ICLR2	Interrupt Controller Level register 2	395
CP6, CR9	ICFP2	Interrupt Controller FIQ Pending register 2	373
CP6, CR10	ICPR2	Interrupt Controller Pending register 2	355

9.4.1 Interrupt Controller Pending Registers (ICPR and ICPR2)

The ICPR and ICPR2 are two 32-bit read-only registers that show all pending interrupts in the system. The contents are not affected by the state of the Mask registers (ICMR and ICMR2). Both the IRQ-level and FIQ-level pending interrupts are read as set in this register. The register is cleared during reset. See Table 128 and Table 129 for details.

Several units have more than one source per interrupt signal. When one of these units signals an interrupt, software identifies the interrupt through the registers (ICPR and ICPR2) or by reading the ICHP that contains the peripheral ID with the interrupt that has highest active, unmasked priority. Software then reads the status register of the interrupting unit to identify which source in the unit signalled the interrupt. For all interrupts with only one corresponding source, the interrupt-handler routine must use only the interrupt controller registers to identify the exact cause of the interrupt. After the interrupt from a source is serviced, the corresponding bit the corresponding bit in ICPR is automatically cleared.

Table 128: ICPR Bit Definitions

Physical Address: 0x40D0_0010 **ICPR** **Interrupt Controller**
Coprocessor Register: CP6, CR4

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
31	R	RTC_AL	Real-Time Clock Alarm 0 = No RTC equals alarm register interrupt. 1 = RTC equals alarm register interrupt.
30	R	RTC_HZ	One-Hz Clock 0 = No one Hz clock TIC. 1 = One Hz clock TIC.

Table 128: ICPR Bit Definitions (Continued)

Physical Address: 0x40D0_0010 **ICPR** **Interrupt Controller**
Coprocessor Register: CP6, CR4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
29	R	OST_3	OS Timer 3 0 = OS timer does not equal Match register 3. 1 = OS timer equals Match register 3.
28	R	OST_2	OS Timer 2 0 = OS timer does not equal Match register 2. 1 = OS timer equals Match register 2.
27	R	OST_1	OS Timer 1 0 = OS timer does not equal Match register 1. 1 = OS timer equals Match register 1.
26	R	OST_0	OS Timer 0 0 = OS timer does not equal Match register 0. 1 = OS timer equals Match register 0.
25	R	DMAC	DMA Controller 0 = No DMA channel service request. 1 = DMA Channel service request.
24	R	SSP1	SSP 1 0 = SSP 1 has not requested service. 1 = SSP 1 has requested service.
23	R	MMC	MultiMediaCard 1 0 = No flash card status changed/error detected. 1 = Flash Card status change/error detected.
22	R	UART1	UART1 0 = No transmit or receive error in UART1. 1 = Transmit or receive error in UART1.
21	R	UART2	UART2 0 = No transmit or receive error in UART2. 1 = Transmit or receive error in UART2.

Copyright © 2009 Marvell

Table 128: ICPR Bit Definitions (Continued)

Physical Address: 0x40D0_0010 **ICPR** **Interrupt Controller**
Coprocessor Register: CP6, CR4

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
20	R	UART3	UART3 0 = No transmit or receive error in UART3. 1 = Transmit or receive error in UART3.
19	—	—	Reserved
18	R	I2C	I^2C 0 = No I^2C service request. 1 = I^2C service request.
17	R	LCD	LCD 0 = No LCD controller service request. 1 = LCD controller service request.
16	R	SSP2	SSP2 0 = No SSP 2 service request. 1 = SSP 2 service request.
15	R	USIM1	USIM 0 = No smart card interface status changes/error. 1 = Smart card interface status changes/error.
14	R	AC97	AC97 0 = No AC ‘97 interrupt. 1 = AC ‘97 interrupt.
13	R	I2S	SSP4 0 = No I^2S interrupt. 1 = I^2S interrupt.
12	R	PML	PML 0 = No PML (performance monitor) interrupt. 1 = PML (performance monitor) interrupt.

Table 128: ICPR Bit Definitions (Continued)

Physical Address: 0x40D0_0010 **ICPR** **Interrupt Controller**
Coprocessor Register: CP6, CR4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
11	R	USBD	USB Device Controller (PXA32x and PXA30x Only) 0 = No USB Device Controller interrupt. 1 = USB Device Controller interrupt.
	—	—	Reserved (PXA31x Only)
10	R	GPIO_X	GPIO_X 0 = No GPIO_x edge (other than GPIO_0 or GPIO_1) detected. 1 = GPIO_x edge (other than GPIO_0 or GPIO_1) detected.
9	R	GPIO_1	GPIO_1 0 = No GPIO<1> edge detected. 1 = GPIO<1> edge detected.
8	R	GPIO_0	GPIO_0 0 = No GPIO<0> edge detected. 1 = GPIO<0> edge detected.
7	R	OST_4_11	OS Timers 4–11 0 = No OS timer match 4–11. 1 = OS timer match 4–11.
6	R	PWR_I2C	Power I^2C 0 = No I^2C Power Unit interrupt. 1 = I^2C Power Unit interrupt.
5	—	—	Reserved
4	R	KEYPAD	KEYPAD 0 = No Keypad Controller interrupt. 1 = Keypad Controller interrupt.

Copyright © 2009 Marvell

Table 128: ICPR Bit Definitions (Continued)

Physical Address: **ICPR** **Interrupt Controller**
0x40D0_0010
Coprocessor Register:
CP6, CR4

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
3	R	USBH1	USB Host Interrupt 1 0 = No USB Host interrupt 1 (OHCI). 1 = USB Host interrupt 1 (OHCI) **NOTE:** When an OHCI interrupt occurs software must read the UHC Interrupt Status (UHCINTS) Register to determine the source of the interrupt.
2	R	USBH2	USB Host Interrupt 2 0 = No USB Host interrupt 2 (Non-OHCI) 1 = USB Host interrupt 2 (Non-OHCI) **NOTE:** When a Non-OHCI interrupt occurs software must read the UHC Status (UHCSTAT) Register to determine the source of the interrupt.
1	—	—	Reserved
0	R	SSP3	SSP 3 0 = No SSP3 service request. 1 = SSP3 service request.

Table 129: ICPR2 Bit Definitions

Physical Address 0x40D0_00AC **ICPR2** **Interrupt Controller**
Coprocessor Register: CP6, CR10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Resreved			CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved			CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
31:24	—	—	Reserved
23	R	MMC3	0 = MMC3 has NOT requested service. (PXA31x Only) 1 = MMC3 has requested service.
	—	—	Reserved (PXA32x and PXA30x Only)
22	R	MVED	0 = MVED has NOT requested service.(PXA31x Only) 1 = MVED has requested service.
	—	—	Reserved (PXA32x and PXA30x Only)
21	R	BPB2IMG	0 = BPB2IMG has NOT requested service.(PXA31x Only) 1 = BPB2IMG has requested service.
	—	—	Reserved (PXA32x and PXA30x Only)
20	R	ACCU	0 = No Application Subsystem Clock unit service request. 1 = Application Subsystem Clock unit service request.
19	R	DMC	0 = No dynamic memory controller service request. 1 = Dynamic memory controller service request.

Copyright © 2009 Marvell

Table 129: ICPR2 Bit Definitions (Continued)

Physical Address **ICPR2** **Interrupt Controller**
0x40D0_00AC
Coprocessor Register:
CP6, CR10

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Resreved			CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved			CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
18	R	WAKEUP1	0 = No EXT_WAKEUP<1> service request. 1 = EXT_WAKEUP<1> service request. **NOTE:** (PXA32x Only)
	R	—	Reserved (PXA31x and PXA30x Only)
17	R	WAKEUP0	0 = No EXT_WAKEUP<0> service request. 1 = EXT_WAKEUP<0> has requested service.
16	R	DMA_MVED	0 = DMA_MVED has NOT requested service. 1 = DMA_MVED controller has requested service. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
15	R	SPMU	0 = No SPMU controller service request. 1 = SPMU controller service request.
14	R	USB 2	0 = No USB 2.0 Device Controller service request. 1 = USB 2.0 Device Controller service request.
13	R	NAND INF	0 = No NAND controller service request. 1 = NAND controller service request.
12	R	ONE WIRE	0 = No one-wire controller service request. 1 = One-wire controller service request.

Table 129: ICPR2 Bit Definitions (Continued)

Physical Address 0x40D0_00AC **ICPR2** **Interrupt Controller**
Coprocessor Register: CP6, CR10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Resreved			CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved			CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
11:10	—	—	Reserved
9	R	MMC 2	0 = No MMC2 controller service request. 1 = MMC2 controller service request.
8	—	—	Reserved
7	R	GRAPHICS	0 = No graphics controller service request. 1 = Graphics controller service request.
6	R	USIM 2	0 = No USIM2 controller service request. 1 = USIM2 service request.
5	—	—	Reserved
4	R	TSI	0 = The touchscreen interface has NOT requested service. (PXA32x Only) 1 = The touchscreen interface has requested service.
	—	—	Reserved (PXA31x and PXA30x Only)
3	—	—	Reserved
2	R	CONSUMER IR	0 = No consumer IR controller service request. 1 = Consumer IR controller service request.

Copyright © 2009 Marvell

Table 129: ICPR2 Bit Definitions (Continued)

Physical Address 0x40D0_00AC Coprocessor Register: CP6, CR10	ICPR2	Interrupt Controller

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Resreved			CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved		MMC 2	Reserved	GRAPHICS	USIM 2	Reserved			CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
1	R	CIF	Capture Interface 0 = No Intel Quick-Capture Interface controller service request. 1 = Intel Quick-Capture Interface controller service request.
0	—	—	Reserved

9.4.2 Interrupt Controller IRQ Pending Registers (ICIP and ICIP2)

ICIP and ICIP2 have one bit per interrupt source. A bit is set when the corresponding peripheral has a pending unmasked IRQ interrupt waiting to be served. In general, software reads the status registers of the interrupting unit for information on how to service the interrupt. Interrupt bits in the ICPR and ICPR2 are read-only and represent the logical OR of the status bits for a given interrupt in the source unit. After an interrupt is serviced, the handler clears the pending interrupt at the source by writing to the required status bit. For details, refer to the respective peripheral chapters.

Clearing the interrupt-status bit at the source automatically clears the corresponding ICIP (or ICIP2) bit. Table 130 and Table 131 show the bit locations that correspond to the independent interrupt-pending status flags in the ICIP and ICIP2. An interrupting source creates an IRQ if the corresponding bit in the ICLR or ICLR2 is cleared.

Table 130: ICIP Bit Definitions

Physical Address: 0x40D0_0000 **ICIP** **Interrupt Controller**
Coprocessor Register: CP6, CR0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
31	R	RTC_AL	Real-Time Clock Alarm 0 = One of the requirements for setting the bit has not been met. 1 = RTC equals Alarm register, interrupt level<31> = 0, and either mask bit 31 = 1 or DIM bit = 0.
30	R	RTC_HZ	One Hz Clock 0 = One of the requirements for setting the bit has not been met. 1 = One Hz clock TIC occurred, interrupt level<30> = 0, and either mask bit 30 = 1 or DIM bit = 0.
29	R	OST_3	OS Timer 3 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals match register 3, interrupt level<29> = 0, and either mask bit 29 = 1 or DIM bit = 0.
28	R	OST_2	OS Timer 2 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals Match register 2, interrupt level<28> = 0, and either mask bit 28 =1 or DIM bit = 0.
27	R	OST_1	OS Timer 1 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals Match register 1, interrupt level<27> = 0, and either mask bit<27>=1 or DIM bit = 0.
26	R	OST_0	OS Timer 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals Match register 0, interrupt level 26 = 0, and either mask bit 26 = 1 or DIM bit = 0.
25	R	DMAC	DMA Controller 0 = One of the requirements for setting the bit has not been met. 1 = DMA channel service request has occurred, interrupt level 25 = 0, and either mask bit 25 = 1 or DIM bit = 0.

Copyright © 2009 Marvell

Table 130: ICIP Bit Definitions (Continued)

Physical Address: 0x40D0_0000 **ICIP** **Interrupt Controller**
Coprocessor Register: CP6, CR0

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
24	R	SSP1	SSP 1 0 = One of the requirements for setting the bit has not been met. 1 = SSP 1 service request has occurred, interrupt level<24>=0, and either mask bit<24>=1 or DIM bit = 0.
23	R	MMC1	MultiMediaCard 0 = One of the requirements for setting the bit has not been met. 1 = Flash card status has changed or any error has been detected, interrupt level 23 = 0, and either mask bit 23 = 1 or DIM bit = 0.
22	R	UART1	UART1 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in UART1, interrupt level 22 = 0, and either mask bit 22 = 1 or DIM bit = 0.
21	R	UART2	UART2 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in UART2, interrupt level 21 = 0, and either mask bit 21 = 1or DIM bit = 0.
20	R	UART3	UART3 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in UART3, interrupt level 20 = 0, and either mask bit 20 = 1 or DIM bit = 0.
19	—	—	Reserved
18	R	I2C	I^2C 0 = One of the requirements for setting the bit has not been met. 1 = I^2C service request has occurred, interrupt level 18 = 0, and either mask bit 18 = 1 or DIM bit = 0.
17	R	LCD	LCD 0 = One of the requirements for setting the bit has not been met. 1 = LCD controller service request has occurred, interrupt level 17 = 0, and either mask bit 17 = 1 or DIM bit = 0.

Table 130: ICIP Bit Definitions (Continued)

Physical Address: 0x40D0_0000 **ICIP** **Interrupt Controller**
Coprocessor Register: CP6, CR0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
16	R	SSP2	SSP 2 0 = One of the requirements for setting the bit has not been met. 1 = SSP 2 service requests has occurred, interrupt level 16 = 0, and either mask bit 16 = 1 or DIM bit = 0.
15	R	USIM1	USIM1 0 = One of the requirements for setting the bit has not been met. 1 = Smart card interface status/error has occurred, interrupt level 15 = 0, and either mask bit 15 = 1 or DIM bit = 0.
14	R	AC97	AC97 0 = One of the requirements for setting the bit has not been met. 1 = AC97 interrupt has occurred, interrupt level 14 = 0, and either mask bit 14 = 1, or DIM bit = 0.
13	R	SSP4	SSP4 0 = One of the requirements for setting the bit has not been met. 1 = I^2S interrupt has occurred, interrupt level 13 = 0, and either mask bit 13 = 1 or DIM bit = 0.
12	R	PML	Performance Monitor Unit 0 = One of the requirements for setting the bit has not been met. 1 = PML (performance monitor) interrupt has occurred, interrupt level<12> = 0, and either mask bit<12> = 1 or DIM bit = 0.
11	R	USBD	USB Device Controller (PXA32x and PXA30x Only) 0 = One of the requirements for setting the bit has not been met. 1 = USB Device Controller interrupt has occurred, interrupt level<11> = 0, and either mask bit<11> = 1 or DIM bit = 0.
	—	—	Reserved (PXA31x Only)
10	R	GPIO_x	GPIO_x 0 = One of the requirements for setting the bit has not been met. 1 = GPIO_x (other than GPIO_0 or GPIO_1) edge detect = 1, interrupt level 10 = 0, and either mask bit 10 = 1 or DIM bit = 0.

Copyright © 2009 Marvell

Table 130: ICIP Bit Definitions (Continued)

Physical Address: 0x40D0_0000 **ICIP** **Interrupt Controller**
Coprocessor Register: CP6, CR0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
9	R	GPIO_1	GPIO_1 0 = One of the requirements for setting the bit has not been met. 1 = GPIO<1> has detected an edge, interrupt level 9 = 0, and either mask bit 9 = 1 or DIM bit = 0.
8	R	GPIO_0	GPIO_0 0 = One of the requirements for setting the bit has not been met. 1 = GPIO<0> has detected an edge, interrupt level 8 = 0, and either mask bit 8 = 1 or DIM bit = 0.
7	R	OST_4_11	OS Timer 4-11 0 = One of the requirements for setting the bit has not been met. 1 = OS timer match 4-11 has occurred, interrupt level 7 = 0, and either mask bit 7 = 1 or DIM bit = 0.
6	R	PWR_I2C	Power I^2C 0 = One of the requirements for setting the bit has not been met. 1 = I^2C power unit interrupt has occurred, interrupt level 6 = 0, and either mask bit 6 = 1 or DIM bit = 0.
5	—	—	Reserved
4	R	KEYPAD	Keypad 0 = One of the requirements for setting the bit has not been met. 1 = Keypad controller interrupt has occurred, interrupt level 4 = 0, and either mask bit 4 = 1 or DIM bit = 0.
3	R	USBH1	USB Host Interrupt 1 0 = One of the requirements for setting the bit has not been met.(OHCI). 1 = USB Host interrupt 1 (OHCI) has occured, interrupt level 3 = 0, and either mask bit 3 = 1 or DIM bit = 0. **NOTE:** When an OHCI interrupt occurs software must read the UHC Interrupt Status (UHCINTS) Register to determine the source of the interrupt.

Table 130: ICIP Bit Definitions (Continued)

Physical Address: 0x40D0_0000 **ICIP** **Interrupt Controller**
Coprocessor Register: CP6, CR0

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
2	R	USBH2	USB Host Interrupt 2 0 = One of the requirements for setting the bit has not been met. (Non-OHCI) 1 = USB Host interrupt 2 (Non-OHCI) has occured, interrupt level 2 = 0, and either mask bit 2 = 1 or DIM bit = 0. **NOTE:** When a Non-OHCI interrupt occurs software must read the UHC Status (UHCSTAT) Register to determine the source of the interrupt.
1	—	—	Reserved
0	R	SSP3	SSP 3 0 = One of the requirements for setting the bit has not been met. 1 = SSP 3 service request has occurred, interrupt level 0 = 0, and either mask bit 0 = 1 or DIM bit = 0.

Copyright © 2009 Marvell

Table 131: ICIP2 Bit Definitions

Physical Address **ICIP2** **Interrupt Controller**
0x40D0_009C
Coprocessor Register: CP6, CR6

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
31:24	—	—	Reserved
23	R	MMC3	0 = MMC3 has NOT requested service. 1 = MMC3 has requested service. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
22	R	MVED	0 = MVED has NOT requested service. 1 = MVED has requested service. **NOTE:** (PXA31x Only)
	—	—	Reserved(PXA32x and PXA30x Only)
21	R	BPB2IMG	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (53) = 0b0) **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
20	R	ACCU	0 = No Application Subsystem Clock Unit interrupt notification 1 = Application Subsystem Clock Unit interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (52) = 0b0)

Copyright © 2009 Marvell

Table 131: ICIP2 Bit Definitions (Continued)

Physical Address **ICIP2** **Interrupt Controller**
0x40D0_009C
Coprocessor Register: CP6, CR6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
19	R	DMC	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (51) = 0b0)
18	R	WAKEUP1	0 = No interrupt notification (PXA32x Only) 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (50) = 0b0)
	—	—	Reserved (PXA31x and PXA30x Only)
17	R	WAKEUP0	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (49) = 0b0)
16	R	DMA_MVED	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (48) = 0b0) **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
15	R	SPMU	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (47) = 0b0)

Copyright © 2009 Marvell

Table 131: ICIP2 Bit Definitions (Continued)

Physical Address **ICIP2** **Interrupt Controller**
0x40D0_009C
Coprocessor Register: CP6, CR6

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
14	R	USB 2	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (46) = 0b0)
13	R	NAND INF	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (45) = 0b0)
12	R	ONE WIRE	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (44) = 0b0)
11:10	—	—	Reserved
9	R	MMC 2	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (41) = 0b0)
8	—	—	Reserved
7	R	GRAPHICS	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (39) = 0b0)

Table 131: ICIP2 Bit Definitions (Continued)

Physical Address 0x40D0_009C **ICIP2** **Interrupt Controller**
Coprocessor Register: CP6, CR6

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Resreved	Resreved	Resreved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
6	R	USIM 2	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (38) = 0b0)
5	—	—	Reserved
4	R	TSI	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (36) = 0b0) **NOTE:** (PXA32x Only)
	—	—	Reserved (PXA31x and PXA30x Only)
3	—	—	Reserved
2	R	CONSUMER IR	Consumer IR 0 = No interrupt notification 1 = Consumer IR interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (34) = 0b0)
1	R	CIF	Capture Interface 0 = No interrupt notification 1 = Capture interface interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (33) = 0b0)
0	—	—	Reserved

Copyright © 2009 Marvell

9.4.3 Interrupt Controller FIQ Pending Registers (ICFP and ICFP2)

ICFP and ICFP2 have one bit per interrupt source. A bit is set if the corresponding peripheral has a pending unmasked FIQ interrupt waiting to be served (see Table 132 and Table 133).

Table 132: ICFP Bit Definitions

Physical Address 0x40D0_000C **ICFP** **Interrupt Controller**
Coprocessor Register: CP6, CR3

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
31	R	RTC_AL	Real-Time Clock Alarm 0 = No interrupt notification 1 = RTC equals alarm register has occurred, interrupt level<31> = 1, and either mask bit<31> = 1 or DIM bit = 0.
30	R	RTC_HZ	One Hz Clock 0 = No interrupt notification 1 = One Hz clock TIC has occurred, interrupt level<30> = 1, and either mask bit<30> = 1 or DIM bit = 0.
29	R	OST_3	OS Timer 3 0 = No interrupt notification 1 = OS timer equals match register 3, interrupt level<29> = 1, and either mask bit<29> = 1 or DIM bit = 0.
28	R	OST_2	OS Timer 2 0 = No interrupt notification 1 = OS timer equals match register 2, interrupt level<28> = 1, and either mask bit<28> = 1 or DIM bit = 0.
27	R	OST_1	OS Timer 1 0 = No interrupt notification 1 = OS timer equals match register 1, interrupt level<27> = 1, and either mask bit<27> = 1 or DIM bit = 0.
26	R	OST_0	OS Timer 0 0 = No interrupt notification 1 = OS timer equals match register 0, interrupt level<26> = 1, and either mask bit<26> = 1 or DIM bit = 0.

Copyright © 2009 Marvell

Table 132: ICFP Bit Definitions (Continued)

Physical Address 0x40D0_000C **Coprocessor Register:** CP6, CR3 — **ICFP** — **Interrupt Controller**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
25	R	DMAC	DMAC 0 = No interrupt notification 1 = DMA Channel service request has occurred, interrupt level<25> = 1, and either mask bit<25> = 1 or DIM bit = 0.
24	R	SSP1	SSP 1 0 = No interrupt notification 1 = SSP 1 service request has occurred, interrupt level<24> = 1, and either mask bit<24> = 1 or DIM bit = 0.
23	R	MMC1	MultiMediaCard 1 0 = No interrupt notification 1 = Flash Card status has changed or any error has been detected, interrupt level<23> = 1, and either mask bit<23> = 1 or DIM bit = 0.
22	R	UART1	UART1 0 = No interrupt notification 1 = A transmit or receive error in UART1, interrupt level<22> = 1, and either mask bit<22> = 1 or DIM bit = 0.
21	R	UART2	UART2 0 = No interrupt notification 1 = A transmit or receive error in UART2, interrupt level<21> = 1, and either mask bit<21> = 1 or DIM bit = 0.
20	R	UART3	UART3 0 = No interrupt notification 1 = A transmit or receive error in UART3, interrupt level<20> = 1, and either mask bit<20> = 1 or DIM bit = 0.
19	—	—	Reserved
18	R	I2C	I^2C 0 = No interrupt notification 1 = A I^2C service request, interrupt level<18> = 1, and either mask bit<18> = 1 or DIM bit = 0.

Copyright © 2009 Marvell

Table 132: ICFP Bit Definitions (Continued)

Physical Address **ICFP** **Interrupt Controller**
0x40D0_000C
Coprocessor Register:
CP6, CR3

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
17	R	LCD	LCD 0 = One of the requirements for setting the bit has not been met. 1 = An LCD controller service request, interrupt level<17> = 1, and either mask bit<17> = 1 or DIM bit = 0.
16	R	SSP2	SSP 2 0 = No interrupt notification 1 = An SSP 2 service request, interrupt level<16> = 1, and either mask bit<16> = 1 or DIM bit = 0.
15	R	USIM1	USIM1 0 = No interrupt notification 1 = A smart card interface status/error, interrupt level<15> = 1, and either mask bit<15> = 1 or DIM bit = 0.
14	R	AC97	AC97 0 = No interrupt notification 1 = An AC97 interrupt, interrupt level 14 = 1, and either mask bit 14 = 1 or DIM bit = 0.
13	R	SSP4	SSP4 0 = No interrupt notification 1 = An SSP4 interrupt, interrupt level 13 = 1, and either mask bit 13 = 1 or DIM bit = 0.
12	R	PML	Performance Monitor Unit 0 = No interrupt notification 1 = A PML interrupt, interrupt level 12 = 1, and either mask bit 12 = 1 or DIM bit = 0.
11	R	USBD	USB Device Controller (PXA32x and PXA30x Only) 0 = No interrupt notification 1 = A USB Client interrupt, interrupt level 11 = 1, and either mask bit 11 = 1 or DIM bit = 0.
	—	—	Reserved (PXA31x Only)

Table 132: ICFP Bit Definitions (Continued)

Physical Address 0x40D0_000C | **ICFP** | **Interrupt Controller**
Coprocessor Register: CP6, CR3

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
10	R	GPIO_X	GPIO x 0 = No interrupt notification. 1 = A GPIO_x (other than GPIO_0 and GPIO_1) edge detect =1, interrupt level 10 = 1, and either mask bit 10 = 1 or DIM bit = 0.
9	R	GPIO_1	GPIO 1 0 = No interrupt notification 1 = GPIO<1> detected an edge, interrupt level 9 = 1, and either mask bit<9> = 1 or DIM bit = 0.
8	R	GPIO_0	GPIO 0 0 = No interrupt notification 1 = GPIO<0> detected an edge, interrupt level 8 = 1, and either mask bit 8 = 1 or DIM bit = 0.
7	R	OST_4_11	OS Timer 4-11 0 = No interrupt notification 1 = An OS timer match 4-11, interrupt level 7 = 1, and either mask bit 7 = 1 or DIM bit = 0.
6	R	PWR_I2C	Power I^2C 0 = No interrupt notification 1 = An I^2C Power Unit interrupt, interrupt level 6 = 1, and either mask bit 6 = 1 or DIM bit = 0.
5	—	—	Reserved
4	R	KEYPAD	Keypad 0 = No interrupt notification 1 = A Keypad Controller interrupt, interrupt level<4> = 1, and either mask bit<4> = 1 or DIM bit = 0.

Copyright © 2009 Marvell

Table 132: ICFP Bit Definitions (Continued)

Physical Address **ICFP** **Interrupt Controller**
0x40D0_000C
Coprocessor Register:
CP6, CR3

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
3	R	USBH1	USB Host Interrupt 1 0 = No USB Host interrupt 1 notification (OHCI). 1 = USB Host interrupt 1 (OHCI) service request, interrupt level<3> = 1, and either mask bit 3 = 1 or DIM bit = 0. **NOTE:** When an OHCI interrupt occurs software must read the UHC Interrupt Status (UHCINTS) Register to determine the source of the interrupt.
2	R	USBH2	USB Host Interrupt 2 0 = No USB Host interrupt 2 notification (Non-OHCI) 1 = USB Host interrupt 2 (Non-OHCI) service request, interrupt level<2> = 1, and either mask bit 2 = 1 or DIM bit = 0. **NOTE:** When a Non-OHCI interrupt occurs software must read the UHC Status (UHCSTAT) Register to determine the source of the interrupt.
1	—	—	Reserved
0	R	SSP3	SSP 3 0 = No interrupt notification 1 = SSP three service request, interrupt level<0> = 1, and either mask bit 0 = 1 or DIM bit = 0.

Table 133: ICFP2 Bit Definitions

Physical Address **ICFP2** **Interrupt Controller**
0x40D0_00A8
Coprocessor Register:
CP6, CR9

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
31:24	—	—	reserved
23	R	MMC3	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (53) = 0b0) **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
22	R	MVED	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (53) = 0b0) **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
21	R	BPB2IMG	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (53) = 0b0) **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)

Copyright © 2009 Marvell

Table 133: ICFP2 Bit Definitions (Continued)

Physical Address **ICFP2** **Interrupt Controller**
0x40D0_00A8
Coprocessor Register:
CP6, CR9

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
20	R	ACCU	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (52) = 0b1)
19	R	DMC	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (51) = 0b1)
18	R	WAKEUP1	0 = No interrupt notification Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (50) = 0b1) **NOTE:** (PXA32x Only)
	—	—	Reserved (PXA31x and PXA30x Only)
17	R	WAKEUP0	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (49) = 0b1)
16	—	—	Reserved

Table 133: ICFP2 Bit Definitions (Continued)

Physical Address 0x40D0_00A8 **Coprocessor Register:** CP6, CR9 — **ICFP2** — **Interrupt Controller**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
16	R	DMA_MVED	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (48) = 0b1) **NOTE:** (PXA31x Only)
	—	—	reserved (PXA32x and PXA30x Only)
15	R	SPMU	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (47) = 0b1)
14	R	USB 2	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (46) = 0b1)
13	R	NAND INF	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (45) = 0b1)
12	R	ONE WIRE	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (44) = 0b1)
11:10	—	—	Reserved

Copyright © 2009 Marvell

Table 133: ICFP2 Bit Definitions (Continued)

Physical Address **ICFP2** **Interrupt Controller**
0x40D0_00A8
Coprocessor Register:
CP6, CR9

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
9	R	MMC 2	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (41) = 0b1)
8	—	—	Reserved
7	R	GRAPHICS	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (39) = 0b1)
6	R	USIM 2	0 = No interrupt notification 1 = Interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (38) = 0b1)
5	—	—	Reserved
4	R	TSI	0 = No interrupt notification 1 = interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (36) = 0b1) **NOTE:** (PXA32x Only)
	—	—	Reserved (PXA31x and PXA30x Only)
3	—	—	Reserved

Table 133: ICFP2 Bit Definitions (Continued)

Physical Address 0x40D0_00A8 Coprocessor Register: CP6, CR9	ICFP2	Interrupt Controller

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved											ACCU	DMC	Reserved	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
2	R	CONSUMER IR	Consumer IR 0 = No interrupt notification 1 = Consumer IR interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (34) = 0b1)
1	R	CIF	Capture Interface 0 = No interrupt notification 1 = Capture Interface interrupt occurs and ((mask bit(0) = 0b1) OR (DIM bit = 0b0)) and (interrupt level (33) = 0b1)
0	—	—	Reserved

9.4.4 Interrupt Controller Mask Registers (ICMR and ICMR2)

ICMR and ICMR2 contain one mask bit per pending interrupt. Any active interrupt from the source is sent to the IRQ or FIQ under the following conditions:

- If the corresponding mask bit in the ICMR (or ICMR2) is set and the processor is in the S0/D0/C0 state.
- If the corresponding mask bit in the ICMR (or ICMR2) is set and the processor is in S0/D0/C1 state and the ICCR[DIM] bit is set
- If the processor is in S0/D0/C1 state and the ICCR[DIM] bit is cleared, regardless of the mask bits in the ICMR (or ICMR2) register.

Copyright © 2009 Marvell

The ICMR and ICMR2 bits are initialized and reset to zero, which indicates that all interrupts are masked at reset and that software must configure ICMR and ICMR2.

Table 134 and Table 135 shows the ICMR and the ICMR2 register bit locations that correspond to the interrupt mask bits.

Table 134: ICMR Bit Definitions

Physical Address 0x40D0_0004 **ICMR** **Interrupt Controller**
Coprocessor Register: CP6, CR1

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
31	R/W	RTC_AL	Real-Time Clock Alarm 0 = Masked. 1 = RTC equals Alarm register interrupt is not to be masked.
30	R/W	RTC_HZ	One Hz Clock 0 = Masked. 1 = One Hz clock TIC interrupt is not to be masked.
29	R/W	OST_3	OS Timer 3 0 = Masked. 1 = OS timer equals Match register 3 interrupt is not to be masked.
28	R/W	OST_2	OS Timer 2 0 = Masked. 1 = OS timer equals Match register 2 interrupt is not to be masked.
27	R/W	OST_1	OS Timer 1 0 = Masked. 1 = OS timer equals Match register 1 interrupt is not to be masked.
26	R/W	OST_0	OS Timer 0 0 = Masked. 1 = OS timer equals Match register 0 interrupt is not to be masked.
25	R/W	DMAC	DMA Controller 0 = Masked. 1 = DMA channel service request interrupt is not to be masked.

Table 134: ICMR Bit Definitions (Continued)

Physical Address 0x40D0_0004 **ICMR** **Interrupt Controller**
Coprocessor Register: CP6, CR1

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
24	R/W	SSP1	SSP 1 0 = Masked. 1 = SSP 1 service request interrupt is not to be masked.
23	R/W	MMC1	MultiMediaCard 0 = Masked. 1 = MultiMediaCard interrupt is not to be masked.
22	R/W	UART1	UART1 0 = Masked. 1 = UART1 interrupt is not to be masked.
21	R/W	UART2	UART2 0 = Masked. 1 = UART2 interrupt is not to be masked.
20	R/W	UART3	UART3 0 = Masked. 1 = UART3 interrupt is not to be masked.
19	—	—	Reserved
18	R/W	I2C	I^2C 0 = Masked. 1 = I^2C interrupt is not to be masked.
17	R/W	LCD	LCD Controller 0 = Masked. 1 = LCD controller interrupt is not to be masked.
16	R/W	SSP2	SSP 2 0 = Masked. 1 = SSP 2 service request interrupt is not to be masked.
15	R/W	USIM1	USIM 1 0 = Masked. 1 = Smart card interface status/error interrupt is not to be masked.

Copyright © 2009 Marvell

Table 134: ICMR Bit Definitions (Continued)

Physical Address **ICMR** **Interrupt Controller**
0x40D0_0004
Coprocessor Register:
CP6, CR1

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
14	R/W	AC97	AC97 0 = Masked. 1 = AC97 interrupt is not to be masked.
13	R/W	SSP4	SSP4 0 = Masked. 1 = I^2S interrupt is not to be masked.
12	R/W	PML	Performance Monitor Unit 0 = Masked. 1 = PML interrupt is not to be masked.
11	R/W	USBC	USB Device Controller (PXA32x and PXA30x Only) 0 = Masked. 1 = USB client interrupt is not to be masked.
	—	—	Reserved (PXA31x Only)
10	R/W	GPIO_X	GPIO_x 0 = Masked. 1 = GPIO_x (other than GPIO_0 and GPIO_1) edge detected interrupt is not to be masked.
9	R/W	GPIO_1	GPIO_1 0 = Masked. 1 = GPIO<1> edge detect interrupt is not to be masked.
8	R/W	GPIO_0	GPIO_0 0 = Masked. 1 = GPIO<0> edge detect interrupt is not to be masked.
7	R/W	OST_4_11	OS Timer 4-11 0 = Masked. 1 = OS timer match 4-11 interrupt is not to be masked.

Table 134: ICMR Bit Definitions (Continued)

Physical Address 0x40D0_0004 **ICMR** **Interrupt Controller**
Coprocessor Register: CP6, CR1

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	?	0

Bits	Access	Name	Description
6	R/W	PWR_I2C	Power Manager I^2C 0 = Masked. 1 = I^2C power interrupt is not to be masked.
5	—	—	Reserved
4	R/W	KEYPAD	Keypad Controller 0 = Masked. 1 = Keypad controller interrupt is not to be masked.
3	R/W	USBH1	USB Host Interrupt 1 0 = Masked. 1 = USB host interrupt 1 (OHCI) is not to be masked.
2	R/W	USBH2	USB Host Interrupt 2 0 = Masked. 1 = USB host interrupt 2 (Non-OHCI) is not to be masked.
1	—	—	Reserved
0	R/W	SSP3	SSP 3 0 = Masked. 1 = SSP 3 service request interrupt is not to be masked.

Copyright © 2009 Marvell

Table 135: ICMR2 Bit Definitions

Physical Address **ICMR2** **Interrupt Controller**
0x40D0_00A0
Coprocessor Register:
CP6, CR7

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	WAKEUP 1	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
31:20	—	—	reserved
23	R/W	MMC3	MMC3 interrupt from BPB2 0 = Masked. 1 = Interrupt is not to be masked. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
22	R/W	MVED	MVED Interrupt from Island AXI 0 = Masked. 1 = Interrupt is not to be masked. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
21	R/W	BPB2IMG	BPB2IMG Bridge from Island AXI 0 = Masked. 1 = Interrupt is not to be masked. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)

Table 135: ICMR2 Bit Definitions (Continued)

Physical Address 0x40D0_00A0 **ICMR2** **Interrupt Controller**
Coprocessor Register: CP6, CR7

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	WAKEUP 1	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
20	R/W	ACCU	Application Subsystem CCU 0 = Masked. 1 = Interrupt is not to be masked.
19	R/W	DMC	DMC 0 = Masked. 1 = Interrupt is not to be masked.
18	R/W	WAKEUP<1>	WAKEUP<1> 0 = Masked. 1 = Interrupt is not to be masked.
	—	—	Reserved
17	R/W	WAKEUP<0>	WAKEUP<0> 0 = Masked. 1 = Interrupt is not to be masked.
16	—	—	Reserved
15	R/W	SGP PMU	SGP Master PMU 0 = Masked. 1 = Interrupt is not to be masked.

Copyright © 2009 Marvell

Table 135: ICMR2 Bit Definitions (Continued)

Physical Address **ICMR2** **Interrupt Controller**
0x40D0_00A0
Coprocessor Register:
CP6, CR7

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	WAKEUP 1	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
14	R/W	USB 2	USB 2 0 = Masked. 1 = Interrupt is not to be masked.
13	R/W	NAND INF	NAND interface 0 = Masked. 1 = Interrupt is not to be masked.
12	R/W	ONE WIRE	One Wire 0 = Masked. 1 = Interrupt is not to be masked.
11:10	—	—	Reserved
9	R/W	MMC 2	MMC 2 0 = Masked. 1 = Interrupt is not to be masked.
8	—	—	Reserved
7	R/W	GRAPHICS	Graphics 0 = Masked. 1 = Interrupt is not to be masked.

Table 135: ICMR2 Bit Definitions (Continued)

Physical Address 0x40D0_00A0 **ICMR2** **Interrupt Controller**
Coprocessor Register: CP6, CR7

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	WAKEUP 1	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
PXA31x Only	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
PXA30x Only	Reserved								Reserved	Reserved	Reserved	ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
6	R/W	USIM 2	USIM 2 0 = Masked. 1 = Interrupt is not to be masked.
5	—	—	Reserved
4	R/W	TSI	TSI 0 = Masked. 1 = Interrupt is not to be masked. **NOTE:** (PXA32x Only)
	—	—	Reserved (PXA31x and PXA30x Only)
3	—	—	Reserved
2	R/W	CONSUMER IR	Consumer IR 0 = Masked. 1 = Interrupt is not to be masked.
1	R/W	CIF	Capture Interface 0 = Masked. 1 = Interrupt is not to be masked.
0	—	—	Reserved

Copyright © 2009 Marvell

9.4.5 Interrupt Controller Level Registers (ICLR and ICLR2)

ICLR and ICLR2 control whether a pending interrupt generates an FIQ or an IRQ interrupt. If a pending interrupt is unmasked, the corresponding ICLR (or ICLR2) bit field is decoded to select which CPU interrupt is asserted. Table 136 and Table 137 show the location of all interrupt-level bits in the ICLR and the ICLR2. The ICLR and the ICLR2 registers are initialized to all zeros at reset and software must configure them to reflect the correct value for normal operation.

Table 136: ICLR Bit Definitions

Physical Address **ICLR** **Interrupt Controller**
0x40D0 0008
Coprocessor Register:
CP6, CR2

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	RTC_AL	Real Time Clock Alarm 0 = RTC equals Alarm register interrupt creates an IRQ. 1 = RTC equals Alarm register interrupt creates an FIQ.
30	R/W	RTC_HZ	One Hz Clock 0 = One Hz clock TIC interrupt creates an IRQ. 1 = One Hz clock TIC interrupt creates an FIQ.
29	R/W	OST_3	OS Timer 3 0 = OS timer equals Match register 3 interrupt creates an IRQ. 1 = OS timer equals Match register 3 interrupt creates an FIQ.
28	R/W	OST_2	OS Timer 2 0 = OS timer equals Match register 2 interrupt creates an IRQ. 1 = OS timer equals Match register 2 interrupt creates an FIQ.
27	R/W	OST_1	OS Timer 1 0 = OS timer equals Match register 1 interrupt creates an IRQ. 1 = OS timer equals Match register 1 interrupt creates an FIQ.
26	R/W	OST_0	OS Timer 0 0 = OS timer equals Match register 0 interrupt creates an IRQ. 1 = OS timer equals Match register 0 interrupt creates an FIQ.
25	R/W	DMAC	DMA Controller 0 = DMA channel service request interrupt creates an IRQ. 1 = DMA channel service request interrupt creates an FIQ.

Table 136: ICLR Bit Definitions (Continued)

Physical Address 0x40D0 0008 **ICLR** **Interrupt Controller**
Coprocessor Register: CP6, CR2

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
24	R/W	SSP1	SSP 1 0 = SSP 1 service request interrupt creates an IRQ. 1 = SSP 1 service request interrupt creates an FIQ.
23	R/W	MMC1	MultiMediaCard 1 0 = MMC interrupt creates an IRQ. 1 = MMC interrupt creates an FIQ.
22	R/W	UART1	UART1 0 = UART1 interrupt creates an IRQ. 1 = UART1 interrupt creates an FIQ.
21	R/W	UART2	UART2 0 = UART2 interrupt creates an IRQ. 1 = UART2 interrupt creates an FIQ.
20	R/W	UART3	UART3 0 = UART3 interrupt creates an IRQ. 1 = UART3 interrupt creates an FIQ.
19	—	—	Reserved
18	R/W	I2C	**I^2C**: 0 = I^2C interrupt creates an IRQ. 1 = I^2C interrupt creates an FIQ.
17	R/W	LCD	LCD Controller 0 = LCD controller interrupt creates an IRQ. 1 = LCD controller interrupt creates an FIQ.
16	R/W	SSP2	SSP 2 0 = SSP 2 service request interrupt creates an IRQ. 1 = SSP 2 service request interrupt creates an FIQ.
15	R/W	USIM1	USIM:1 0 = Smart card interface status/error interrupt creates an IRQ. 1 = Smart card interface status/error interrupt creates an FIQ.

Copyright © 2009 Marvell

Table 136: ICLR Bit Definitions (Continued)

Physical Address **ICLR** **Interrupt Controller**
0x40D0 0008
Coprocessor Register:
CP6, CR2

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
14	R/W	AC97	AC97 0 = AC97 interrupt creates an IRQ. 1 = AC97 interrupt creates an FIQ.
13	R/W	SSP4	SSP4 0 = I^2S interrupt creates an IRQ. 1 = I^2S interrupt creates an FIQ.
12	R/W	PML	Performance Monitor Unit 0 = PML interrupt creates an IRQ. 1 = PML interrupt creates an FIQ.
11	R/W	USBD	USB Device Controller (PXA32x and PXA30x Only) 0 = USB Device interrupt creates an IRQ. 1 = USB Device interrupt creates an FIQ.
	—	—	Reserved (PXA31x Only)
10	R/W	GPIO_X	GPIO_x 0 = GPIO_x (other than GPIO_0 and GPIO_1) edge detected interrupt creates an IRQ. 1 = GPIO_x (other than GPIO_0 and GPIO_1) edge detected interrupt creates an FIQ.
9	R/W	GPIO_1	GPIO_1 0 = GPIO<1> edge detect interrupt creates an IRQ. 1 = GPIO<1> edge detect interrupt creates an FIQ.
8	R/W	GPIO_0	GPIO_0 0 = GPIO<0> edge detect interrupt creates an IRQ. 1 = GPIO<0> edge detect interrupt creates an FIQ.
7	R/W	OST_4_11	OS Timer 4-11 0 = OS timer match 4-11 interrupt creates an IRQ. 1 = OS timer match 4-11 interrupt creates an FIQ.

Table 136: ICLR Bit Definitions (Continued)

Physical Address 0x40D0 0008 **Coprocessor Register:** CP6, CR2 — **ICLR** — **Interrupt Controller**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	USBD	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
PXA31x Only	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC 1	UART1	UART2	UART3	Reserved	I2C	LCD	SSP2	USIM1	AC97	SSP4	PML	Reserved	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Reserved	KEYPAD	USBH1	USBH2	Reserved	SSP3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0

Bits	Access	Name	Description
6	R/W	PWR_I2C	Power I^2C 0 = I^2C power unit interrupt creates an IRQ. 1 = I^2C power unit interrupt creates an FIQ.
5	—	—	Reserved
4	R/W	KEYPAD	Keypad 0 = Keypad controller interrupt creates an IRQ. 1 = Keypad controller interrupt creates an FIQ.
3	R/W	USBH1	USB Host Interrupt 1 0 = USB host interrupt 1 (OHCI) creates an IRQ. 1 = USB host interrupt 1 (OHCI) creates an FIQ.
2	R/W	USBH2	USB Host Interrupt 2 0 = USB host interrupt 2 creates an IRQ. 1 = USB host interrupt 2 creates an FIQ.
1	—	—	Reserved
0	R/W	SSP3	SSP 3 0 = SSP 3 service request interrupt creates an IRQ. 1 = SSP 3 service request interrupt creates an FIQ.

Copyright © 2009 Marvell

Table 137: ICLR2 Bit Definitions

Physical Address **ICLR2** **Interrupt Controller**
0x40D0 00A4
Coprocessor Register: CP6, CR8

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
	Reserved											ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
31:24	—	—	reserved
23	R/W	MMC3	MMC3 interrupt from BPB2 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
22	R/W	MVED	MVED Bridge from Island AXI 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)
21	R/W	BPB2IMG	BPB2IMG Bridge from Island AXI 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ. **NOTE:** (PXA31x Only)
	—	—	Reserved (PXA32x and PXA30x Only)

Table 137: ICLR2 Bit Definitions (Continued)

Physical Address 0x40D0 00A4 **ICLR2** **Interrupt Controller**
Coprocessor Register: CP6, CR8

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
	Reserved											ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
20	R/W	ACCU	Processor CCU 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
19	R/W	DMC	DMC 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
18	R/W	WAKEUP1	WAKEUP<1> (PXA32x Only) 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
	—	—	Reserved (PXA31x and PXA30x Only)
17	R/W	WAKEUP0	WAKEUP<0> 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
16	R/W	DMA_MVED	Video encoder decoder DMA (PXA31x Only) 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
	—	—	reserved (PXA32x and PXA30x Only)

Copyright © 2009 Marvell

Table 137: ICLR2 Bit Definitions (Continued)

Physical Address **ICLR2** **Interrupt Controller**
0x40D0 00A4
Coprocessor Register: CP6, CR8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
	Reserved											ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
15	R/W	SGP PMU	SGP PMU 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
14	R/W	USB 2	USB 2 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
13	R/W	NAND INF	Nand Interface 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
12	R/W	ONE WIRE	One Wire 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
11:10	—	—	Reserved
9	R/W	MMC 2	MMC 2 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
8	—	—	Reserved

Table 137: ICLR2 Bit Definitions (Continued)

Physical Address 0x40D0 00A4 **ICLR2** **Interrupt Controller**
Coprocessor Register: CP6, CR8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved											ACCU	DMC	WAKEUP1	WAKEUP0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	TSI	Reserved	CONSUMER IR	CIF	Reserved
	Reserved								MMC3	MVED	BPB2IMG	ACCU	DMC	Reserved	WAKEUP 0	DMA_MVED	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
	Reserved											ACCU	DMC	Reserved	WAKEUP 0	Reserved	SPMU	USB 2	NAND INF	ONE WIRE	Reserved	Reserved	MMC 2	Reserved	GRAPHICS	USIM 2	Reserved	resreved	Reserved	CONSUMER IR	CIF	Reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	?	?	0	?	0	0	?	0	?	0	0	?

Bits	Access	Name	Description
7	R/W	GRAPHICS	Graphics 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
6	R/W	USIM 2	USIM 2 0 = Service request interrupt creates an IRQ. 1 = Service request interrupt creates a FIQ.
5	—	—	Reserved
4	R/W	TSI	0 = TSI: Service request interrupt creates an IRQ. (PXA32x Only) 1 = Service request interrupt creates a FIQ.
	—	—	Reserved (PXA31x and PXA30x Only)
3	—	—	Reserved
2	R/W	CONSUMER IR	Consumer IR 0 = Consumer IR service request interrupt creates an IRQ. 1 = Consumer IR service request interrupt creates a FIQ.
1	R/W	CIF	Capture Interface 0 = Capture interface service request interrupt creates an IRQ. 1 = Capture interface service request interrupt creates a FIQ.
0	—	—	Reserved

Copyright © 2009 Marvell

9.4.6 Interrupt Controller Control Register (ICCR)

ICCR contains a single control bit, the disable-idle mask bit (DIM). DIM bit controls interrupt generation only when CPU is in S0/D0/C1(idle) or S2/D3/C4(sleep) mode. When set, this bit inhibits the feature in S0/D0/C1 mode, in which any active interrupt can interrupt the CPU, regardless of the value in ICMR. Interrupt service routine is run if the interrupt mask bit is set or if DIM=0 and a masked interrupt wakes the processor. In the latter case, the interrupt handler does not see any interrupt status bits and returns immediately. Table 138 shows the location of the DIM bit in the ICCR. This is a read/write register. Ignore reads from Reserved bits. Reserved bits must be written with zeros.

Table 138: ICCR Bit Definitions

Physical Address 0x40D0 0014 | **ICCR** | **Interrupt Controller**

User Settings																																
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Reserved																															**DIM**
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	**Access**	**Name**	**Description**
31:1	—	—	Reserved
0	R/W	DIM	0 = Any interrupt in ICPR wakes up the processor out of S0/D0/C1 modes. This affects only the wake-up, not the interrupt bit. 1 = Only active, unmasked interrupts (as defined in the ICMR) bring the processor out of S0/D0/C1 modes. Cleared during resets. This bit affects only the wake-up from S0/D0/C1 modes.DIM is ignored in the S0/D0/C0 run mode.

9.4.7 Interrupt Priority Registers 0 to 55

The interrupt priority registers set a peripheral ID for each priority value. At reset, these registers are marked invalid. Software must program the values in these registers after reset before it unmasks the interrupts. These registers allow unique priority values to be defined for each peripheral. The interrupt controller uses the priority values to resolve situations in which multiple active unmasked interrupts occur at the same time. Each register contains the peripheral ID assigned to the corresponding priority. The priority value for the register is the register number. Priority 0 is the highest priority and 55 is the lowest priority. The peripheral ID with Priority 0 is written to IPR0. IPR52 contains the peripheral ID with priority 52, and so on.

The contents of the IPRs typically do not change during execution. If the register contents are changed during execution, all interrupts must be masked until the changes written to the IPRs take effect. Interrupts can be disabled by writing to the Core Program Status register (CPSR) and altering its F-bit and I-bit.

The IPRx[31] is the valid bit (0b1 = valid and 0b0 = invalid). The peripheral IDs range between 0 and 55. Peripheral IDs for different sources are shown in Table 125. The lower six bits are used to assign peripheral IDs. A valid peripheral ID must be in the priority registers. However, if a non-valid or non-existing peripheral ID (say 59 or 63 for example) is programmed into a priority register, it is ignored and has no effect on the determination of the highest-priority FIQ and IRQ interrupts. To ease the highest-priority selection process in the hardware, only one peripheral ID can be assigned to a single priority value. In addition, hardware does not offer any protection against assigning one

peripheral ID to multiple priority values. Software must ensure that such cases are avoided. Where multiple-priority values are assigned to one peripheral, the hardware uses only the highest-priority value and ignores the others. Table 139 shows the format for IPR registers. These are read/write registers. Ignore reads from reserved bits. Reserved bits must be written with zeros.

Table 139: IPR0/55 Bit Definitions

Physical Addresses 0x40D0_001C–0098 (IPR 00–31) and 0x40D0_00B0-010C (IPR32-55)	IPR0–IPR55	Interrupt Controller

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VAL	Reserved																									PID					
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	VAL	Valid Bit 0 = Peripheral ID in the register is not valid. 1 = Peripheral ID in the register is valid.
30:6	—	—	Reserved
5:0	R/W	PID	Peripheral ID For this Priority IPR[n] Peripheral ID of peripheral with priority n (n = IPR number). Valid IDs: 0 through 55. **Note:** Not all peripheral IDs are valid. Refer to Table 125 for the list of valid peripheral IDs.

9.4.8 Interrupt Control Highest Priority Register (ICHP)

ICHP contains the highest-priority peripheral ID that caused an interrupt. The register contains peripheral IDs for the level of interrupts for FIQ and IRQ. If there is no interrupting peripheral for a particular interrupt level, the corresponding peripheral ID field is invalidated. The register is updated when an unmasked interrupt occurs. If IPR are partially-defined, the ICHP uses the partial information to determine the highest-priority peripheral. If none of the IPR fields are defined, ICHP contains invalidated values in both fields. After an interrupt is served and the status bit is set in the Peripheral Status register, the ICHP register is updated with the peripheral ID with the next highest priority. Changes in the IPR registers can also lead to an update in ICHP. Table 140 shows the format for the ICHP register. This is a read-only register. Ignore reads from reserved bits.

Copyright © 2009 Marvell

Table 140: ICHP Bit Definitions

Physical Address **ICHP** **Interrupt Controller**
0x40D0_0018
Coprocessor Register:
CP6, CR5

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VAL_IRQ	Reserved									IRQ						VAL_FIQ	Reserved									FIQ					
Reset	0	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31	R	VAL_IRQ	Valid IRQ 0 = No valid peripheral ID is causing an IRQ interrupt. 1 = Valid peripheral ID is causing an IRQ interrupt.
30:22	—	—	Reserved
21:16	R	IRQ	IRQ Highest-Priority Field Peripheral ID with the highest IRQ priority. When no interrupt occurs, this bit is 0.
15	R	VAL_FIQ	Valid FIQ 0 = No valid peripheral ID is causing an FIQ interrupt. 1 = Valid peripheral ID is causing an FIQ interrupt.
14:6	—	—	Reserved
5:0	R	FIQ	FIQ Highest Priority Field Peripheral ID with the highest FIQ priority. When no interrupt occurs, this bit is 0.

Copyright © 2009 Marvell

10 Real-Time Clock (RTC)

The real-time clock (RTC) controller is a general-purpose, real-time reference for the system. The timer, wristwatch, stopwatch, periodic interrupt, and the trimmer modules provide the basic functionality of the RTC controller.

10.1 Features

The processor RTC controller has the following features:

- Timer feature (See Section 10.3.1)
  - User-programmable, free-running counter
  - User-programmable alarm register
  - Resolution of one second
- Wristwatch feature (See Section 10.3.2)
  - User-programmable, free-running counter containing time of the day in terms of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - User-programmable Alarm registers to generate alarms in terms of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - Resolution of one second
- Stopwatch feature (See Section 10.3.3)
  - Programmable Counter register that contains the time elapsed between two events in terms of hours, minutes, seconds, and hundredths of a second
  - Two user-programmable Alarm registers to generate alarms in terms of hours, minutes, seconds, and hundredths of a second
  - Resolution of one 100th of a second
- Periodic Interrupt feature (See Section 10.3.4)
  - Programmable Alarm register to generate periodic interrupts at regular intervals
  - Resolution of one millisecond
- Trimmer feature (See Section 10.3.5)
  - User-programmable Trimmer register to generate a precise 1-Hz clock for the timer and wristwatch modules

10.2 Signals

Table 141 describes the signal associated with the RTC controller.

Table 141: Real-Time Clock Controller I/O Signal

Signal Name	Type	Polarity	Description
HZ_CLK	Output	NA	Generated by the trimmer module of the RTC for use by the timer module and the wristwatch module of the RTC. The accuracy of the 1-Hz clock is one 32.768-kHz clock cycle. The high period of the clock pulse remains a constant and only the low period is adjusted as the user varies the trim to match the frequency.

10.3 Operation

The RTC controller uses the following modules:

- Timer
- Wristwatch
- Stopwatch
- Periodic Interrupt
- Trimmer

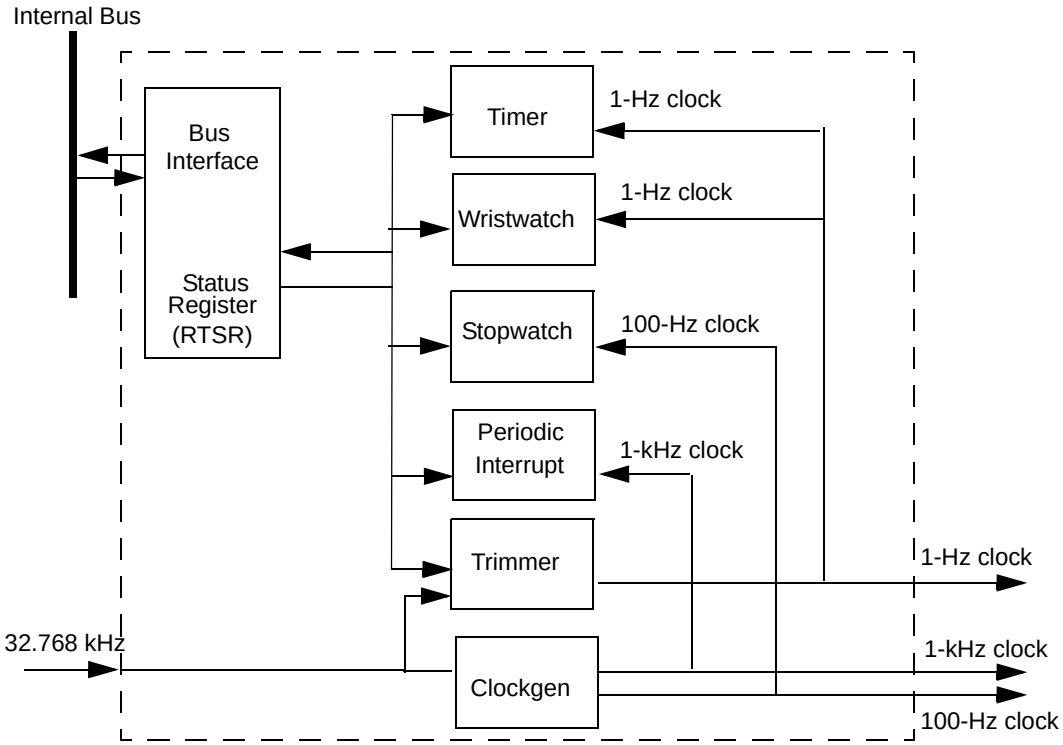
The peripheral-bus interface block contains the RTC Status register (RTSR) that provides the interface between the core and the RTC data. Figure 24 is a block diagram of the RTC controller and the RTC dedicated I/O.

Note

The signals labeled 1-Hz clock in Figure 24 are internal timing signals and are not the same as the external signal HZ_CLK.

Copyright © 2009 Marvell

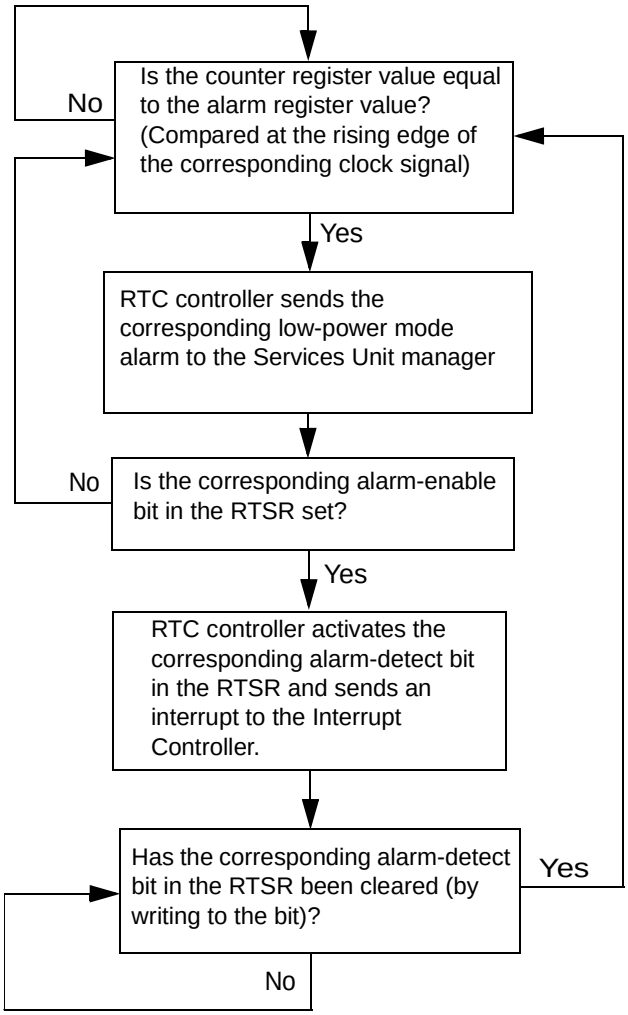
Figure 24: RTC Block Diagram



Except for the RTC Timer Trim register (RTTR), all registers in the RTC controller are reset by the hardware reset and the watchdog reset. The RTTR is reset by the hardware reset. The other resets, low-power mode exit reset, and GPIO reset, do not affect any RTC controller blocks.

Four of the five modules of the RTC controller, not including the trimmer, operate the same way. Figure 25 shows the operational flow of one of these four identical modules. All RTC modules are described in detail in subsequent sections.

Figure 25: Operational Flow of the RTC Modules



Each RTC module has one or more Counter registers and one or more corresponding Alarm registers. For example, the timer module has one Counter register and one Alarm register. The counters and alarms are configured as follows:

1. First, the preferred alarm-set conditions are written to the Alarm register.
2. The corresponding alarm-enable bit in the RTC Status register (RTSR) is then set.
3. If a counter has a count-enable bit, the corresponding count-enable bit must be set to enable the counter to start counting. The count-enable bits for the corresponding Counter registers reside in the RTSR. The stopwatch and periodic-interrupt modules have count-enable bits. The timer and wristwatch modules consist of free-running counters without any count-enable bits.
4. As shown in Table 142, when the data in the Counter register and the Alarm register are equal, the RTC controller checks whether the corresponding alarm-enable bit in the RTSR is set.
5. If so, the corresponding alarm-detect bit in the RTSR is set, indicating that an alarm is detected. This information is forwarded to the interrupt controller and to the power manager as shown in Figure 26.
6. The corresponding alarm-detect bit in the RTSR is cleared.

This process is identical for all alarm-detection events.

Copyright © 2009 Marvell

Table 142 lists the internal registers of the RTC controller and the module associated with each. Refer to Table 145: "RTC Controller Register Summary" for the location of each of these registers.

Table 142: RTC Controller Alarm Bit Location Summary

Module	Counter Register	Alarm Register(s)	Alarm Detect Bit	Alarm Enable Bit
Timer	RCNR	RTAR	RTSR[AL]	RTSR[ALE]
Wristwatch	RDCR RYCR	RDAR1, RYAR1 RDAR2, RYAR2	RTSR[RDAL1] RTSR[RDAL2]	RTSR[RDALE1] RTSR[RDALE2]
Stopwatch	SWCR	SWAR1 SWAR2	RTSR[SWAL1] RTSR[SWAL2]	RTSR[SWALE1] RTSR[SWALE2]
Periodic Interrupt	RTCPICR	PIAR	RTSR[PIAL]	RTSR[PIALE]
Trimmer	RTTR	N/A	N/A	N/A

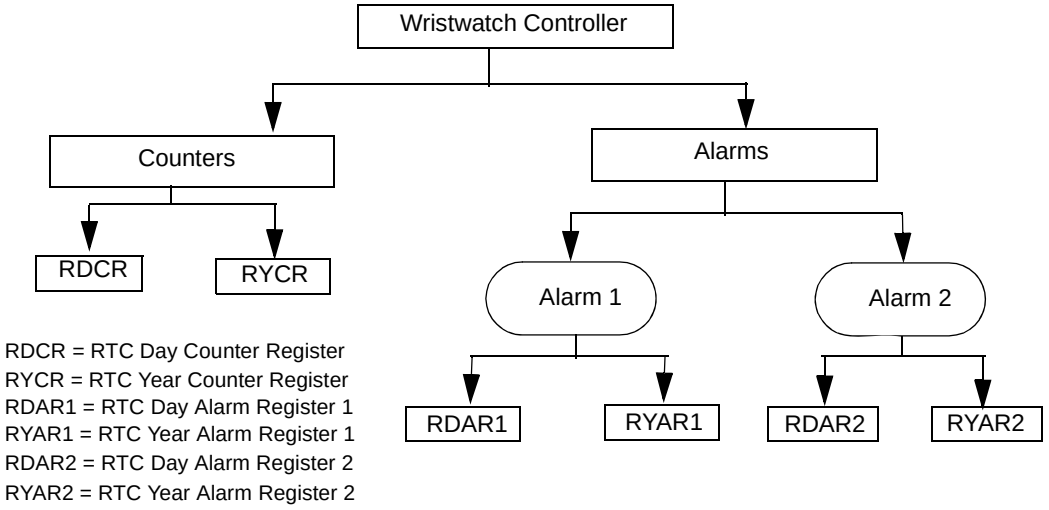
10.3.1 Timer Module

The RTC timer module consists of a free-running counter, RTC Counter register (RCNR), that starts incrementing after a hardware reset or watchdog reset is de-asserted. The count value is incremented at each rising edge of the 1-Hz clock. This value of this counter can be altered by writing to it. The value of the counter is unaffected by transitions into and out of low-power (S0/D1/C2, S0/D2/C2, S2/D3/C4) or idle (S0/D0/C1) modes. The corresponding RTC Alarm register, RTAR, can be written with a value to compare against the counter. On each rising edge of the 1-Hz clock, the counter is incremented and then compared to the value of RTAR. If the values match and if the alarm-enable bit (RTSR[ALE]) is set, the corresponding alarm-detect bit (RTSR[AL]) is set.

10.3.2 Wristwatch Module

The RTC wristwatch module consists of two sub-modules, counters and alarms, as shown in Figure 26.

Figure 26: RTC Wristwatch Module



The counters portion of the wristwatch module consists of two 32-bit, free-running counters (RDCR and RYCR) and two 32-bit alarms. Alarm 1 contains RDAR1 and RYAR1. Alarm 2 contains RDAR2 and RYAR2.

The wristwatch module runs in all power modes. The counters sub-module counts the current time and year. The time of the day in terms of seconds, minutes, hours, day-of-week, and week-of-month is available through the RDCR. The year, month, and day-of-month is available through the RYCR.

Alarm 1 has a corresponding alarm-detect bit (RTSR[RDAL1]) and a corresponding alarm-enable bit (RTSR[RDALE1]). Refer to columns labeled “Alarm Detect bit” and “Alarm Enable bit” in Table 142.

Similarly, Alarm 2 has a corresponding alarm-detect bit (RTSR[RDAL2]) and a corresponding alarm-enable bit (RTSR[RDALE2]). Refer to columns labeled “Alarm Detect bit” and “Alarm Enable bit” in Table 142.

10.3.2.1 Programming Wristwatch Registers

The wristwatch module has two counter registers (RDCR and RYCR) and four alarm registers (RDAR1, RYAR1, RDAR2, and RYAR2). The RDCR (refer to Table 154) and RDAR1/2 (refer to Table 149) each have five fields:

- Seconds
- Minutes
- Hour
- Day-of-week (DOW)
- Week-of-month (WOM)

The RYCR counter (refer to Table 155) the RYAR1/2 (refer to Table 150) each have three fields:

- Day-of-month (DOM)
- Month
- Year

10.3.2.2 Allowable Values for Wristwatch Register Fields

Table 143 provides all the valid and invalid values for each field of the counter and alarm registers.

Table 143: Valid and Invalid Data For The Wristwatch Register Fields

Field Name	Number of Bits Needed in Binary	Valid Data	Invalid Data
Seconds	6	0 to 59	60,61,62,63
Minutes	6	0 to 59	60,61,62,63
Hours	5	0 to 23	24 to 31
Day of Week (DOW)	3	1 to 7 1 = Sunday	0
Week of Month (WOM)	3	1 to 5	0,6,7
Day of Month (DOM)	5	Refer to Table 144	Refer to Table 144
Month	4	1 to 12	0,13,14,15

Copyright © 2009 Marvell

Table 143: Valid and Invalid Data For The Wristwatch Register Fields (Continued)

Field Name	Number of Bits Needed in Binary	Valid Data	Invalid Data
Year	12	0 to 4095	NA

Note

The Day-of-Month (DOM) field is a special case. Depending on the month and the type of year, the validity of the data in this field varies. Table 144 lists the valid values for this field for each month.

Table 144: Valid Data for Day of Month (DOM) Field In RYCR

Month	Year	Valid Data	Invalid Data
January, March, May, July, August, October, December	Any	1 to 31	0
April, June, September, November	Any	1 to 30	0,31
February	Leap Year	1 to 29	0,30,31
	Nonleap Year	1 to 28	0,29,30,31

For the months January, March, May, July, August, October, and December, the count can range from 1 to 31 because these months have 31 days. Because 31 is the maximum decimal number for the DOM field, five bits are needed to represent this number. Zero is the only non-valid data for these months.

For the months April, June, September, and November, the count ranges from 1 to 30 because these months have 30 days. The non-valid data in this instance is 31 and zero.

For leap years, the count in February ranges from 1 to 29; and the non-valid data in this instance is 30, 31, and zero. If the year is not a leap year, the count in February ranges from 1 to 28; therefore, the non-valid data in this example is 29, 30, 31, and zero.

10.3.2.3 Effects of Data Written to Wristwatch Register Fields

This section describes the effects of writing the Wristwatch register fields with valid and non-valid data. Zero and non-zero data is described separately for the alarm registers.

10.3.2.3.1 Writing Counter Registers with Valid and Invalid Data

The counter registers are updated only for a Write to the RTC Day Counter register (RDCR). When software writes to the RYCR register, the new RYCR value is stored in an internal register. This new value is written into the RYCR register only after a Write to the RDCR. If there is no Write to the RDCR, the new RYCR value in the internal register is never written into RYCR, and users read the original RYCR register value.

Any attempt to write non-valid data to the counter registers results in unpredictable behavior.

Copyright © 2009 Marvell

Valid and correct data must be written to the Wristwatch registers (RDCR and RYCR) initially. Subsequently, the registers are to be treated as read-only registers. The RDCR and RYCR registers rarely need to be updated after they receive an initial Write of valid data.

10.3.2.3.2 Writing Alarm Registers with Valid Data

Alarm 1 consists of the Alarm registers RDAR1 and RYAR1. Alarm 2 consists of the Alarm registers RDAR2 and RYAR2. A Write to these Alarm registers with valid data follows the same procedures as writing valid data to the Counter registers.

10.3.2.3.3 Writing Alarm Registers with Invalid Data (Other than Zero)

Except for the RDARx[WOM] field, writing non-valid data to the Alarm registers results in no match between the Counter and Alarm registers. For the WOM field, the non-valid data is 6 and 7. If 6 is written to the WOM field, the first and third weeks are matched. If 7 is written to the WOM field, the second and fourth weeks are matched. If RYARx[DOM] contains valid data, RDARx[DOW] and RDARx[WOM] are ignored for matching the data.

When RYARx[DOM] contains valid data, example values are as follows:

- RDAR1[SECONDS] contains 0b00_0000
- RDAR1[MINUTES] contains 0b00_0000
- RDAR1[HOURS] contains 0b00_0000
- RDAR1[DOW] contains 0b111
- RDAR1[WOM] contains 0b110
- RYAR1[DOM] contains 0b0_0001
- RYAR1[MONTH] contains 0b0001
- RYAR1[YEAR] contains 0b0111_1101_0000

In this example, the RYAR1[DOM] field contains valid data. As a result, RDAR1[DOW] and RDAR1[WOM] are ignored and the alarm is set on 01/01/2000 at 0:00:00 hours.

If invalid data is written to RYARx[DOM], the data in RDARx[DOW] and RDARx[WOM] is considered only for generating the alarm. In the example shown here, if RYAR1[DOM] contains invalid data, the data written in RDAR1[DOW] is 7 (Saturday) and the data written in RDAR1[WOM] field is 6, the alarm is set on the first and the third Saturday of the first month (January) in the year 2000 at 0:00:00 hours.

10.3.2.3.4 Writing Alarm Registers with Non-valid (Zero) Data

In some fields of the Alarm registers, zero is a non-valid value. The effects of zero in the fields of the alarm registers are as follows:

- Hours, minutes and seconds fields: Zero is valid for these fields.
- Day-of-Week (DOW), or Week-of-Month (WOM), Day-of-Month (DOM), Month or Year fields: Zero is not valid for these fields. If zero is written into any of these fields, it is ignored while generating the alarm. For example, if a zero is written into a DOW field, the alarm is set every day at the time that is written in the Hours, Minutes, and Seconds field. If zeroes are written into all the fields of the Alarm register, the alarm occurs at 0:00:00 hours every day.

  If the same data is written in all fields except the Month field, where a zero was entered, the hardware sets the alarm every month. Therefore, the alarm sets on the first day of every month in the year 2000 at 0:00:00 hours.

Copyright © 2009 Marvell

Note

The Hour field works in 24-hour mode only. Software must be used to convert time to 12-hour mode.

10.3.3 Stopwatch Module

The RTC stopwatch module consists of a StopWatch Counter register (SWCR) that measures the elapsed time between two events. This counter is activated with the corresponding stopwatch count-enable bit (RTSR[SWCE]).

The stopwatch has two alarm registers: StopWatch Alarm Register 1 (SWAR1) and StopWatch Alarm Register 2 (SWAR2). The corresponding alarm-detect bits and alarm-enable bits are listed in Table 142. The stopwatch 100th counter is clocked with a 100 Hz clock signal and increments at each rising edge of the 100 Hz clock signal. The stopwatch seconds, minutes and hours are clocked based on the trimmer 1 Hz clock. Notice that because the 100th and seconds fields of the stopwatch are not clocked by the same source the 100th value is not completely accurate.

Write a new value to SWAR1 to reset the SWCR counter to zero. Any value written to SWAR1 clears SWCR. Perform these steps to use the stopwatch function as a stopwatch:

1. Set the stopwatch count-enable bit (RTSR[SWCE]) so the counter can increment.
2. Clear RTSR[SWCE]. The counter stops counting, and SWCR contains the elapsed time between the two actions, setting and clearing RTSR[SWCE].
3. If SWCE is set again, the counter starts counting from where it stopped.

If another session is needed, (that is, with the counter starting to count from zero), a new value must be written into SWAR1 to reset the stopwatch counter and SWCR to zero.

10.3.3.1 Interval Interrupts

The SWCR can also be used to generate an interval interrupt. The required elapsed time is written to an Alarm register (SWAR1 or SWAR2) and then the corresponding alarm-enable (RTSR[SWALE1] or RTSR[SWALE2]) is set. When the Alarm register and the Counter register are equal, the RTC logic sets the corresponding alarm-detect bit (RTSR[SWAL1] or RTSR[SWAL2]).

Note

The value written to SWAR1 or SWAR2 is the elapsed time required, not the time (based on the current time) when the alarm should trigger because any Write to a StopWatch Alarm register clears the SWCR.

Refer to the following example procedure to use SWCR to generate an interval interrupt:

1. Write 60 minutes into SWAR1 (0x0007_8000) and 90 minutes into SWAR2 (0x000B_C000).
2. Set the alarm-enable bits (RTRS[SWALE1] and RTSR[SWALE2].
3. Set the count-enable bit (RTSR[SWCE]), which enables the counter (the count starts from zero).
4. When the count in the SWCR reaches 60 minutes, the first alarm-detect bit in the RTSR[SWAL1] is set.
5. The counter continues counting. When the count in the SWCR reaches 90 minutes, the second alarm-detect bit in the RTSR[SWAL2] is set.

6. If the count-enable bit (RTSR[SWCE]) is still set, the counter continues to count. When SWCE is cleared, the counter stops counting.

In this example, if SWCE is cleared when the counter reaches 100 minutes, the counter stops counting, and SWCR contains a value of 100 minutes.

When new data is written into SWAR1, SWCR is reset, a new session begins, and the counter rolls over to zero. If we modify the example by clearing the count-enable bit in the RTSR and then writing new data, such as 70 minutes, into SWAR1, the counter resets from 100 minutes to zero and is ready for a new session. Enabling the count-enable bit again causes the counter to increment from zero.

Note

When a new session is required, the count-enable bit must be set only after the new data is written into the Alarm register. The delay between the two Writes must be at least two CPU clock cycles.

10.3.4 Periodic Interrupt Module

The RTC periodic-interrupt controller module generates alarms periodically, depending on the interval programmed into the Periodic Interrupt Alarm register (PIAR). The counter is clocked on the positive edge of a 1 kHz clock. The Periodic Interrupt Counter register (RTCPICR) contains the current value of the periodic-interrupt counter.

The periodic-interrupt counter increments only when the periodic-interrupt count-enable bit (RTSR[PICE]) is set. When PICE is cleared, the counter stops counting at the value it reaches. If the counter is then re-enabled by setting PICE), it resumes counting from the value where it stopped.

The periodic-interrupt alarm-enable bit (RTSR[PIALE]) must be set for the periodic-interrupt alarm bit (RTSR[PIAL]) to be set. When the Periodic Interrupt Counter register (RTCPICR) equals the Periodic Interrupt Alarm register (PIAR), PIAL is set if PIALE is set and the periodic-interrupt counter resets to zero. This process repeats as long as PICE remains set.

For example, if the RTCPICR has 78 (0x4E) written into the PIAR, an alarm occurs (the RTSR[PIAL] is set) every 78 ms. To start a new session, write to the Periodic Interrupt Alarm register (PIAR). The PICE bit must be enabled only after PIAR is written with new data. There must be a delay of at least two CPU cycles between the two actions.

Note

The hardware behavior may be unpredictable when the PIAR is cleared and the periodic-interrupt alarm-enable bit (RTSR[PIALE]) is set. Zero is not a valid value for the Periodic Interrupt Alarm register. The general use of this alarm is to generate interrupts and wakeups to the processor in all low-power modes, including S0/D1/C2, S0D2/C2, S2/D3/C4 and S3/D4/C4. Therefore, a valid value (non-zero) must be written into PIAR. The maximum value that can be written is 65,535.

10.3.5 Trimmer Module

The RTC trimmer module generates the 1 Hz clock by dividing the 32.768 kHz crystal-oscillator output by approximately 32,768. The inherent inaccuracies of crystals aggravated by the varying capacitance of board connections causes the timebase to be somewhat inaccurate, requiring a periodic adjustment to the 1 Hz clock period.Through the RTTR, the processor allows the 1 Hz

Copyright © 2009 Marvell

timebase to be trimmed to an accuracy of +/- 5 seconds per month through the RTC trimming mechanism. The trimming procedure is described in Section 10.3.5.1.

Note

The trimming mechanism trims the 1 Hz clock only and not the 100 Hz clock, so the 100 Hz clock may be little off if the crystal oscillator frequency is not exactly 32768 Hz. For the stopwatch counter to operate correctly, the maximum deviation of the crystal oscillator frequency allowed is +/-100ppm from the base 32768 Hz.

10.3.5.1 Trim Procedure

The 1 Hz clock driving the RTC is obtained by dividing the output of the oscillator multiplexer. At reset, the RTTR contains 0x0000_7FFF, which yields an approximate 1 Hz clock. The RTTR is reset to its default value of 0x0000_7FFF each time the hardware reset bit is set.

To generate the RTTR value, the output frequency of the oscillator multiplexer must first be measured (approximately 32 kHz) using an accurate timebase, such as a frequency counter. Refer to the GPIO chapter for details on how to select the appropriate GPIO alternate function to make the oscillator multiplexer externally visible. The trim is accomplished by dividing the output of the oscillator by an integer value and then performing fine-grain fractional adjustment by periodically deleting clocks from the stream driving this integer divider.

10.3.5.2 RTTR Value Calculations

After the true oscillator frequency is known, it must be split into integer and fractional portions, as follows:

1. Write the integer portion of the value (minus one) into RTTR[CK_DIV].
2. Compare this value against a 16-bit counter clocked by the output of the oscillator multiplexer (approximately 32 kHz).
3. The counter resets and generates a pulse when the two values are equal. This pulse constitutes the raw 1 Hz signal.

The fractional part of the adjustment is performed by periodically deleting clocks from the clock stream driving the integer counter. The period, called the “trim interval,” is hardwired to be 2^{10} –1 seconds (approximately 17 minutes). The number of clocks deleted, called the “trim-delete value,” is a 10-bit programmable counter allowing from 0 to (2^{10} -1) 32-kHz clocks to be deleted from the input-clock stream once per trim interval. RTTR[DEL] represents the number of clocks deleted per trim operation.

In summary, every 2^{10} –1 seconds, the integer counter stops clocking for a period equal to the fractional error that has accumulated. If this counter is programmed to zero, no trim operations occur, and the RTC is clocked with the raw 32-kHz clock. The relationship between the nominal 1 Hz clock frequency and the nominal 32 kHz clock (f1 and f32K, respectively) is shown in the following equation:

$$f1= \frac{(210\text{-}1)*(RTTR[CK_DIV]+1) - RTTR[DEL]}{(210\text{-}1)*(RTTR[CK_DIV]+1)} * \frac{f32k}{(RTTR[CK_DIV]+1)}$$

- f1 = HZ clock frequency
- f32k = RTC internal clock - the 32.678 kHz crystal output
- RTTR[DEL] = RTTR(25:16)

- RTTR[CK_DIV] = RTTR(15:0)

Trim Example 1: Measured Value Has No Fractional Component

In this example,

- The required HZ clock frequency is 1 Hz.
- The oscillator output is measured as 36045.000 cycles/s (Hz).
- This output is exactly 3277 cycles over the nominal frequency of the crystal (32.768 kHz) and has no fractional component.
- Only the integer trim function is needed, and no fractional trim is required.
- RTTR[CK_DIV] is loaded with the binary equivalent of 36045 –1, or 0x0000_8CCC. RTTR[DEL] is left at zero (power-up state) to disable fractional trimming.

This trim exercise leaves an error of zero in trimming.

Trim Example 2: Measured Value Has a Fractional Component

A measured oscillator frequency with a fractional component occurs more frequently than an oscillator frequency without a fractional component. This example describes a measured oscillator frequency with a fractional component:

- The required clock-output frequency is 1 Hz.
- If the oscillator output is measured as 32768.92 cycles/s (Hz), an integer trim is necessary so that the *average* number of cycles counted before generating one 1 Hz clock is 32768.92.
- The integer field RTTR[CK_DIV] is loaded with the hexadecimal equivalent of 32768 –1 or 0x0000_7FFF (reset value).
- Because the actual clock frequency is 0.92 cycles per second faster than the integer value, the 1-Hz clock generated by only the integer trimming is slightly faster than needed and must be slowed down.
- Accordingly, program the fractional trim to delete 0.92 cycles per second on average to bring the 1 Hz output frequency down to the proper value.
- Because the trimming procedure is performed every 1023 (2^{10} –1) seconds, the trim must be set to delete 941.16 clocks every 1023 seconds (.92 x 1023 = 941.16).
- Load the counter with the hexadecimal equivalent of 941 (0x3AD). The fractional component of this value cannot be trimmed, and it constitutes the error in trimming.

This trim setting leaves an error of 0.16 cycles per 1023 seconds. The error calculation yields results in parts-per-million (ppm). An example error calculation for a measured value with a fractional component is as follows:

$$Error = \frac{0.16\ cycles}{1023\ sec} X \frac{1\ sec}{32768\ cycles} = ((0.004773))\ ppm$$

Maximum Error Calculation Versus Real-Time Clock Accuracy

As seen from the trim results, the maximum possible error approaches 1 clock per 2^{10}-1 seconds. Therefore, the ppm error yields:

$$Error\ (maximum) = \frac{1\ cycle}{1023\ sec} X \frac{1sec}{32768 cycles} = 0.0298\ ppm$$

To maintain an accuracy of +/- 5 seconds per month, the required accuracy is calculated to be:

Copyright © 2009 Marvell

$$Error = \frac{5\ sec}{month} X \frac{1\ month}{2592000\ sec} = 1.9\ ppm$$

This calculation indicates that the 1 Hz clock output can be made very accurate through the use of the trim procedure. Similarly, use the trim procedure to compensate for a range of factors that can affect crystal oscillators:

- Manufacturing and supplier variance in the crystals
- Crystal aging effects
- System voltage differences
- System manufacturing variance

The trim procedure can counteract these factors by providing a highly accurate mechanism to remove the variance and shifts from the manufacturing- and static-environment variables on an individual system level. Because this is a calibration solution, it is not practical for dynamic changes in the system and environment. It can most likely be accomplished only in a factory setting due to the equipment required.

10.4 Register Descriptions

The RTC registers control the RTC and communicate it status. Most registers are read/write registers. Marvell strongly recommends that the operating system use the core memory-management unit (MMU) protection mechanisms to prevent inadvertent Writes to the RCNR. See the *Intel XScale® Core Developer's Manual* for information on the MMU.

Because of the asynchronous nature of the 1 Hz clock relative to the processor clock, Writes to these registers are controlled by a hardware mechanism that delays the actual Write until the data can be properly synchronized. For multiple Writes to RTC Counter and Alarm registers in quick succession, the final update to the RTC register may be delayed by a maximum of six 32-kHz clock cycles. The RTC Counter and Alarm registers can be read at any time. Reads reflect the value in the register after it increments or after it is written.

Table 145 summarizes the RTC registers and the memory-mapped addresses to access them. For easy reference, the summary table includes the page number of the detailed description for each register. The remainder of this section describes the individual registers in detail.

Table 145: RTC Controller Register Summary

Physical Address	Name	Description	Page
0x4090_0000	RCNR	RTC Counter register	423
0x4090_0004	RTAR	RTC Alarm register	419
0x4090_0008	RTSR	RTC Status register	417
0x4090_000C	RTTR	RTC Timer Trim register	416
0x4090_0010	RDCR	RTC Day Counter register	423
0x4090_0014	RYCR	RTC Year Counter register	424
0x4090_0018	RDAR1	Wristwatch Day Alarm register 1	419
0x4090_001C	RYAR1	Wristwatch Year Alarm register 1	420

Copyright © 2009 Marvell

Table 145: RTC Controller Register Summary (Continued)

Physical Address	Name	Description	Page
0x4090_0020	RDAR2	Wristwatch Day Alarm register 2	419
0x4090_0024	RYAR2	Wristwatch Year Alarm register 2	420
0x4090_0028	SWCR	Stopwatch Counter register	424
0x4090_002C	SWAR1	Stopwatch Alarm register 1	421
0x4090_0030	SWAR2	Stopwatch Alarm register 2	421
0x4090_0034	RTCPICR	Periodic Interrupt Counter register	425
0x4090_0038	PIAR	Periodic Interrupt Alarm register	422

10.4.1 RTC Timer Trim Register (RTTR)

RTTR, shown in Table 146, configures the frequency of the 1 Hz clock. The reset value of this register (0x0000_7FFF) is such that a perfect 32.768 kHz crystal would result in a 1 Hz clock (see Section 10.3.5.1 for details on how to calculate the value for this register). The RTTR is reset only by hardware to ensure the validity of the data written into this register. RTTR[31] is provided as a lock bit. The data in RTTR can be changed only if LCK is clear. When LCK is set, only a hardware reset can clear the RTTR. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 146: RTTR Bit Definitions

	Physical Address 0x4090_000C																															
	RTTR																RTC Controller															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCK	Reserved					DEL										CK_DIV															
Reset	0	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31	R/W	LCK	Lock bit for the RTTR: 0 = Value of RTTR can be overwritten 1 = Value of RTTR can not be overwritten **NOTE:** When LCK is set, the value of RTTR can be changed only after a hardware reset.
30:26	—	—	Reserved
25:16	R/W†	DEL	Trim Delete Count The number of 32-kHz clocks to delete when clock trimming begins.
15:0	R/W†	CK_DIV	Clock Divider count The number of 32 kHz clock cycles, plus one, per 1 Hz clock cycle. The minimum value is 0x20.

† These bit fields can be written only if RTTR[LCK] is clear.

Copyright © 2009 Marvell

10.4.2 RTC Status Register (RTSR)

RTSR, shown in Table 147, contains alarm-enable bits, count-enable bits, and alarm-detect bits. The alarm-enable bits are for the timer (ALE), wristwatch (RDALE1 and RDALE2), periodic interrupt (PIALE), and stopwatch (SWALE1 and SWALE2). The RTSR also contains a 1 Hz clock edge-detection-enable bit (HZE).

The RTSR contains two count-enable bits: one for the stopwatch (SWCE) and one for the periodic interrupt (PICE). The counters increment only if their corresponding count-enable bits are set.

The RTC logic sets the alarm-detect bits if the corresponding enable bits are set and the alarm conditions are met. The alarm detect bit contents are routed to the interrupt controller where they can be enabled to cause a second-level interrupt. The alarm-detect bits are reset by writing 0b1 to the bit(s) to be cleared. In a low-power mode, an alarm-detect bit is updated if an RTC alarm is detected and the corresponding alarm-enable bit in the RTSR is set. The RTSR contains alarm-detect bits for the timer (Al), wristwatch (RDAL1 and RDAL2), stopwatch (SWAL1 and SWAL2), periodic interrupt (PIAL), and the 1-Hz clock edge-detect (HZ).

> **Note**
>
> Before enabling any RTC alarm, write the corresponding Alarm registers with valid values. Otherwise, there are spurious wakeups from these alarm sources. For example, SWAR1/2 must be programmed before SWALE1 is enabled. Also, the inactive alarm must be programmed to a value much higher than the active alarm to prevent spurious wakeups.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 147: RTSR Bit Definitions

	Physical Address 0x4090_0008																															
	RTSR																				RTC Controller											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																PICE	PIALE	PIAL	SWCE	SWALE2	SWAL2	SWALE1	SWAL1	RDALE2	RDAL2	RDALE1	RDAL1	HZE	ALE	HZ	AL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	Reserved
15	R/W	PICE	Periodic Interrupt Count Enable for RTCPICR Count Register 0 = Counter incrementing is disabled. 1 = Counter incrementing is enabled.
14	R/W	PIALE	Periodic Interrupt Alarm Enable 0 = PIALE is not enabled; PIAL is not set. 1 = PIALE is enabled; when there is a match between PIAR and RTCPICR, PIAL is set.

Table 147: RTSR Bit Definitions (Continued)

	Physical Address 0x4090_0008																															
	RTSR																RTC Controller															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																PICE	PIALE	PIAL	SWCE	SWALE2	SWAL2	SWALE1	SWAL1	RDALE2	RDAL2	RDALE1	RDAL1	HZE	ALE	HZ	AL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
13	R/WC†	PIAL	Periodic Interrupt Alarm Status 0 = Alarm not detected. 1 = Alarm match and PIALE is enabled.
12	R/W	SWCE	Stopwatch Count Enable for SWCR Count Register 0 = Counter incrementing is disabled. 1 = Counter incrementing is enabled.
11	R/W	SWALE2	Stopwatch Alarm Enable for Stopwatch Alarm 2 0 = Alarm 2 is not enabled. 1 = Alarm 2 is enabled.
10	R/WC†	SWAL2	Stopwatch Alarm 2 Status 0 = No alarm 2 has been detected. 1 = Alarm 2 is detected (SWAR2 matches SWCR) and SWALE2 is set.
9	R/W	SWALE1	Stopwatch Alarm Enable for stopwatch alarm 1: 0 = Alarm not enabled. 1 = Alarm is enabled.
8	R/WC†	SWAL1	Stopwatch Alarm 1status: 0 = Alarm detected. 1 = Alarm detected (SWAR1 matches SWCR) and SWALE1 is set.
7	R/W	RDALE2	Wristwatch Alarm Enable for wristwatch alarm 2: 0 = Alarm 2 is not enabled. 1 = Alarm 2 is enabled.
6	R/WC†	RDAL2	Wristwatch Alarm 2 status: 0 = Alarm not detected. 1 = Alarm detected (RDAR2 matches RDCR and RYAR2 matches RYCR) and RDALE2 bit is set.
5	R/W	RDALE1	Wristwatch Alarm Enable for Wristwatch Alarm 1 0 = Alarm not enabled. 1 = Alarm is enabled.
4	R/WC†	RDAL1	Wristwatch Alarm 1 Status 0 = No alarm detected. 1 = Alarm is detected (RDAR1 matches RDCR and RYAR1 matches RYCR) and RDALE1 bit is set.
3	R/W	HZE	HZ interrupt Enable 0 = Interrupt not enabled. 1 = Interrupt enabled.

Copyright © 2009 Marvell

Table 147: RTSR Bit Definitions (Continued)

	Physical Address 0x4090_0008																															
	RTSR																															
	RTC Controller																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																PICE	PIALE	PIAL	SWCE	SWALE2	SWAL2	SWALE1	SWAL1	RDALE2	RDAL2	RDALE1	RDAL1	HZE	ALE	HZ	AL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	ALE	RTC Alarm interrupt Enable 0 = Interrupt not enabled. 1 = Interrupt is enabled.
1	R/WC†	HZ	HZ rising-edge detected: 0 = No rising edge detected. 1 = Rising edge detected and HZE is set.
0	R/WC†	AL	RTC Alarm detected: 0 = No alarm detected. 1 = Alarm detected (RCNR matches RTAR) and ALE is set.

† To clear this bit, write 0b1 to it.

10.4.3 RTC Alarm Register (RTAR)

RTAR, shown in Table 148, is compared with the RCNR after each rising edge of the 1-Hz clock. If the two are equal and RTSR[ALE] is set, then RTSR[AL] is set. These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 148: RTAR Bit Definitions

	Physical Address 0x4090_0004																															
	RTAR																															
	RTC Controller																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RTMV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:0	R/W	RTMV	RTC Target Match Value — the value compared with the RTC counter.

10.4.4 Wristwatch Day Alarm Registers (RDARx)

RDARx, shown in Table 149, are compared to the RDCR and RYCR, respectively, along with the Wristwatch Year Alarm registers (RYAR1/2) after each rising edge of the 1 Hz clock. If the conditions result in a match and the corresponding wristwatch alarm-enable bit (RTSR[RDALE1/2]) is set, the RTC controller logic sets the corresponding wristwatch alarm-detect bit (RTSR[RDAL1/2]).

> **Note**
>
> Both the RDAR1/2 registers must be programmed in pairs. If only one is used, the other register must be programmed to a maximum value of 0xFFFF_FFFF to prevent any spurious wakeups. Additionally, users must first write to the RYARx registers because they are locked after the corresponding RDARx registers are written.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 149: RDARx Bit Definitions

	Physical Address 0x4090_0018 0x4090_0020											RDAR1 RDAR2										RTC Controller										
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									WOM			DOW			HOURS					MINUTES						SECONDS					
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:23	—	—	Reserved
22:20	R/W	WOM	Match value for week of month
19:17	R/W	DOW	Match value for day of week
16:12	R/W	HOURS	Match value for hours
11:6	R/W	MINUTES	Match value for minutes
5:0	R/W	SECONDS	Match value for seconds

10.4.5 Wristwatch Year Alarm Registers (RYARx)

RYARx is defined in Table 150. After each rising edge of the 1 Hz clock, these registers, along with the Wristwatch Day Alarm registers (RDAR1/2), are compared to the RYCR and RDCR, respectively. If the conditions result in a match and the corresponding wristwatch alarm-enable bit (RTSR[RDALE1/2]) is set, the RTC controller logic sets the corresponding wristwatch alarm-detect bit (RTSR[RDAL1/2]).

> **Note**
>
> Both the RYAR1/2 registers must be programmed in pairs. If only one is used, the other register must be programmed to a max value of 0xFFFF_FFFF to prevent any spurious wakeups. Additionally these registers should be written before writing to the RDARx registers as they are locked for protection once corresponding RDARx registers are written.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 150: RYARx Bit Definitions

	Physical Address																															
	0x4090_001C																															
	0x4090_0024																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved											YEAR												MONTH				DOM				
Reset	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:21	—	—	Reserved
20:9	R/W	YEAR	Match value for the year count, from 1 to 4096
8:5	R/W	MONTH	Match value for the month
4:0	R/W	DOM	Match value for the day of month

10.4.6 Stopwatch Alarm Registers (SWARx)

SWARx, shown in Table 151, are compared to the SWCR after each rising edge of the 1 Hz clock. If there is a match and the corresponding stopwatch alarm-enable bit (RTSR[SWALE1/2]) is set, the RTC logic sets the corresponding stopwatch alarm-detect bit (RTSR[SWAL1/2]). The stopwatch 100th counter is clocked with a 100-Hz clock signal and increments at each rising edge of the 100-Hz clock signal. The stopwatch seconds, minutes and hours are clocked based on the trimmer 1-Hz clock. Because the 100th and seconds fields of the stopwatch are not clocked by the same source the 100th value is not completely accurate.

Note

Both the SWAR1/2 registers must be programmed in pairs. If only one is used, the other register must be programmed to a max value of 0xFFFF_FFFF to prevent any spurious wakeups.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 151: SWARx Bit Definitions

	Physical Address																															
	0x4090_002C																															
	0x4090_0030																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HOURS					MINUTES						SECONDS						HUNDRETHS						
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:24	—	—	Reserved

Table 151: SWARx Bit Definitions (Continued)

	Physical Address 0x4090_002C SWAR1 0x4090_0030 SWAR2																																RTC Controller
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HOURS					MINUTES						SECONDS						HUNDRETHS						
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
23:19	R/W	HOURS	Match value for the stopwatch time in hours
18:13	R/W	MINUTES	Match value for the stopwatch time in minutes
12:7	R/W	SECONDS	Match value for the stopwatch time in seconds
6:0	R/W	HUNDRETHS	Match value for the stopwatch time in hundredths of a second

10.4.7 Periodic Interrupt Alarm Register (PIAR)

PIAR, shown in Table 152, is compared to the RTCPICR after each rising edge of the 1 kHz clock. If their contents are equal and RTSR[PIALE] is set, RTSR[PIAL] is set and the periodic interrupt counter and RTCPICR are reset to zero. This process repeats as long as the count-enable RTSR[PICE] remains set. Any Write to the PIAR clears the periodic-interrupt counter and RTCPICR. The RTSR[PICE] bit must be enabled only after PIAR is written with new data. There must be a delay of at least two CPU cycles between the two actions.

Note

Zero is a non-valid value for the PIAR and yields unpredictable results. The maximum value that can be written is 65535.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 152: PIAR Bit Definitions

	Physical Address 0x4090_0038 PIAR																																RTC Controller
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																MILLISECONDS																
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	Reserved
15:0	R/W	MILLISECONDS	This value is compared with the Periodic Interrupt counter.

Copyright © 2009 Marvell

10.4.8 RTC Counter Register (RCNR)

RCNR, shown in Table 153, reflects the current value of the RTC counter. RCNR is incremented at each rising edge of a 1 Hz clock signal, which is obtained after trimming the approximate 32 kHz clock signal in the trimmer module. A crystal clock oscillator provides the 32.768 kHz clock. These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 153: RCNR Bit Definitions

Physical Address 0x4090_0000																																	
RCNR																																	
RTC Controller																																	
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RCV																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:0	R/W	RCV	Count Value Current value of the RTC counter.

10.4.9 RTC Day Counter Register (RDCR)

RDCR, shown in Table 154, reflects the current time in seconds, minutes, hours, day-of-the-week, and week-of-the-month. The Week-Of-Month (WOM) and Day-Of-Week (DOW) fields are used together to represent the current day and week of the month. For example: DOW = 3 indicates Tuesday and WOM = 1 indicates the first week of the month, so together this indicates Tuesday of the first week of the month. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 154: RDCR Bit Definitions

Physical Address 0x4090_0010																																	
RDCR																																	
RTC Controller																																	
User Settings																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									WOM			DOW			HOURS					MINUTES						SECONDS					
Reset	?	?	?	?	?	?	?	?	?	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:23	—	—	Reserved
22:20	R/W	WOM	Week of Month 1 (first week) through 5 (fifth week)
19:17	R/W	DOW	Day of Week 1 (Sunday) through 7 (Saturday)
16:12	R/W	HOURS	Hours 0 to 23
11:6	R/W	MINUTES	Minutes 0 to 59

Table 154: RDCR Bit Definitions (Continued)

Physical Address 0x4090_0010																																			
RDCR																																			
RTC Controller																																			
User Settings																																			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	Reserved									WOM			DOW			HOURS					MINUTES						SECONDS								
Reset	?	?	?	?	?	?	?	?	?	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Bits	Access	Name	Description
5:0	R/W	SECONDS	Seconds 0 to 59

10.4.10 RTC Year Counter Register (RYCR)

RYCR, shown in Table 155, contains the current time in the day-of-the-month, month, and year. The year count starts at 0 and ends at 4095. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 155: RYCR Bit Definitions

Physical Address 0x4090_0014																																			
RYCR																																			
RTC Controller																																			
User Settings																																			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	Reserved											YEAR												MONTH				DOM							
Reset	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1			

Bits	Access	Name	Description
31:21	—	—	Reserved
20:9	R/W	YEAR	Current Year 0 to 4095.
8:5	R/W	MONTH	Month 1 (January) through 12 (December).
4:0	R/W	DOM	Day of Month Starts with 1. Zero is not allowed. The maximum count varies, based on the month and the type of year. Table 144 lists the allowable values.

10.4.11 Stopwatch Counter Register (SWCR)

SWCR, shown in Table 156, contains the elapsed time of the stopwatch in hours, minutes, seconds, and hundredths of a second. SWCR increments only when the count-enable bit (RTSR[SWCE]) is set so that the counter continuously counts. When the counter reaches the maximum possible time (24 hours minus 1/100sec), it reverts to zero and starts counting again. SWCR can be programmed for use as a stopwatch or to generate an interval interrupt. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 156: SWCR Bit Definitions

	Physical Address 0x4090_0028																															
	SWCR																				RTC Controller											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HOURS					MINUTES						SECONDS						HUNDRETHS						
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:24	—	—	Reserved
23:19	R/W	HOURS	Number of elapsed hours in the stopwatch counter
18:13	R/W	MINUTES	Number of elapsed minutes in the stopwatch counter
12:7	R/W	SECONDS	Number of elapsed seconds in the stopwatch counter
6:0	R/W	HUNDRETHS	Hundreths of a second in the stopwatch counter

10.4.12 Periodic Interrupt Counter Register (RTCPICR)

RTCPICR, shown in Table 157, contains the current count of the 1 kHz periodic interrupt counter and is clocked on the positive edge of the 1 kHz clock. The RTCPICR increments only when the periodic interrupt count enable bit (RTSR[PICE]) is set. If RTSR[PICE] is disabled, the periodic interrupt counter stops incrementing at the value last reached. If re-enabled, the counter resumes counting from the value it stopped at previously.

When the RTCPICR equals the PIAR and RTSR[PIALE] is set, RTSR[PIAL] is set by the RTC logic and the periodic interrupt counter rolls over to zero. This process repeats as long as RTSR[PICE] is set. When a new session (count started from zero) begins, the periodic interrupt counter must be cleared by executing a Write to the PIAR. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 157: RTCPICR Bit Definitions

	Physical Address 0x4090_0034																															
	RTCPICR																				RTC Controller											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																MILLISECONDS															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	Reserved
15:0	R/W	MILLISECONDS	Periodic interrupt time in milliseconds

Copyright © 2009 Marvell

11 Operating System Timers

The operating-system timers block provides a set of timer channels for use by software to generate timed interrupts (or wakeup events). These interrupts are generated by two sets of timer channels. The first set, which provides one counter and four OS Match registers, are clocked from a 3.25 MHz clock. The second set, which provides eight counters and eight OS Match registers, can be clocked from the 32.768 KHz timer clock or a 13 MHz clock or an externally-supplied clock, providing a wide range of timer resolutions.

11.1 Features

The first timer set provides the following features:

- Single counter operating at 3.25 MHz
- Four OS Match registers
- Watchdog function (operates only in S0/D0/C0 and S0/D0/C1)

The second timer set provides eight additional timer channels with the following features:

- Eight independent channels, each consisting of:
  - Counter
  - Match register
  - Control register
- Software-selectable, independent clock for each counter:
  - 32.768-KHz clock for low power
  - 13 MHz clock for high accuracy
  - Externally supplied clock for network synchronization
- Counter resolutions of 1 second, 1 millisecond, 1/32768th of a second, and 1 microsecond
- SSP Frame Detect or from UDC (USB 1.1 client port) can be used as the counter resolution
- Periodic and one-shot timers
- Two external synchronization events
- Operation during low-power modes (S0/D1/C2, S0/D2/C2 and S2/D3/C4)

11.2 Signals

This section describes the external signals of the timers. Table 158 summarizes the signals.

Table 158: Operating System Timers Interface Signals Summary

Name	Type	Description
EXT_SYNC<1:0>	Input	External synchronization signals used to reset timers. When OMCRx[S] is enabled for either of these signals, the corresponding OSCRx register is reset when a rising edge is detected on the appropriate EXT_SYNC signal. (See Table 160 and Table 167.)
CLK_EXT	Input	External Clock Input

Table 158: Operating System Timers Interface Signals Summary (Continued)

Name	Type	Description
CHOUT<1:0>	Output	CHOUT1 is a periodic output clock generated from channel 11. CHOUT0 is a periodic output clock generated from channel 10. These output signals generate output clocks from timer channels 11 and 10, respectively. If the corresponding channel is programmed as periodic, the signal changes state each time a match occurs. These signals stay low if the associated channel is programmed as non-periodic.

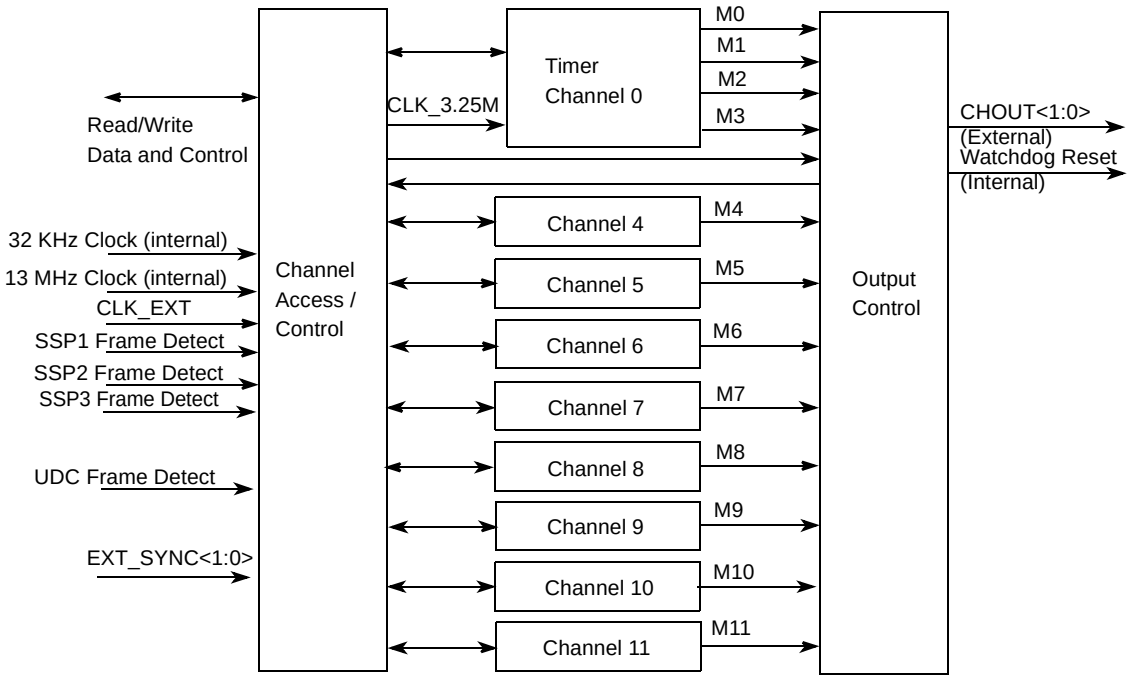
11.3 Operation

This section describes the operation of the operating-system timers block. Throughout this section, the terms *compare* and *match* describe when a counter register, OSCRx, is being compared to an OS Match register, OSMRx. A *compare* occurs at the rising edge of every corresponding channel clock. A *match* occurs when a compare is performed and the value in OSCRx is the same as the value in OSMRx. A match triggers an interrupt if the corresponding bit is set in the OIER register shown in Table 165. This module is compatible with the Marvell® PXA27x processor and partially compatible with the Marvell® PXA25x processor, as described in Section 11.3.3.

11.3.1 Block Diagram

Figure 27 is a block diagram of the OS timers block.

Figure 27: Operating System Timers Block Diagram



Copyright © 2009 Marvell

11.3.2 Compares and Matches

Each of the OS Timer Counter registers (OSCRx) is incremented on the rising edge of the corresponding clock. The OSCRx register is compared with the appropriate OS Timer Match register (OSMRx). When a match occurs, the match bit in the OS Timer Status register (OSSR) is set only if the corresponding interrupt-enable bit is set in the OS Timer Interrupt Enable register (OIER). There are two exceptions to this rule:

- OMCRx[C] is cleared. If OMCRx[C] is not set (see Table 160), OSMR[7:4] are compared to OSCR4 to set the corresponding bit in OSSR[M7:M4] and generate an interrupt if the corresponding interrupt-enable bit in OIER is set. Similarly, OSMR[11:8] are compared to OSCR8 to set OSSR[M11:M8] and generate interrupts if the corresponding interrupt-enable bits are set in OIER.
- Watchdog reset. When the watchdog match-enable bit OWER[WME] is set, OSCR0 compares to OSMR3. If a match is detected, a reset is applied to the processor and most internal states are cleared. The watchdog reset does not set OSSR[M3]. The interrupt-enable bit (OIER[E3]) does not need to be set for a match to generate a watchdog reset.

For Counter channels 5, 6, 7, 9, 10, and 11, when OMCRx[C] is clear, the corresponding OSCRx register is not incremented. For these channels, when OMCRx[C] is set, a Write to the corresponding OSCRx register starts the channel. For Channel 4 and Channel 8 only, counter operation is as if OMCRx[C] is always set.

11.3.3 Marvell® PXA25x Processor Compatibility

The first counter register (OSCR0) and four OS Match registers (OSMR[0:3]) are backwards-compatible with the PXA25x processor. The watchdog-reset functionality is also carried over. However, the input clock used to increment the OSCR0 counter has changed. For the PXA25x processor, this clock is 3.6864 MHz. For the PXA30x and PXA31x processors, the clock frequency is 3.25 MHz. Recalculate any time periods that must be exact.

11.3.4 Timer Channels

In addition to the OSCR0 and four OS Match registers OSMR[3:0], the processor provides eight additional timer channels (OSCR[11:4]) with a wide range of counter resolutions, interval and periodic timers, synchronization features, the ability to generate an output waveform, and low-power mode operability.

11.3.5 Counter Resolutions

The following sections discuss the possible resolutions for each counter register in the OS timer.

11.3.5.1 Clock Generation for Counter 0 Register

The OSCR0 counter register always uses the rising edge of the 3.25 MHz clock to increment. This clock is generated in the channel access/control block by dividing the 13 MHz input clock by four.

11.3.5.2 Clock Generation for Channels 4–7

Each of the four counters (OSCR[7:4]) can be configured to run at the following speeds:

- 32.768 KHz (1/32768 of a second). This is the 32.768 KHz clock.
- 1 Hz (1 second). This clock is generated using the 32.768 KHz clock.
- 1 KHz (1 millisecond). This clock is generated using the 32.768 KHz clock, and it runs at approximately 1 mS. The interval between “ticks” *average* 1 mS, but the time between individual ticks varies because the counter resolution is derived from the 32.768 KHz clock.
- 1 MHz (1 microsecond): This clock is derived from the 13 MHz clock.
- The external clock input CLK_EXT.

11.3.5.3 Clock Generation for Channels 8–11

Each of the four counters (OSCR[8:11]) can be configured to run at the following speeds:

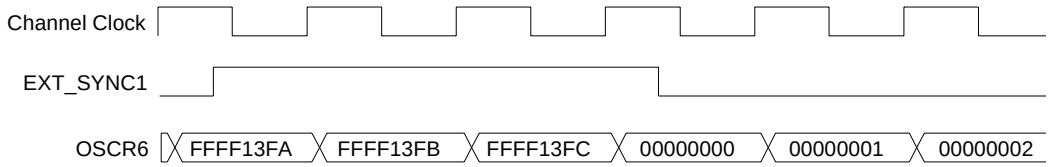
- 32.768 kHz (1/32768 of a second). This is the 32.768 kHz clock.
- 1 Hz (1 second). This clock is generated using the 32.768 kHz clock.
- 1 kHz (1 millisecond). This clock is generated using the 32.768 kHz clock and approximates 1 mS. The interval between ticks *averages* 1 mS, but the time between individual ticks varies because the counter resolution is derived from the 32.768 kHz clock.
- 1 MHz (1 microsecond). This clock is derived from the 13 MHz clock.
- External clock input CLK_EXT.
- Frame-detect signal from SSP1, SSPSFRM.
- Frame-detect signal from SSP2, SSPSFRM2.
- Frame-detect signal from SSP3, SSPSFRM3.
- Start-of-frame signal from USB 1.1 client controller (UDC).

11.3.6 External Synchronization (EXT_SYNC<1:0>)

For channel counters 4 through 11, the rising edge of either of the external synchronization input signals EXT_SYNC<1:0> can be used to reset the corresponding OSCRx when programmed through the OMCRx[S] field. The timing diagram in Figure 28 is an example:

- OMCR6[S] is programmed to 0b10 (reset on rising edge of EXT_SYNC1).
- OMCR6[CRES] is programmed to 0b001 (the 32.768 kHz clock).

Figure 28: Example: Reset of OSCR6 Based on Rising Edge of EXT_SYNC1



OSCR6 increments as shown after EXT_SYNC1 goes through an internal two-stage synchronizer. “Channel Clock” refers to the internal clock for a timer channel. The frequency of the channel clock is determined by OMCRx[CRES[3]] and OMCR[CRES]

11.3.6.1 Output Generation for CHOUT<1:0>

For Channels 11 and 10, respectively, if OMCRx[P] is set, the channel counter OSCRx is periodic. Therefore, when a match occurs, the channel counter continues counting. If OMCRx[R] is also set, the OSCRx counter resets when this match occurs. How the output pin CHOUTx is affected is summarized below:

- If both OMCRx[R] and OMCRx[P] are cleared, the counter starts at 0x0000_0000 and increments at the frequency of the channel clock until OSCRx matches the value programmed in OSMRx. Once this match occurs, the counter OSCRx stops incrementing and holds its match value. The corresponding CHOUTx output pin is held low.
- If OMCRx[R] is set and OMCRx[P] is cleared, the counter starts at 0x0000_0000 and increments at the frequency of the channel clock until OSCRx matches the value programmed in OSMRx. Once this match occurs, the counter OSCRx resets to 0x0000_0000 and stops incrementing. The corresponding CHOUTx output pin is held low.

Copyright © 2009 Marvell

- If OMCRx[R] is cleared, and OMCRx[P] is set, the counter starts at 0x0000_0000 and increments at the frequency of the channel clock until OSCRx matches the value programmed in OSMRx. Upon finding this match, CHOUTx is inverted, and the counter OSCRx continues counting. When the counter reaches 0xFFFF_FFFF, it rolls over and continues counting from 0x0000_0000.
- If both OMCRx[R] and OMCRx[P] are set, the counter starts at 0x0000_0000 and increments at the frequency of the channel clock until OSCRx matches the value programmed in OSMRx. Upon finding this match, CHOUTx is inverted, the counter OSCRx is reset to 0x0000_0000, and incrementing of OSCRx resumes.

11.3.7 Snapshot Mode

Snapshot mode is enabled by setting OMCRx[N] for Channels 11 and 9. When a Read operation of OSCR11 is performed in Snapshot mode, a snapshot of the value of OSCR10 is taken and is transferred to OS Timer Snapshot register, OSNR. Similarly, if Snapshot mode is enabled for Channel 9, a Read from OSCR9 results in copying the current value of OSCR8 to OSNR. This mode enables users to read the timer values in two channels simultaneously. Snapshot mode is available only for Channels 11 and 9.

11.3.8 Operation in Low-Power Modes

Counter operation in S0/D0CS/C0 mode is the same as in S0/D0/C0 mode. Counters are stopped but not reset when they are not operating during low-power modes. Counters continue incrementing and checking for matches when operating during low-power modes. No counters operate in S3/D4/C4 mode. The 13-MHz input clock is off in D2 and D1 mode.

Counter operation for low-power modes (S0/D1/C2, S0/D2/C2, and S2/D3/C4) is summarized as follows:

- OSCR0 does not operate during any low-power mode. Therefore, a watchdog reset cannot occur during S0/D1/C2 or S0/D2/C2 operation.
- Counter register OSCR[11:4] operates during low-power modes if the clock the counter is programmed to use (OMCRx[CRES]) is generated from the 32.768 KHz input clock, CLK_32K.
- Counter registers OSCR[11:4] do not operate during low-power modes if the clock the counter is programmed to use (OMCRx[CRES]) is generated from the 13-MHz input clock.
- Counter registers OSCR[11:4] do not operate during low-power modes if the clock the counter is programmed to use (OMCRx[CRES]) is the external-input clock, CLK_EXT.
- If CLK_32K is selected for a timer channel that is operational in low-power modes, the timer requires an extra cycle to take action based on the match. For example, if the R bit is set in OMCRx to clear the counter when match occurs, the counter increments one more time after matching the OSMR value before it is reset to 0.
- If the PCCE bit is set in Application Subsystem Clock Configuration register (ACCR), the 1 MHz clock does not start for 1 µs after exiting a low-power mode.

11.4 Register Descriptions

Table 159 shows the registers associated with the OS timers and the physical addresses used to access them. For easy reference, the summary table includes the page number of the detailed description for each register. The remainder of this section discusses each of these registers in detail.

Table 159: OS Timers Register Summary

Physical Address	Name	Description	Page
0x40A0_0000	OSMR0	OS Timer Match 0 register	page 439
0x40A0_0004	OSMR1	OS Timer Match 1 register	page 439
0x40A0_0008	OSMR2	OS Timer Match 2 register	page 439
0x40A0_000C	OSMR3	OS Timer Match 3 register	page 439
0x40A0_0010	OSCR0	OS Timer Counter register 0	page 440
0x40A0_0014	OSSR	OS Timer Status register (used for all counters)	page 442
0x40A0_0018	OWER	OS Timer Watchdog Enable register	page 440
0x40A0_001C	OIER	OS Timer Interrupt Enable register (used for all counters)	page 440
0x40A0_0020	OSNR	OS Timer Snapshot register	page 442
0x40A0_0024– 0x40A0_003C	—	Reserved	
0x40A0_0040	OSCR4	OS Timer Counter register 4	page 441
0x40A0_0044	OSCR5	OS Timer Counter register 5	page 441
0x40A0_0048	OSCR6	OS Timer Counter register 6	page 441
0x40A0_004C	OSCR7	OS Timer Counter register 7	page 441
0x40A0_0050	OSCR8	OS Timer Counter register 8	page 441
0x40A0_0054	OSCR9	OS Timer Counter register 9	page 441
0x40A0_0058	OSCR10	OS Timer Counter register 10	page 441
0x40A0_005C	OSCR11	OS Timer Counter register 11	page 441
0x40A0 0060– 0x40A0 007C	—	Reserved	
0x40A0_0080	OSMR4	OS Timer Match register 4	page 439
0x40A0_0084	OSMR5	OS Timer Match register 5	page 439
0x40A0_0088	OSMR6	OS Timer Match register 6	page 439
0x40A0_008C	OSMR7	OS Timer Match register 7	page 439
0x40A0_0090	OSMR8	OS Timer Match register 8	page 439
0x40A0_0094	OSMR9	OS Timer Match register 9	page 439
0x40A0_0098	OSMR10	OS Timer Match register 10	page 439
0x40A0_009C	OSMR11	OS Timer Match register 11	page 439

Copyright © 2009 Marvell

Table 159: OS Timers Register Summary (Continued)

Physical Address	Name	Description	Page
0x40A0_00A0 –0x40A0_00BC	—	Reserved	
0x40A0_00C0	OMCR4	OS Match Control register 4	page 433
0x40A0_00C4	OMCR5	OS Match Control register 5	page 433
0x40A0_00C8	OMCR6	OS Match Control register 6	page 433
0x40A0_00CC	OMCR7	OS Match Control register 7	page 433
0x40A0_00D0	OMCR8	OS Match Control register 8	page 435
0x40A0_00D4	OMCR9	OS Match Control register 9	page 437
0x40A0_00D8	OMCR10	OS Match Control register 10	page 435
0x40A0_00DC	OMCR11	OS Match Control register 11	page 437
0x40A0_00E0– 0x40AF_FFFF	—	Reserved	

11.4.1 OS Match Control Registers (OMCRx)

OMCRx, shown in Table 160, Table 161, and Table 162, control the operation of timer Channels 4 through 11. Only Channels 4 through 11 are controlled by OMCR registers. Channel 0 is not controlled by OMCR registers. Periodic control of the timer channels using OMCRx[P] and OMCRx[R] bits of Channels 10 and 11 can, respectively, be used to drive CHOUT<1:0>. The remainder of the timer channels work the same way as Channel 10 and 11 except that they do not drive CHOUT<1:0>. These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 160: OMCR4/5/6/7 Bit Definitions

0x40A0_00C0
0x40A0_00C4
0x40A0_00C8
0x40A0_00CC

OMCR4
OMCR5
OMCR6
OMCR7

OS TImers

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	Reserved
7	R/W	C	Channel to Match Against 0 = Channel x matches against OSCR4. 1 = Channel x matches against OSCRx.

Table 160: OMCR4/5/6/7 Bit Definitions (Continued)

0x40A0_00C0 0x40A0_00C4 0x40A0_00C8 0x40A0_00CC	OMCR4 OMCR5 OMCR6 OMCR7	OS TImers

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
6	R/W	P	Periodic Timer 0 = The channel stops incrementing after detecting a *match.* 1 = The channel continues incrementing after detecting a *match.*
5:4	R/W	S	External Synchronization Control 0b00 = No external synchronization 0b01 = Reset OSCRx on the rising edge of EXT_SYNC0 0b10 = Reset OSCRx on the rising edge of EXT_SYNC1 0b11 = Reserved
3	R/W	R	Reset OSCRx on *Match* 0 = OSCRx does not reset when a *match occurs.* 1 = OSCRx does reset when a *match occurs.*
2:0	R/W	CRES	Counter Resolution 0b000 = The counter is disabled. 0b001 = 1/32768 of a second. This counter resolution is derived from the 32.768 KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b010 = 1 millisecond. The interval between “ticks” *averages* one millisecond, but the time between individual “ticks” varies because the counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b011 = 1 second. This counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b100 = 1 microsecond. This counter resolution is derived from the 13-MHz clock. Any channel using this resolution stops counting in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4 mode. • 0b101 = Externally supplied clock from CLK_EXT. The counter resolution is the clock period of the externally supplied clock. Any channel using this resolution stops counting in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4 mode. • 0b110 - 0b111 = Reserved

Copyright © 2009 Marvell

Table 161: OMCR8/10 Bit Definitions

	0x40A0_00D0 0x40A0_00D8																OMCR8 OMCR10									OS TImers							
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Reserved																							**CRES[3]**	**C**	**P**	**S**		**R**	**CRES**		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:9	—	—	Reserved
8	R/W	CRES[3]	Used in conjunction with CRES to configure the counter resolution. This is prepended (as the MSB) to CRES (OMCR8/10[2:1]).
7	R/W	C	Channel to Match Against 0 = Channel x matches against OSCR8. 1 = Channel x matches against OSCRx.
6	R/W	P	Periodic Timer 0 = The channel stops incrementing after detecting a *match.* 1 = The channel continues incrementing after detecting a *match.*
5:4	R/W	S	External Synchronization Control 0b00 = No external synchronization. 0b01 = Reset OSCRx on the rising edge of EXT_SYNC0. 0b10 = Reset OSCRx on the rising edge of EXT_SYNC1. 0b11 = Reserved
3	R/W	R	Reset OSCRx on *Match* 0 = OSCRx does not reset when a *match occurs.* 1 = OSCRx does reset when a *match occurs.*

Table 161: OMCR8/10 Bit Definitions (Continued)

	0x40A0_00D0 0x40A0_00D8																OMCR8 OMCR10								OS TImers							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																							CRES[3]	C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2:0	R/W	CRES	Counter Resolution CRES[3] is prepended (as the MSB) to the value of CRES to configure the counter resolution. 0b0000 = The counter is disabled. 0b0001 = 1/32768th of a second. This counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. 0b0010 = 1 millisecond. The interval between “ticks” *averages* one millisecond, but the time between individual “ticks” varies because the counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b0011 = 1 second. This counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b0100 = 1 microsecond. This counter resolution is derived from the 13-MHz clock. Any channel using this resolution stops counting in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b0101 = Externally supplied clock from CLK_EXT. The counter resolution is the clock period of the externally supplied clock. Any channel using this resolution stops counting in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4 mode. • 0b0110 = SSP1 Frame Detect. The counter resolution is SSP1 frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b0111 = SSP2 Frame Detect. The counter resolution is SSP2 frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b1000 = SSP3 Frame Detect. The counter resolution is SSP3 frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b1001 = UDC Frame Detect. The counter resolution is UDC (USB 1.1 client port) frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b1100–0b1111 = Reserved

Copyright © 2009 Marvell

Table 162: OMCR9/11 Bit Definitions

	0x40A0_00D4 0x40A0_00DC																								OMCR9 OMCR11										OS TImers
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Reserved																						**N**	**CRES[3]**	**C**	**P**	**S**		**R**	**CRES**		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:10	—	—	Reserved
9	R/W	N	Snapshot Mode Channel 9: 0 = Snapshot mode is disabled. 1 = Read from OSCR9 copies contents of OSCR8 to OSNR. Channel 11: 0 = Snapshot mode is disabled. 1 = Read from OSCR11 copies contents of OSCR10 to OSNR.
8	R/W	CRES[3]	Used in conjunction with CRES to configure the counter resolution. This is prepended (as the MSB) to CRES (OMCR8/10[2:1]).
7	R/W	C	Channel to Match Against 0 = Channel x matches against OSCR8. 1 = Channel x matches against OSCRx.
6	R/W	P	Periodic Timer 0 = The channel stops incrementing after detecting a *match.* 1 = The channel continues incrementing after detecting a *match.*
5:4	R/W	S	External Synchronization Control 0b00 = No external synchronization 0b01 = Reset OSCRx on the rising edge of EXT_SYNC0. 0b10 = Reset OSCRx on the rising edge of EXT_SYNC1. 0b11 = Reserved
3	R/W	R	Reset OSCRx on *Match* 0 = OSCRx does not reset when a *match occurs.* 1 = OSCRx does reset when a *match occurs.*

Table 162: OMCR9/11 Bit Definitions (Continued)

0x40A0_00D4 0x40A0_00DC																									OMCR9 OMCR11										OS TImers
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																						N	CRES[3]	C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2:0	R/W	CRES	Counter Resolution. CRES[3] is prepended (as the MSB) to the value of CRES to configure the counter resolution. • 0b0000 = The counter is disabled. • 0b0001 = 1/32768th of a second. This counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b0010 = 1 millisecond. The interval between “ticks” *averages* one millisecond, but the time between individual “ticks” varies because the counter resolution is derived from the 32.768-KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b0011 = 1 second. This counter resolution is derived from the 32.768 KHz clock. Any channel using this resolution continues operation in all power modes except S3/D4/C4. • 0b0100 = 1 microsecond. This counter resolution is derived from the 13-MHz clock. Any channel using this resolution stops counting in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b0101 = Externally supplied clock from CLK_EXT. The counter resolution is the clock period of the externally supplied clock. Any channel using this resolution stops counting in S0/D1/C2, S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b0110 = SSP1 Frame Detect. The counter resolution is SSP1 frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b0111 = SSP2 Frame Detect. The counter resolution is SSP2 frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b1000 = SSP3 Frame Detect. The counter resolution is SSP3 frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b1001 = UDC Frame Detect. The counter resolution is UDC (USB 1.1 client port) frame-detect rate. Any channel using this resolution stops counting in S0/D2/C2, S2/D3/C4, or S3/D4/C4. • 0b1100 –0b1111 = Reserved

11.4.2 OS Timer Match Registers (OSMRx)

OSMR0–3, shown in Table 163, are compared against the OSCR0 register after every rising edge of the 3.25 MHz clock. The OSMR3 register can also serve as a watchdog timer (see Section 11.4.7).OSMR[4:11] are compared against an OS Counter register after every rising edge of the clock selected by OMCR[CRES3] and OMCR[CRES] of the appropriate OS Match Control register (see Section 11.4.1). If any of the OSMR registers match [OSCR + 1] and the corresponding interrupt-enable bit is set in the OIER register, the corresponding status bit in the OSSR is set. The status bits are routed to the interrupt controller where they generate a CPU interrupt if the

Copyright © 2009 Marvell

corresponding interrupt mask bits in the interrupt controller are enabled. These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 163: OSMR0–11 Bit Definitions

0x40A0_0000	OSMR0	OS Timers
0x40A0_0004	OSMR1	
0x40A0_0008	OSMR2	
0x40A0_000C	OSMR3	
0x40A0_0080	OSMR4	
0x40A0_0084	OSMR5	
0x40A0_0088	OSMR6	
0x40A0_008C	OSMR7	
0x40A0_0090	OSMR8	
0x40A0_0094	OSMR9	
0x40A0_0098	OSMR10	
0x40A0_009C	OSMR11	

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

Bits	Access	Name	Description
31:0	R/W	Match_Value	Match Value The value that is compared with OSCRx.

11.4.3 OS Timer Watchdog Match Enable Register (OWER)

OWER, shown in Table 164, can be programmed to generate a watchdog-reset signal. OSCR0 is compared to OSMR3 every rising edge of the 3.25 MHz clock. When OWER[WME] is set, and a match is detected, the general watchdog reset is set. The watchdog reset is then applied to the processor and most of internal states are cleared.

Follow this procedure when using OSMR3 as a watchdog timer:

- Each time the operating system services the register, read the current value of the counter and add a number to the value read. The number added corresponds to the amount of time before the next timeout. Remember to account for counter wraparound when performing these calculations.
- Write this number back to OSMR3.
- The OS must repeat this procedure periodically to correctly prevent a match from occurring. If a match occurs, the OS timer asserts the watchdog reset.

The Watchdog-Enable register OWER[WME] when set enables the watchdog function. Once the OWER[WME] is enabled, it can be disabled only by one of the reset functions (hardware, sleep, watchdog, or GPIO). Writing a zero to the enable bit after it has been set has no effect. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 164: OWER Bit Definitions

0x40A0_0018	OWER Register																															OS Timers
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																															WME
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Access	Name	Description
31:1	Reserved	—	Reserved
0	R/W	WME	Watchdog Match Enable 0 = OSMR3 *match* with OSCR0 does not cause a reset of the processor. 1 = OSMR3 *match* with OSCR0 causes a reset of the processor.

11.4.4 OS Timer Interrupt Enable Register (OIER)

OIER, shown in Table 165, contains 12 enable bits indicating whether a match between one of the OS Match Registers and an OS Counter register sets a status bit in the OSSR. Each OS Match register has a corresponding enable bit. Clearing an enable bit does not clear the corresponding interrupt status bit if that bit is already set. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 165: OIER Bit Definitions

0x40A0_001C	OIER																															OS Timers
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:12	Reserved	—	Reserved
11:0	R/W	E{*x*}	Interrupt Enable Channel *x* 0 = Do not assert OSSR[Mx] if a match occurs between OSMRx and OSCRx. 1 = Assert OSSR[x] if a match occurs between OSMRx and OSCRx.

11.4.5 OS Timer Count Register 0 (OSCR0)

OSCR0, shown in Table 166, is incremented on rising edges of the 3.25-MHz clock. The counter can be read or written at any time. This free-running counter rolls over when the maximum value is reached. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Copyright © 2009 Marvell

Table 166: OSCR0 Bit Definitions

0x40A0_ 0010	OSCR0	OS Timers

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

Bits	Access	Name	Description
31:0	R/W	Count	Count for Channel 0. Increments on every rising edge of the 3.25 MHz clock and is the count for OS match registers 0:3, which includes the watchdog function.

11.4.6 OS Timer Count Registers (OSCR[4:11])

OSCR[11:4], shown in Table 167, increment on rising edges of the clock selected by the CRES field of the appropriate Match Control register (see OS Match Control Registers (OMCRx)). A counter can be read or written at any time. These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 167: OSCR[4:11] Bit Definitions

0x40A0_0040	OSCR4
0x40A0_0044	OSCR5
0x40A0_0048	OSCR6
0x40A0_004C	OSCR7
0x40A0_0050	OSCR8
0x40A0_0054	OSCR9
0x40A0_0058	OSCR10
0x40A0_005C	OSCR11

OS Timers

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

Bits	Access	Name	Description
31:0	R/W	Count	Count for Channels 4–11 Increments on every rising edge of the clock specified by OMCRx[CRES[3]] and OMCRx[CRES].

11.4.7 OS Timer Status Register (OSSR)

OSSR, shown in Table 168, contains status bits to indicate whether a match has occurred between any of the 12 OS Match registers and the OS Counter registers. These bits are set when a match event occurs and the corresponding interrupt-enable bit is set in the OIER register. If a non-zero match value is loaded that equals the value of the corresponding OS Counter register, or a Counter register is loaded with non-zero a value that equals the corresponding OS Match register, the status bit is set immediately before the next rising clock edge for the counters if the respective OIER(*x*) is set to 1. The OSSR bits are cleared by writing a one to the proper bit position. Writing a zero to any bit locations in this register has no effect.

Match registers 0–3 generate separate interrupts to the interrupt controller but OS Match registers 4 to 11 are ORed together to generate a single interrupt only. Software must determine the interrupt source from the value in the OSSR register. These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 168: OSSR Bit Definitions

0x40A0_0014																																
OSSR																																
OS TImers																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:12	Reserved	—	Reserved
11:0	Read /Write 1 to Clear	M{*x*}	Match Status Channel *x* If OIER(*x*) is set, then: 0 = No OSMR*x match* has occurred since the last clear. 1 = An OSMR*x match* has occurred since the last clear.

11.4.8 OS Timer Snapshot Register (OSNR)

OSNR, shown in Table 169, contains the snapshot value of either OSCR10 or OSCR8. When Snapshot mode is enabled in OMCR9 and a Read is performed from OSCR9 register, the contents of OSCR8 are copied to OSNR. Similarly, when Snapshot mode is enabled in OMCR11 and a Read is performed from OSCR11, the contents of OSCR10 are copied to OSNR. A compare of OSCR9/11 to OSNR may be off by 1 if the register pair 8/9 or 10/11 are configured exactly the same. OSNR value could be one more than OSCR9 at the time of the Read. If software performs another Read, then the values are equal.

Table 169: OSNR Bit Definitions

0x40A0_0020																																
OSNR																																
OS Timers																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Snapshot_Value																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:0	R	Snapshot_ Value	Snapshot value of Channel 10 or Channel 8.

Copyright © 2009 Marvell

12 Performance Monitoring and Debug

This chapter describes the performance monitoring functions of the processor. Most of these functions are provided using the core performance counters so that additional functions can be monitored within the core. These functions are connected to performance monitoring events 0x80–0x87 of the application subsystem core.

There are no significant differences in the processors for the performance monitoring unit.

There are no feature differences for the performance monitoring and debug funtions among the PXA3xx Processor Family products; however, there are differences in the Multi-Function Pin Monitor Register (MFP_MONITOR) that must be accounted for.

12.1 Features

Following are the performance monitoring and debug features:

- Use of core performance counters
- Four 32-bit performance counters so that four unique events can be monitored simultaneously
- One 32-bit counter that counts core clock cycles
- Over 70 events that can be monitored within the core
- Eight processor-level events fed to the core; the core can monitor four
- Interrupt indicating an overflow on one of the counters, allowing the software routine to read and accumulate the register contents
- Counters accessible using Coprocessor 14 on the core

12.2 Software

This documentation describes in detail the events in the Marvell Performance Data Collector application software that directly uses the performance monitoring and debug unit. Refer to the *Users Guide - PXA3xx Processor Family - Marvell Performance Data collector for Linux* and *Users Guide - Marvell Performance Data Collector for Microsoft Windows Mobile 6*.

12.3 Signals

I/O signals are not associated with the performance monitoring unit.

12.4 Operation

All counters and control registers are accessible through coprocessor 14 from the core using MRC and MCR operations. The core uses the following registers to control the operation of the counters:

- Performance Monitor Control register (PMNC)
- Interrupt Enable register (INTEN)
- Overflow Flag Status register (FLAG)
- Event Select register (EVTSEL)

These registers are described in Chapter 10, *Manzano Core External Architecture Specification* (EAS). The EVTSEL register is used to select which of the eight processor-level events are counted by each of the four counters. Only eight events can be supplied to the core at a time (of which only four can be monitored).

Note

Performance can be monitored only while the core is active and the clocks to the core are running. No performance monitoring is performed in low-power modes including 60 MHz (D0CS) run/idle modes.

12.4.1 Processor-Level Performance Events

The events listed in Table 170 are produced and can be connected to the eight event signals supplied to the core. These events are connected to performance monitoring events 0x80–0x87 of the processor core. Note that only four events can be simultaneously monitored through software.

Table 170: ASSP Processor Performance Monitor Events

Event Number	ASSP Event Description
0x00	Core is performing a new instruction fetch Number of new core instruction fetches
0x01	Core is performing a new data fetch Number of new core data fetches
0x02	Core read request count Number of new core read requests
0x03	Reserved
0x04	Reserved
0x05	Reserved
0x06	Reserved
0x07	Reserved
0x08	Reserved
0x09	System bus 1 bus usage Number of cycles the system bus 1 bus is occupied
0x0A	System bus 2 bus usage Number of cycles the system bus 2 bus is occupied
0x0B	Reserved
0x0C	Reserved
0x0D	Reserved
0x0E	Dynamic memory queue occupied Number of cycles when the dynamic memory controller queue is not empty
0x0F	Dynamic memory queue occupied by more than one request Number of cycles when the dynamic memory controller queue has two or more requests

Copyright © 2009 Marvell

Table 170: ASSP Processor Performance Monitor Events (Continued)

Event Number	ASSP Event Description
0x10	Dynamic memory queue occupied by more than two requests Number of cycles when the dynamic memory controller queue has three or more requests
0x11	Dynamic memory queue occupied by more than three requests Number of cycles when the dynamic memory controller queue is full
0x12	Static memory queue occupied Number of cycles when the static memory controller queue is not empty
0x13	Static memory queue occupied by more than one request Number of cycles when the static memory controller queue has two or more requests
0x14	Static memory queue occupied by more than two requests Number of cycles when the static memory controller queue has three or more requests
0x15	Static memory queue occupied by more than three requests Number of cycles when the static memory controller queue is full
0x16	Reserved
0x17	Reserved
0x18	Reserved
0x19	Reserved
0x1A	Internal SRAM memory queue occupied Number of cycles when the internal memory controller queue is not empty
0x1B	Internal SRAM memory queue occupied by more than one request Number of cycles when the internal memory controller queue has two or more requests
0x1C	Internal SRAM memory queue occupied by more than two requests Number of cycles when the internal memory controller queue three or more requests
0x1D	Internal SRAM memory queue occupied by more than three requests Number of cycles when the internal memory controller queue is full
0x1E	Number of cycles when External Memory controller bus is occupied
0x1F	Number of cycles when External Data Flash bus is occupied
0x20	Reserved
0x21	Reserved
0x22	Reserved
0x23	Reserved
0x24	Core write access count Number of writes from the core
0x25	Reserved
0x26	Reserved

Table 170: ASSP Processor Performance Monitor Events (Continued)

Event Number	ASSP Event Description
0x27	Reserved
0x28	Reserved
0x29	Reserved
0x2A	System bus 1 bus request Length of time that at least one bus request is asserted on system bus 1
0x2B	System bus 2 bus request Length of time that at least one bus request is asserted on system bus 2
0x2C	System bus 1 bus retries Number of retries on system bus 1
0x2D	System bus 2 bus retries Number of retries on system bus 2
0x2E	Temperature level 1 Time the processor spends in temperature range 1
0x2F	Temperature level 2 Time the processor spends in temperature range 2
0x30	Temperature level 3 Time the processor spends in temperature range 3
0x31	Temperature level 4 Time the processor spends in temperature range 4
0x32	Core read/write latency measurement Amount of time when core has more than one read/write request outstanding
0x33	Core read/write latency measurement Amount of time when core has more than two read/write requests outstanding
0x34	Core read/write latency measurement Amount of time when core has more than three read/write requests outstanding
0x35	Core read/write latency measurement Amount of time when core has more than four read/write requests outstanding
0x36	System bus 1 to internal memory read/write latency measurement Amount of time when system bus 1 to internal memory has more than one read/write requests outstanding
0x37	System bus 1 to internal memory read/write latency measurement Amount of time when system bus 1 to internal memory has more than two read/write requests outstanding
0x38	System bus 1 to internal memory read/write latency measurement Amount of time when system bus 1 to internal memory has more than three read/write requests outstanding
0x39	System bus 1 to internal memory read/write latency measurement Amount of time when system bus 1 to internal memory has more than four read/write requests outstanding

Copyright © 2009 Marvell

Table 170: ASSP Processor Performance Monitor Events (Continued)

Event Number	ASSP Event Description
0x3A	System bus 1 to dynamic/static memory read/write latency measurement Amount of time when system bus 1 to dynamic/static memory has more than one read/write requests outstanding
0x3B	System bus 1 to dynamic/static memory read/write latency measurement Amount of time when system bus 1 to dynamic/static memory has more than two read/write requests outstanding
0x3C	System bus 1 to dynamic/static memory read/write latency measurement Amount of time when system bus 1 to dynamic/static memory has more than three read/write requests outstanding
0x3D	System bus 1 to dynamic/static memory read/write latency measurement Amount of time when system bus 1 to dynamic/static memory has more than four read/write requests outstanding
0x3E	System bus 2 to internal memory read/write latency measurement Amount of time when system bus 2 to internal memory has more than one read/write requests outstanding
0x3F	System bus 2 to internal memory read/write latency measurement Amount of time when system bus 2 to internal memory has more than two read/write requests outstanding
0x40	System bus 2 to internal memory read/write latency measurement Amount of time when system bus 2 to internal memory has more than three read/write requests outstanding
0x41	System bus 2 to internal memory read/write latency measurement Amount of time when system bus 2 to internal memory has more than four read/write requests outstanding
0x42	System bus 2 to dynamic/static memory read/write latency measurement Amount of time when system bus 2 to dynamic/static memory has more than one read/write requests outstanding
0x43	System bus 2 to dynamic/static memory read/write latency measurement Amount of time when system bus 2 to dynamic/static memory has more than two read/write requests outstanding
0x44	System bus 2 to dynamic/static memory read/write latency measurement Amount of time when system bus 2 to dynamic/static memory has more than three read/write requests outstanding
0x45	System bus 2 to dynamic/static memory read/write latency measurement Amount of time when system bus 2 to dynamic/static memory has more than four read/write requests outstanding
0x46	Reserved

Copyright © 2009 Marvell

12.4.2 Core-Level Performance Events

The events listed in Table 2 describes the low-level core events that can be monitored.

Table 171: Core Processor Performance Monitor Events

Event Number	Core Event Description
0x00	L1 Instruction Cache miss Number of L1 Instruction Cache misses
0x01	L1 Instruction Cache non-delivery Number of L1 Instruction Cache non-deliveries
0x02	Data dependency stall Number of Data dependency stalls
0x03	Instruction Translation Lookaside Buffer (TLB) miss Number of Instruction TLB misses
0x04	Data Translation Lookaside Buffer (TLB) miss Number of Data TLB misses
0x05	Branch Instruction retired Number of Branch Instructions retired
0x06	Branch Misdirection Number of Branch Mispredictions
0x07	Instruction retired Number of Instructions retired
0x08	Data Buffer stall duration Number of cycles for the Data Buffer is stalled
0x09	Data Buffer stall Number of Data Buffer stalls
0x0A	Data Cache access Number of Data Cache accesses
0x0B	Data Cache miss Number of Data Cache misses
0x0C	Data Cache write-back Number of Data Cache write-backs
0x0D	Program Counter (PC) software changes Number of PC software changes
0x0E	All Branch Instructions retired Number of All Branch Instructions retired
0x0F	Instruction Issue Cycle Number of Instruction Issues
0x10	Reserved
0x11	Reserved

Copyright © 2009 Marvell

Table 171: Core Processor Performance Monitor Events (Continued)

Event Number	Core Event Description
0x12	Reserved
0x13	Reserved
0x14	Reserved
0x15	Reserved
0x16	Reserved
0x17	Reserved
0x18	Reserved
0x19	Pipeline Flush Number of Pipeline Flushes
0x1A	All Program Counter (PC) changes Number of All PC changes
0x1B	Multiplier in use Number of cycles Multiplier in use
0x1C	Multiplier stalled pipeline Number of stalled cycles in Multiplier pipeline
0x1D	Coprocessor stalled pipeline Number of cycles when the coprocessor stalls the pipeline
0x1E	Data cache stalled pipeline Number of cycles when the data cache stalls the pipeline
0x1F	Reserved
0x20	Unified L2 cache request (only available for PXA320) Number of unified L2 Cache requests
0x21	Reserved
0x22	Reserved
0x23	Unified L2 cache miss Number of Unified L2 Cache misses
0x24-0x3F	Reserved
0x40	Address bus transaction Number of Address bus transactions
0x41	Self-initiated address bus transaction Number of initiated address bus transactions
0x42	Reserved
0x43	Reserved

Table 171: Core Processor Performance Monitor Events (Continued)

Event Number	Core Event Description
0x44	Reserved
0x45	Reserved
0x46	Reserved
0x47	Self Initiated data bus transaction Number of self initiated data bus transactions
0x48	Data bus transaction Number of data bus transactions

12.4.3 Debug Functionality

The basic functionality in the performance monitoring module is control and support of the debug functions. Registers configure the interaction between units. Various events in different parts of the system can be coordinated. Therefore, an event in one part of the system can force debug operations in all other parts of the system.

Several different parts of the system can detect an event and be combined to form an overall operation:

- Intel XScale® core abrupt-stop event (XCS)
- Intel XScale® core debug event (XDB)
- 2-D graphics abrupt-stop event (2-D graphics CS event)
- 2-D graphics debug event (2-D graphics DB event)

These events can be combined in various ways to get one or more different results. Typically, there is an abrupt-stop event that causes an unrecoverable clock stop of the system/unit and a debug event that places the system/unit is a continued debug state (for example, provokes a breakpoint routine to be run on the core). These are regarded as separate occurrences. Each result is separately programmed and more than one result can occur simultaneously if so programmed:

- Intel XScale® breakpoint
- 2-D graphics breakpoint

A separate register is provided for each of these output results and a bit within this register is associated with each input event. Thus, any input event can be made to provoke one or more output results.

12.5 Register Descriptions

The performance monitor contains registers associated with two functions:

- Performance monitoring
- JTAG access

12.5.1 Register Summary

Table 172 summarizes the performance monitoring and multi-core debug registers. The remainder of this section discusses each of these registers in detail. For easy reference, the summary table includes the page number of the detailed description for each register.

Copyright © 2009 Marvell

Table 172: Performance Monitoring and Multicore Debug Register Summary

Physical Address	Name	Description	Page
0x4600_FF00–0x4600_FF1C	PML_ESL[7:0]	Event Select registers	page 451
0x4600_FF40	MDU_CPU_BP	MDU XScale® Breakpoint register	page 452
0x4600_FF44	—	Reserved	—
0x4600_FF48	—	Reserved	—
0x4600_FF4C	—	Reserved	—
0x4600_FF50	—	Reserved	—
0x4600_FF54	MDU_2DG_EVENT	MDU 2DG Stop register	page 453
0x4600_FF58	MDU_CW_MATCH	MDU CW Match Signal register	page 454
0x4600_FF58	—	Reserved	—
0x4600_FF5C	—	Reserved	—
0x4600_FF80	—	Reserved	—

12.5.1.1 Event Select Registers (PML_ESL_[7:0])

PML_ESL_[7:0], described in Table 173, are used to select which event is supplied to each of the eight input-events to the core. These registers are identical and refer to each of the eight event inputs. Thus, it is possible (though not very useful) to monitor the same event on all eight inputs. For most accurate results, set up this register before the monitoring is enabled in the core. However, be aware that this count is approximate, and absolute accuracy is not possible for all registers.

Table 173: PML_ESEL_[7:0] Bit Definitions

(Sheet 1 of 2)

Physical Address		
0x4600_FF00	PML_ESEL_0	PML/MDU Module
0x4600_FF04	PML_ESEL_1	
0x4600_FF08	PML_ESEL_2	
0x4600_FF0C	PML_ESEL_3	
0x4600_FF10	PML_ESEL_4	
0x4600_FF14	PML_ESEL_5	
0x4600_FF18	PML_ESEL_6	
0x4600_FF1C	PML_ESEL_7	

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																									EVENT NUMBER						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	Reserved	Reserved

Table 173: PML_ESEL_[7:0] Bit Definitions (Continued)

(Sheet 2 of 2)

Physical Address		
0x4600_FF00	PML_ESEL_0	PML/MDU Module
0x4600_FF04	PML_ESEL_1	
0x4600_FF08	PML_ESEL_2	
0x4600_FF0C	PML_ESEL_3	
0x4600_FF10	PML_ESEL_4	
0x4600_FF14	PML_ESEL_5	
0x4600_FF18	PML_ESEL_6	
0x4600_FF1C	PML_ESEL_7	

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																									EVENT NUMBER						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
6:0	W	EVENT NUMBER	Selects which event from the PML event table (Table 170) is to be sent to the core event input for this register (PML_ESEL_0 refers to input 0, PML_ESEL_1 refers to input 1, and so on)

12.5.2 Processor Debug Unit (MDU) Configuration Registers

One register is provided for each destination event. All registers are identical, even though some combinations may not provide any useful function.

12.5.2.1 MDU Intel XScale® Breakpoint Register (MDU_CPU_BP)

MDU_XSCALE_BP, shown in Table 174, causes the application subsystem core to perform a breakpoint operation if triggered.

Table 174: MDU_XSCALE_BP Bit Definitions

Physical Address		
0x4600_FF40	MDU_CPU_BP	PML/MDU Module

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																						Reserved	XCS EVENT	XDB EVENT	Reserved	Reserved	Reserved	2DG CS EVENT	2DG DB EVENT	Reserved	Reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	0	0	?	?

Bits	Access	Name	Description
31:9	—	Reserved	Reserved
8	W	XCS EVENT	0 = Not allowed. 1 = An XScale® abrupt-stop event can provoke the output result.

Copyright © 2009 Marvell

Table 174: MDU_XSCALE_BP Bit Definitions (Continued)

	Physical Address 0x4600_FF40												MDU_CPU_BP										PML/MDU Module									
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																						Reserved	XCS EVENT	XDB EVENT	Reserved	Reserved	Reserved	2DG CS EVENT	2DG DB EVENT	Reserved	Reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	0	0	?	?

Bits	Access	Name	Description
7	W	XDB EVENT	0 = Not allowed. 1 = An XScale® debug event can provoke the output result.
6:4	—	Reserved	Reserved
3	W	2DG CS EVENT	0 = Not allowed. 1 = A 2-D graphics subsystem abrupt-stop event can provoke the output result.
2	W	2DG DB EVENT	0 = Not allowed. 1 = A 2-D graphics subsystem debug event can provoke the output result.
1:0	—	Reserved	Reserved

12.5.2.2 MDU 2DG Stop Register (MDU_2DG_EVENT)

Table 175: MDU_2DG_EVENT Bit Definitions

	Physical Address 0x4600_FF54												MDU_2DG_EVENT										PML/MDU Module									
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																						Reserved	XCS EVENT	XDB EVENT	Reserved	Reserved	Reserved	2DG CS EVENT	2DG DB EVENT	Reserved	Reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	0	0	?	?

Bits	Access	Name	Description
31:9	—	Reserved	Reserved
8	W	XCS EVENT	0 = Not allowed. 1 = An XScale® abrupt-stop event can provoke the output result.

Table 175: MDU_2DG_EVENT Bit Definitions (Continued)

Physical Address 0x4600_FF54	MDU_2DG_EVENT	PML/MDU Module

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved																						Reserved	XCS EVENT	XDB EVENT	Reserved	Reserved	Reserved	2DG CS EVENT	2DG DB EVENT	Reserved	Reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	0	0	?	?

Bits	Access	Name	Description
7	W	XDB EVENT	0 = Not allowed. 1 = An XScale® debug event can provoke the outputresult.
6:4	—	Reserved	Reserved.
3	W	2DG CS EVENT	0 = Not allowed. 1 = A 2-D graphics subsystem stop event can provoke the output.
2	W	2DG DB EVENT	0 = Not allowed. 1 = A 2-D graphics subsystem debug event can provoke the output.
1:0	—	Reserved	Reserved

12.5.2.3 MDU CW Match Signal Register (MDU_CW_MATCH)

Table 176: MDU_CW_MATCH Bit Definitions

Physical Address 0x4600_FF58	MDU_CW_MATCH	PML/MDU Module

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved																						Reserved	XCS EVENT	XDB EVENT	Reserved	Reserved	Reserved	2DG CS EVENT	2DG DB EVENT	Reserved	Reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	0	0	?	?

Bits	Access	Name	Description
31:9	—	Reserved	Reserved
8	W	XCS EVENT	0 = Not allowed 1 = An XScale® event can provoke the output
7	W	XDB EVENT	0 = Not allowed 1 = An XScale® debug event can provoke the output

Copyright © 2009 Marvell

Table 176: MDU_CW_MATCH Bit Definitions (Continued)

Physical Address 0x4600_FF58																																
MDU_CW_MATCH																																
PML/MDU Module																																
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																						Reserved	XCS EVENT	XDB EVENT	Reserved	Reserved	Reserved	2DG CS EVENT	2DG DB EVENT	Reserved	Reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	0	0	?	?

Bits	Access	Name	Description
6:4	—	Reserved	Reserved
3	W	2DG CS EVENT	0 = Not allowed 1 = A 2-D graphics subsystem stop event can provoke the output
2	W	2DG DB EVENT	0 = Not allowed 1 = A 2-D graphics subsystem debug event can provoke the output
1:0	—	Reserved	Reserved

12.5.3 Multi-Function Pin Monitor Register (MFP_MONITOR)

MFP_MONITOR, shown in Table 177, determines the source of the second-level wakeup event (ADxxR[WE_GENERIC[12]).

Table 177: MFP_MONITOR Bit Definitions

	Physical Address 0x4600_FF80												MFP_MONITOR												PML/MDU Module							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	GPIO0	GPIO1	GPIO2	GPIO3	GPIO4	GPIO9	Reserved																									
PXA31x Only	EXT_SYNC0	EXT_SYNC1	CLK_EXT	GPIO0	GPIO1	GPIO2	GPIO53	GPIO76	GPIO83	GPIO127	GPIO0_2	GPIO1_2	MMC3	Reserved																		
PXA30x Only	EXT_SYNC0	EXT_SYNC1	CLK_EXT	GPIO0	GPIO1	GPIO2	GPIO53	GPIO76	GPIO83	GPIO127	GPIO0_2	GPIO1_2	Reserved																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31	R	EXT_SYNC0	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		GPIO0	0 = No edge detected (PXA32x Only) 1 = Edge detected
30	R	EXT_SYNC1	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		GPIO1	0 = No edge detected (PXA32x Only) 1 = Edge detected
29	R	CLK_EXT	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		GPIO2	0 = No edge detected (PXA32x Only) 1 = Edge detected
28	R	GPIO0	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		GPIO3	0 = No edge detected (PXA32x Only) 1 = Edge detected
27	R	GPIO1	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		GPI04	0 = No edge detected (PXA32x Only) 1 = Edge detected

Copyright © 2009 Marvell

Table 177: MFP_MONITOR Bit Definitions (Continued)

	Physical Address 0x4600_FF80												MFP_MONITOR												PML/MDU Module							
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x Only	GPIO0	GPIO1	GPIO2	GPIO3	GPIO4	GPIO9	Reserved																									
PXA31x Only	EXT_SYNC0	EXT_SYNC1	CLK_EXT	GPIO0	GPIO1	GPIO2	GPIO53	GPIO76	GPIO83	GPIO127	GPIO0_2	GPIO1_2	MMC3	Reserved																		
PXA30x Only	EXT_SYNC0	EXT_SYNC1	CLK_EXT	GPIO0	GPIO1	GPIO2	GPIO53	GPIO76	GPIO83	GPIO127	GPIO0_2	GPIO1_2	Reserved																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
26	R	GPIO2	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		GPIO9	0 = No edge detected (PXA32x Only) 1 = Edge detected
25	R	GPIO53	0 = No edge detected (PXA30x and PXA31x Only) 1 = Edge detected
		—	Reserved (PXA32x Only)
24	R	GPIO76	0 = No edge detected 1 = Edge detected
		—	Reserved (PXA32x Only)
23	R	GPIO83	0 = No edge detected 1 = Edge detected
		—	Reserved (PXA32x Only)
22	R	GPIO127	0 = No edge detected 1 = Edge detected
		—	Reserved (PXA32x Only)
21	R	GPIO0_2	0 = No edge detected 1 = Edge detected
		—	Reserved (PXA32x Only)

Table 177: MFP_MONITOR Bit Definitions (Continued)

Physical Address 0x4600_FF80	MFP_MONITOR	PML/MDU Module

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x Only	GPIO0	GPIO1	GPIO2	GPIO3	GPIO4	GPIO9	Reserved																									
PXA31x Only	EXT_SYNC0	EXT_SYNC1	CLK_EXT	GPIO0	GPIO1	GPIO2	GPIO53	GPIO76	GPIO83	GPIO127	GPIO0_2	GPIO1_2	MMC3	Reserved																		
PXA30x Only	EXT_SYNC0	EXT_SYNC1	CLK_EXT	GPIO0	GPIO1	GPIO2	GPIO53	GPIO76	GPIO83	GPIO127	GPIO0_2	GPIO1_2	Reserved																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
20	R	GPIO1_2	0 = No edge detected 1 = Edge detected
		—	Reserved (PXA32x Only)
19	R	MMC3	0 = No edge detected (PXA31x Only) 1 = Edge detected
		—	Reserved (PXA32x and PXA30x Only)
18:0	R	—	Reserved

Copyright © 2009 Marvell

13 System Bus Arbiters

This chapter describes the internal system-bus arbitration mechanism. There are two arbiters in the processor, one on each instance of the system bus.

System Bus 1 has the following clients:

- System bus 1-to-switch connection
- DMA controller
- LCD controller
- USB host controller
- Quick Capture Interface
- NAND interface

System Bus 2 has the following clients:

- System bus 2-to-switch connection
- 2-D graphics engine
- Hardware Video Accelerator (PXA31x Only)
- USB 2.0 client

The arbiter arbitrates for bus access. Arbitration is programmed through the ARB_CNTRL_1 for System Bus 1 clients and ARB_CNTRL_2 registers for System Bus 2 clients. The buses are implemented as a multiplexer (rather than a three-state approach), and the clients can request the bus with no limitations.

There are no System Bus arbiter feature differences among the PXA32x processor, PXA31x processor, and PXA30x processor. Refer to each individual register for other operating differences.

13.1 Features

- Programmable client weights
- Software selectable bus parking
- Bus locking

13.2 Signals

No external signals are associated with either System Bus 1 or System Bus 2 arbiters.

13.3 Operation

Each arbiter is based on the same architecture with different numbers and types of requestors. Each arbiter independently uses the two registers described in Section 13.4.

13.3.1 Programmable Weights

The lower bits of the Arbiter Control registers (as shown in Table 179 and Table 180) determine the arbitration priority of the clients on the bus. The values programmed in the weight fields of the ARB_CNTRL_x registers configure the relative importance of the programmable clients on the bus. Between them, they attempt to capture the range of the transactions on the internal bus.

Initially, the weight assigned to each controller is loaded into separate counters. If all the counters have the same value, they are serviced in a round-robin priority. If the counters have different

values, the counter with the highest value is serviced first and then decremented by one. This process occurs until its value is the same as at least one of the other counters. At this time, the other counter of the same value is serviced and decremented by one. This process continues until all counters have a value of zero. Then the weight values from ARB_CNTRL_x are reloaded into the counters and the process begins again.

Note

Unlike in the Marvell PXA27x processor family, there is no need to force the core to have high performance relative to the other devices. The main core path to memory is now through the switch, and the core accesses on the system buses are mainly software configuration commands.

13.3.2 Bus Parking

Parking is designed to reduce the latency of a transaction by speculatively granting the bus to a client in anticipation of a request from that client. When the bus is not parked, all clients must request (and be granted) the bus before all transactions. Thus, all clients have an equal latency for transactions (assuming equal priority).

When the bus is parked on a client, the parked client does not have to request the bus before initiating the transaction if the bus is idle. The result is shorter latency for the parked client to initiate transactions when the bus is idle. However, all other clients incur a latency penalty of two to three core cycles when they request an access on an idle bus.

The bus is parked with a particular client by setting the park bit in the ARB_CNTRL registers, corresponding to that client. When the bit is set, the bus is granted to that client if no other client asks for the bus.

An exception to this rule occurs when overriding circumstances exist. For instance, the park directive is overridden if the LOCK_FLAG is set and the DMA controller is designated as the park client (by setting ARB_CNTRL_1[DMA_PARK]) and a SWAP operation is in progress from the core through the switch. Other conditions in the system (such as sleep and retries) can also prevent park from taking effect. In all such conditions, the bus is not parked with any client.

The bus is not given to any client if no park bits are set. If more than one park bit is set, the priority is from Bit 31 downwards.

Most peripherals have a single connection to the bus because they only request data from other peripherals or memory. For example, the LCD controller only requests data, it never writes data to memory. Thus, it acts only as a master on the bus. However, the DMA controller, USB 2.0 controller and memory switch controllers each have two connections to the buses to which they connect. These peripherals both read and write data to/from memory and therefore require both a master and slave connection to the bus. The bus can be parked with either the master or slave controllers.

Note

Parking the bus with either of the switch slave controllers is not recommended.

13.3.3 Bus Locking

The processor internal bus offers lock functionality typically used to perform an atomic operation from the core. The arbiter programming model provides flexibility in allowing other clients to access the bus while such an operation is in progress.

Copyright © 2009 Marvell

When the LOCK_FLAG is set, all bus masters except the switch interface are prevented from making new requests for the bus during the lock sequence (which can be many transactions long). If ARB_CNTRL_X[LOCK_FLAG] is clear, all bus masters can access the bus, subject to normal arbitration rules.

13.3.4 System Considerations: System Bus Access Latency

The system has seven masters all connected on the two system buses:

- Switch (application subsystem core)
- DMA controller
- LCD controller
- Quick Capture Camera Interface
- USB 1.1 full speed host
- 2-D graphics
- USB 2.0 high speed client
- *Hardware video accelerator unit (PXA31x only, which has 8 masters)

All accesses to the memories from any of these masters flow through the switch to one of the memory controllers (see Chapter 3, Internal Switch Bus). The memory controllers all have internal buffers that work as FIFOs. All requests to a particular memory area are executed in order.

Programmers can program the system-bus arbitration priorities enabling one of the masters on the bus to access the memory switch with a higher priority than others.

To compute the worst-case latency for an external-memory transfer in a very busy system, where all the system-bus masters are trying continuously to access one of the memories (assuming that the master has highest priority programmed in the arbiter), programmers must account for these transfers ahead of the current master accessing the external bus:

- One current memory access
- Up to four pending transfers in the relevant memory controller queue
- Two transfers on the system bus
- Any other transfers pending within the master
- An SDRAM refresh

Because any of the internal memory accesses generally are lower in latency than the external memory accesses, any accesses having a critical latency requirement (such as USB host isochronous-buffer data) should be placed in internal memory.

13.4 Register Descriptions

13.4.1 Register Summary

Table 178 summarizes registers used for bus arbitration.

Table 178: Internal System Bus Arbiter Register Summary

Physical Address	Description	Page
0x4600_FE00	System Bus Arbiter Control Register 1 (ARB_CNTRL_1)	page 462
0x4600_FE04	System Bus Arbiter Control Register 2 (ARB_CNTRL_2)	page 464

13.4.2 System Bus Arbiter Control Registers (ARB_CNTRL_1 and ARB_CNTRL_2)

13.4.2.1 System Bus Arbiter Control Register 1 (ARB_CNTRL_1)

ARB_CNTRL_1, defined in Table 179, configures bus parking and the peripheral priority for the peripherals on System Bus 1.

13.4.2.2 System Bus Arbiter Control Register 2 (ARB_CNTRL_2)

ARB_CNTRL_2, defined in Table 180, configures bus parking and the peripheral priority for the peripherals on System Bus 2.

Writes to these registers are immediately communicated to the arbiters.

Table 179: ARB_CNTRL_1 Bit Definitions

Physical Address 0x4600_FE00	ARB_CNTRL_1 Reset value: 0x0	Internal Bus Arbiter

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved		DMA_SLV_PARK	SWITCH_SLV_PARK	USBH_PARK	CI_PARK	LCD_PARK	DMA_PARK	SWITCH_PARK	LOCK_FLAG	Reserved						CAMERA_WT				LCD_WT				DMA_WT				SWITCH_WT			
Reset	?	?	0	0	0	0	0	0	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:30	—	Reserved	Reserved
29	R/W	DMA_SLV_PARK	DMA Slave Park 0 = Bus is not parked with the DMA slave controller when idle. 1 = Bus is parked with the DMA slave controller when idle.
28	R/W	SWITCH_SLV_PARK	System Bus 1 Switch Slave Park 0 = Bus is not parked with the switch slave controller when idle. 1 = Bus is parked with the switch slave controller when idle.
27	R/W	USBH_PARK	USB Host Park 0 = Bus is not parked with the USB host controller when idle. 1 = Bus is parked with the USB host controller when idle.
26	R/W	CI_PARK	Quick Capture Interface Park 0 = Bus is not parked with the Quick Capture interface when idle. 1 = Bus is parked with the Quick Capture interface when idle.
25	R/W	LCD_PARK	LCD Park 0 = Bus is not parked with the LCD controller when idle. 1 = Bus is parked with the LCD controller when idle.

Copyright © 2009 Marvell

Table 179: ARB_CNTRL_1 Bit Definitions (Continued)

Physical Address 0x4600_FE00	ARB_CNTRL_1 Reset value: 0x0	Internal Bus Arbiter

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
	Reserved		DMA_SLV_PARK	SWITCH_SLV_PARK	USBH_PARK	CI_PARK	LCD_PARK	DMA_PARK	SWITCH_PARK	LOCK_FLAG	Reserved						CAMERA_WT				LCD_WT				DMA_WT				SWITCH_WT			
Reset	?	?	0	0	0	0	0	0	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
24	R/W	DMA_PARK	DMA Master Park 0 = Bus is not parked with the DMA master controller when idle. 1 = Bus is parked with the DMA master controller when idle.
23	R/W	SWITCH_PARK	System Bus Switch Master Park 0 = Bus is not parked with the switch master controller when idle. 1 = Bus is parked with the switch master controller when idle. Note: Setting this bit is not recommended.
22	R/W	LOCK_FLAG	Lock Flag When a locking master has locked the bus, only that master can gain access to the bus. If no master has locked the bus, any master can access it. 0 = All bus masters gain access to the bus (subject to normal arbitration rules). 1 = Only the master that has locked the bus can access the bus.
21:16	—	Reserved	Reserved
15:12	R/W	CAMERA_WT	Camera Priority Value Values in this field determine the relative priority of camera requests for the bus from the switch (core), LCD, and DMA requests.
11:8	R/W	LCD_WT	LCD Priority Value Values in this field determine the relative priority of LCD requests for the bus from the switch (core), Quick Capture interface, and DMA requests.
7:4	R/W	DMA_WT	DMA Priority Value Values in this field determine the relative priority of DMA requests for the bus from the switch (core), Quick Capture interface, and LCD requests.
3:0	R/W	SWITCH_WT	Switch (Core) Priority Value Values in this field determine the relative priority of the switch (core) requests for the bus from the DMA controller, Quick Capture interface, and LCD requests.

Table 180: ARB_CNTRL_2 Bit Definitions

Physical Address 0x4600_FE04 | **ARB_CNTRL_2** Reset value: 0x0 | **Internal Bus Arbiter**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and and PXA30x only	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	SWITCH_PARK	LOCK_FLAG	Reserved													USB2_WT				2DG_WT				SWITCH_WT			
PXA31x only	HVAU_SLV_PARK	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	HVAU_PARK	SWITCH_PARK	LOCK_FLAG	Reserved							HVAU_WT				USB2_WT				2DG_WT				SWITCH_WT			
Reset	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	USB2_SLV_PARK	USB 2.0 Client Slave Park 0 = Bus is not parked with the USB 2.0 client slave controller when idle. 1 = Bus is parked with the USB 2.0 client slave controller when idle. **NOTE:** PXA32x and PXA30x Only
		HVAU_SLV_PARK	Hardware Video Accelerator Unit Client Slave Park 0 = Bus is not parked with the Video Accelerator slave when idle. 1 = Bus is parked with the Video Accelerator slave when idle. **NOTE:** PXA31x Only
30	—	Reserved	Reserved **NOTE:** PXA32x and PXA30x Only
	R/W	USB2_SLV_PARK	USB 2.0 Client Slave Park 0 = Bus is not parked with the USB 2.0 client slave controller when idle. 1 = Bus is parked with the USB 2.0 client slave controller when idle. **NOTE:** PXA31x Only
29	R/W	SWITCH_SLV_PARK	System Bus 2 Switch Slave Park 0 = Bus is not parked with the switch slave controller when idle. 1 = Bus is parked with the switch slave controller when idle. **NOTE:** PXA32x and PXA30x Only
	—	Reserved	Reserved **NOTE:** PXA31x Only

Copyright © 2009 Marvell

Table 180: ARB_CNTRL_2 Bit Definitions (Continued)

Physical Address 0x4600_FE04 | **ARB_CNTRL_2 Reset value: 0x0** | **Internal Bus Arbiter**

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and and PXA30x only	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	SWITCH_PARK	LOCK_FLAG	Reserved													USB2_WT				2DG_WT				SWITCH_WT			
PXA31x only	HVAU_SLV_PARK	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	HVAU_PARK	SWITCH_PARK	LOCK_FLAG	Reserved							HVAU_WT				USB2_WT				2DG_WT				SWITCH_WT			
Reset	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
28	R/W	USB2_PARK	USB 2.0 Client Master Park 0 = Bus is not parked with the USB 2.0 client master controller when idle. 1 = Bus is parked with the USB 2.0 client master controller when idle. **NOTE:** PXA32x and PXA30x Only
		SWITCH_SLV_PARK	System Bus 2 Switch Slave Park 0 = Bus is not parked with the switch slave controller when idle. 1 = Bus is parked with the switch slave controller when idle. **NOTE:** PXA31x Only
27	R/W	2DG_PARK	2-D Graphics Park 0 = Bus is not parked with the 2-D graphics controller when idle. 1 = Bus is parked with the 2-D graphics controller when idle. **NOTE:** PXA32x and PXA30x Only
		USB2_PARK	USB 2.0 Client Master Park 0 = Bus is not parked with the USB 2.0 client master controller when idle. 1 = Bus is parked with the USB 2.0 client master controller when idle. **NOTE:** PXA31x Only

Table 180: ARB_CNTRL_2 Bit Definitions (Continued)

Physical Address 0x4600_FE04 | **ARB_CNTRL_2** Reset value: 0x0 | **Internal Bus Arbiter**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and and PXA30x only	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	SWITCH_PARK	LOCK_FLAG	Reserved													USB2_WT				2DG_WT				SWITCH_WT			
PXA31x only	HVAU_SLV_PARK	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	HVAU_PARK	SWITCH_PARK	LOCK_FLAG	Reserved							HVAU_WT				USB2_WT				2DG_WT				SWITCH_WT			
Reset	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
26	R/W	SWITCH_PARK	System Bus 2 Switch Master Park 0 = Bus is not parked with the switch master controller when idle. 1 = Bus is parked with the switch master controller when idle. **NOTE:** PXA32x and PXA30x Only
		2DG_PARK	2-D Graphics Park 0 = Bus is not parked with the 2-D graphics controller when idle. 1 = Bus is parked with the 2-D graphics controller when idle. **NOTE:** PXA31x Only
25	R/W	LOCK_FLAG	Lock Flag 0 = All bus masters gain access to the bus (subject to normal arbitration rules). 1 = Only locking masters gain access to the bus. **NOTE:** PXA32x and PXA30x Only
		HVAU_PARK	Hardware Video Accelerator Unit Client Slave Park 0 = Bus is not parked with the 2-D graphics controller when idle. 1 = Bus is parked with the 2-D graphics controller when idle. **NOTE:** PXA31x Only
24	—	Reserved	Reserved **NOTE:** PXA32x and PXA30x Only
	R/W	SWITCH_PARK	System Bus 2 Switch Master Park 0 = Bus is not parked with the switch master controller when idle. 1 = Bus is parked with the switch master controller when idle. **NOTE:** PXA31x Only

Copyright © 2009 Marvell

Table 180: ARB_CNTRL_2 Bit Definitions (Continued)

Physical Address 0x4600_FE04 | **ARB_CNTRL_2 Reset value: 0x0** | **Internal Bus Arbiter**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and and PXA30x only	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	SWITCH_PARK	LOCK_FLAG	Reserved													USB2_WT				2DG_WT				SWITCH_WT			
PXA31x only	HVAU_SLV_PARK	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	HVAU_PARK	SWITCH_PARK	LOCK_FLAG	Reserved							HVAU_WT				USB2_WT				2DG_WT				SWITCH_WT			
Reset	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
23	—	Reserved	Reserved **NOTE:** PXA32x and PXA30x Only
	R/W	LOCK_FLAG	Lock Flag 0 = All bus masters gain access to the bus (subject to normal arbitration rules). 1 = Only locking masters gain access to the bus. **NOTE:** PXA31x Only
22:16	—	Reserved	Reserved
15:12	—	Reserved	Reserved **NOTE:** PXA32x and PXA30x Only
	R/W	HVAU_WT	Hardware Video Accelerator Unit Priority Value Values in this field determine the relative priority of video accelerator requests for the bus with respect to the other System Bus 2 clients. The HVAU programmable priority value configures the relative priority of internal register configuration reads and writes. This bit field does not determine the priority of the HVAU on the system bus from a video encode/decode data perspective. The HVAU has an internal bus connection directly to the internal memory address space or the external DDR SDRAM address space. During an encode or decode execution, the actual data flows across this internal bus but not through the memory switch. The only data relative to the HVAU that flows through the memory switch is the configuration register read/writes from the application subsystem core. **NOTE:** PXA31x Only

Table 180: ARB_CNTRL_2 Bit Definitions (Continued)

Physical Address 0x4600_FE04 | **ARB_CNTRL_2** Reset value: 0x0 | **Internal Bus Arbiter**

	Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings																																
PXA32x and and PXA30x only	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	SWITCH_PARK	LOCK_FLAG	Reserved													USB2_WT				2DG_WT				SWITCH_WT			
PXA31x only	HVAU_SLV_PARK	USB2_SLV_PARK	Reserved	SWITCH_SLV_PARK	USB2_PARK	2DG_PARK	HVAU_PARK	SWITCH_PARK	LOCK_FLAG	Reserved							HVAU_WT				USB2_WT				2DG_WT				SWITCH_WT			
Reset	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
11:8	R/W	USB 2.0_WT	USB 2.0 Client Priority Value Determines the relative priority of USB 2.0 client requests for the bus with respect to the other System Bus 2 clients.
7:4	R/W	2D GRAPHICS_WT	2-D Graphics Priority Value Determines the relative priority of the 2-D graphics requests for the bus with respect to the other System Bus 2 clients.
3:0	R/W	SWITCH_WT	Switch (Core) Priority Value Determine the relative priority of the switch (core) requests for the bus with respect to the other System Bus 2 clients.

Copyright © 2009 Marvell

14 JTAG

This chapter describes the JTAG features of the processor. JTAG provides a way to drive and sample the external pins of the device, regardless of the core state. Therefore, you can test both the device electrical connections to the circuit board and (in conjunction with other devices on the circuit board with a similar interface) the integrity of the circuit board connections between devices. JTAG also provides a mechanism for device debug through the *XScale® CPU debug features*.

JTAG logic includes a test-access port (TAP) controller, TAP pins, an instruction register, and test data registers (TDRs). The registers include the Boundary Scan register (BSR) to control the IO pins directly, a Bypass register, a Device Identification (ID) register, and data-specific registers, as shown in Figure 29. Data are shifted into all registers most significant bit (MSB) first serially. The JTAG interface is controlled through five dedicated TAP controller pins that interface to the TAP controller: TDI, TMS, TCK, nTRST, and TDO.

The JTAG device IDs differ among the various devices in the PXA3xx processor family, but there are no differences in the JTAG logic.

Note

Follow JTAG requirements when developing schematics and designing a board layout. Refer to the *PXA3xx Processor Design Guide* for JTAG signal connection guidelines and requirements.

14.1 Features

The JTAG interface has the following features and conformities:

- JTAG interface that conforms to the IEEE Std. 1149.1 – 1990, IEEE Std. 1149.1a-1993, *Standard Test Access Port and Boundary-Scan Architecture.*
- Support for hardware/software debug

Refer to the IEEE Std. 1149.1 for an explanation of the terms used in this section and for a complete description of the TAP controller states.

14.2 Signals

The JTAG interface is controlled through five dedicated TAP controller pins: TDI, TMS, TCK, nTRST, and TDO as described in Table 181.

Table 181: TAP Controller Pin Definitions

Name	Type	Description
TCK	Input	Clock input for the TAP controller, Instruction register, and test data registers. Limited to 3.25 MHz maximum frequency for Boundary Scan register, and 13 MHz for all other registers. There is no minimum.
TMS	Input	Controls operation of the TAP controller. TMS is sampled on the rising edge of TCK. The TMS input is pulled high when not being driven.

Table 181: TAP Controller Pin Definitions (Continued)

Name	Type	Description
TDI	Input	Serial data input to the instruction and test data registers. Data at TDI is sampled on the rising edge of TCK. TDI is pulled high by an internal pullup resistor when not being driven.
TDO	Output	Serial data output. Data at TDO is clocked out on the falling edge of TCK. It provides an inactive (high-Z) state during non-shift operations to support parallel connection of TDO outputs at the board or module level.
nTRST	Input	Provides asynchronous initialization of the JTAG test logic. Assertion of this pin puts the TAP controller in the TEST_Logic_Reset state. An external source must drive nTRST either before or at the same time as the hardware nRESET pin for correct TAP controller and device operation. The nTRST input is pulled high by an internal pullup resistor when not being driven.

14.3 Operation

This section describes the TAP controller, hardware connection, JTAG instruction set (Instruction register), test-data registers, and data-specific register.

14.3.1 TAP Controller Reset

The JTAG interface includes a TAP controller state machine. To force the TAP controller into the test-logic-reset state (TLRS) at power-on, the nTRST pin can be asserted low, or the TMS pin can be held high for five TCK cycles. Marvell recommends that nTRST be driven from low to high (de-asserted) either before or while the hardware nRESET pin is de-asserted.

Note

nTRST should not be de-asserted after nRESET.

A clock on the TCK pin is not necessary to reset the device at power-on. After power-on, the TAP controller can be reset by either holding TMS high for five TCK cycles or asynchronously through nTRST. This procedure is described in Section . Where JTAG is not used, Marvell recommends that a reset IC be used to cause a reset on nTRST at power-on.

TLRS is entered either synchronously or asychronously as follows:

1. Select system mode (BSR does *not* intercept any signals passing between the pads and the core).
2. Select the Idcode instruction. If TCK is pulsed in the shift-DR state, the contents of the Device Identification (ID) register are clocked out TDO.

Note

While nTRST is asserted, the platform binding operation is blocked. This feature applies only to the PXA31x trusted (secure) device. Refer to the *Boot ROM Reference Manual* for details on platform binding (bound platform).

Copyright © 2009 Marvell

14.3.2 Instruction Register

The Instruction register (IR) controls JTAG circuitry by selecting a specific Test Data register (TDR) to be connected between TDI and TDO and controlling the operation affecting that data. These tasks are accomplished through a set of predefined instructions that can be either mandatory or optional as set forth in the IEEE Std. 1149.1a-1993, user defined, or private. The IR is 11 bits wide.

Three mandatory public instructions are managed: extest, sample/preload, and bypass. Three optional public instructions are managed: idcode, clamp, and highz. Finally, four user-defined instructions are managed: dbgrx, load_sram, dcsr, and dbgtx. The optional public instructions runbist, intest, and usercode are not supported. Table 182 describes each of these instructions.

Table 182: JTAG Instruction Description

Instruction /Requisite	Opcode	Description
extest IEEE 1149.1 Required	0x000	Initiates testing of external circuitry, typically board-level interconnects and external circuitry. **extest** connects the BSR between TDI and TDO in the Shift_DR state only. When **extest** is selected, all output signal pin values are driven by values shifted into the BSR and can change only on the falling-edge of TCK in the Update_DR state. Also, when **extest** is selected, all system input pin states must be loaded into the BSR on the rising-edge of TCK in the Capture_DR state. Values shifted into input latches in the BSR are never used by the internal logic of the processor.
clamp	0x001	Allows the state of the signals driven from the pins to be determined from the BSR while the Bypass register is selected as the serial path between TDI and TDO. Signals driven from the component pins do not change while the **clamp** instruction is selected.
highz	0x002	Floats all three-stateable output and input pins. Also, when this instruction is active, the Bypass register is connected between TDI and TDO. This register can be accessed through the JTAG Test-Access port throughout the device operation. Access to the Bypass register can also be obtained with the **bypass** instruction.
sample/ preload IEEE 1149.1 Required	0x003	Performs two functions: • When the TAP controller is in the Capture-DR state, the **sample** instruction occurs on the rising edge of TCK and provides a non-interfering snapshot of the component's normal operation. The instruction causes BSR cells associated with output pins to **sample** the value driven by or to the processor. Similarly, BSR cells associated with input pins **sample** the value driven on the inputs. • When the TAP controller is in the Update-DR state, the **preload** instruction occurs on the falling edge of TCK. It causes the transfer of data held in the boundary-scan cells to the slave register cells. Typically, the slave-latched data is then applied to the system outputs with the **extest** instruction.
idcode IEEE 1149.1 Optional	0x004	Used in conjunction with the Device Identification register. It connects the ID register between TDI and TDO in the Shift_DR state. When selected, **idcode** parallel-loads the hard-wired identification code (32 bits) on TDO into the ID register on the rising edge of TCK in the Capture_DR state. **NOTE:** The ID register is not altered by data being shifted in on TDI.
dbgrx	0x020	Enables software debug of the application subsystem CPU.
load_sram	0x070	Enables software debug of the application subsystem CPU.
dcsr	0x090 or 0x009	Enables software debug of the application subsystem CPU.

Table 182: JTAG Instruction Description (Continued)

Instruction /Requisite	Opcode	Description
dbgtx	0x100	Enables software debug of the application subsystem CPU.
bypass IEEE 1149.1 Required	0x7ff	Selects the Bypass register between TDI and TDO pins while in SHIFT_DR state, effectively bypassing the processor test logic. 0x0 is captured in the CAPTURE_DR state. While this instruction is in effect, all other test data registers have no effect on the operation of the system. Test data registers with both test and system functionality perform their system functions when this instruction is selected.

14.3.3 Test Data Registers

The test data registers include the Bypass register, Boundary Scan register (BSR), and the Device Identification (ID) register.

14.3.3.1 Bypass Register

The single-bit Bypass register is selected as the path between TDI and TDO so that the device can be bypassed during JTAG testing. Therefore, test data can move more rapidly to and from other components on a board that must perform JTAG test operations. When the bypass, highz, or clamp instruction is the current instruction in the IR, serial data is transferred from TDI to TDO in the shift-DR state with a delay of one TCK cycle. A logic 0x0 is loaded from the parallel input of the Bypass register in the capture-DR state. There is no parallel output from the Bypass register.

14.3.3.2 Boundary Scan Register (BSR)

The Boundary Scan register (BSR) consists of a serially connected set of cells around the periphery of the device at the interface between the core logic and the system input/output pads. This register can be used to isolate the pins from the core logic and then drive or monitor the system pins. The connected BSR cells constitute a Shift register.

To prevent a scan operation from disabling power to the device and/or resetting external components, the following pins are not part of the BSR: USBH2_N, USBH2_P, USBH1_N, USBH1_P, PWR_EN, SYS_EN, CLK_TOUT, EXT_WAKEUP<1:0>, nBATT_FAULT, nRESET_IN, PWR_CAP0, PWR_CAP1, TXTAL_OUT, TXTAL_IN, nRESET_OUT, nGPIO_RESET, PXTAL_OUT, PXTAL_IN, PWR_SCL, PWR_SDA, VCTCXO_EN, CLK_POUT, TESTCLK, TEST, and the five TAP controller pins.

Note

The PXA31x processor does not have the following pins: USBH2_N, USBH2_P, USBH1_n and USBH1_P.

For any instructions that use the BSR, the nBATT_FAULT and nRESET_IN pins must always be de-asserted because the BSR logic powers down when either of these pins is asserted. No JTAG instructions using the BSR can be performed with these pins asserted or with the device in deep sleep (D4/S3). In addition, Marvell recommends that the nRESET_OUT pin be allowed to de-assert before with any instructions that use the BSR are executed. This indicates that the part has powered up correctly.

The BSR is selected as the TDR to be connected between TDI and TDO only during the sample/preload and extest instructions. Values in the BSR are used but are not changed when the clamp instruction is executed.

Copyright © 2009 Marvell

In the normal (system) mode of operation, straight-through connections between the core logic and pins are maintained, and normal system operation is unaffected (the same is true when the sample/preload instruction is selected).

When extest is the currently selected instruction, values can be applied to the output pins regardless of the actual values on the input pins and core logic outputs.

The BSR has indeterminate values after power-on and after a JTAG reset (from asserting nTRST or entering the TLR state).

14.3.3.3 Device Identification (ID) Register

The Device Identification (ID) register, shown in Table 183, is used to read the 32-bit device identification code. No programmable supplementary identification code is provided. When the idcode instruction is current, the ID register is selected as the serial path between TDI and TDO. The high-order four bits of the ID register contain the version number of the silicon and changes with each new revision. There is no parallel output from the ID register. The 32-bit device identification code is loaded into the ID register from its parallel inputs during the capture-DR state.

Table 183: JTAG Device Identification Register

JTAG Device Identification (ID) Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Version				Part Number																JEDEC Code											

Bits	Access	Name	Description
31:28	R	Version	Version number of silicon
27:12	R	Part Number	Part number of the device
11:0	R	JEDEC Code	JEDEC Code of the manufacturer 0x013
NOTE: Refer to the System Architecture chapter for JTAG ID values for all of the PXA3xx processors			

14.3.3.4 XScale® Data-Specific Registers

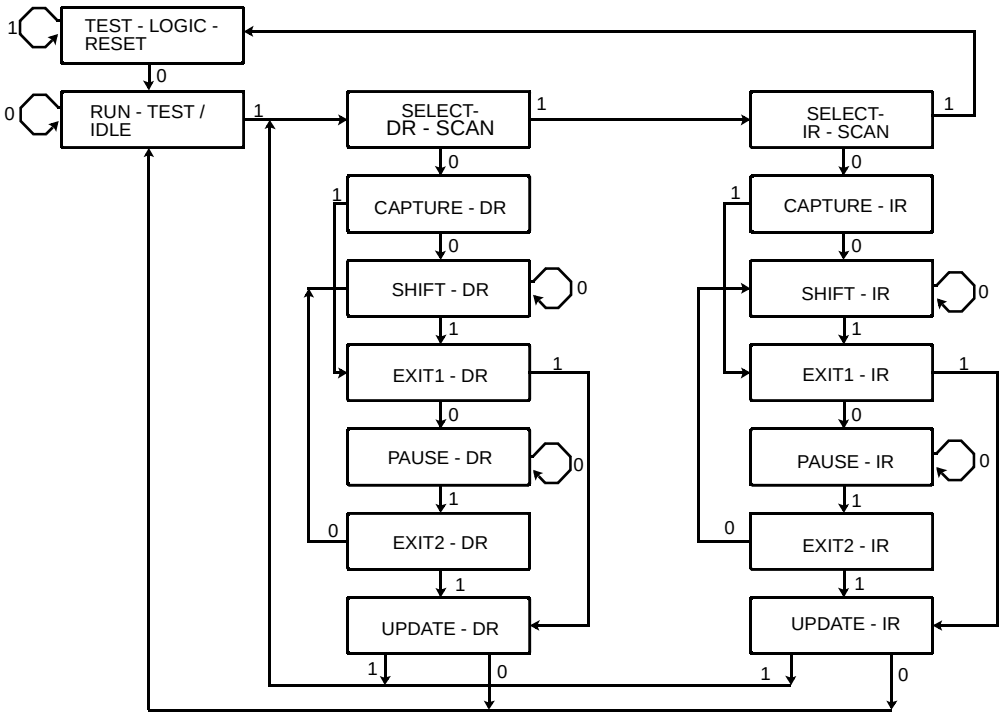
XScale® data registers are not documented here. They are used in conjunction with the user-defined JTAG instructions, which are described the *XScale® Core Developers Manual* available from Intel Corporation.

14.3.4 TAP Controller

The TAP controller is a 16-state synchronous state machine that controls the sequence of JTAG operations. The TAP controller can be controlled through a bus master. It changes state only in response to a rising edge on TCK or power on. The value of the test-mode state (TMS) input signal at a rising edge of TCK controls the sequence of the state machine. The TAP controller is automatically reset on reset exit sequences through nTRST. In addition, the TAP controller can be reset by applying a high signal level on the TMS input for five TCK periods.

Figure 29 shows the state transitions that occur in the TAP controller. Refer to IEEE 1149.1 *Standard Test Access Port and Boundary-Scan Architecture* document for details on the state machine and the public instructions.

Figure 29: TAP Controller State Diagram



NOTE: ALL STATE TRANSITIONS ARE BASED ON THE VALUE OF TMS.

Table 184 describes the TAP controller states in the order that they appear in Figure 29.

Table 184: TAP Controller States

TAP Controller State	Description
Test Logic Reset state	Test logic is disabled to allow normal processor operation. Test logic is disabled when the idcode instruction is enabled. No matter the state of the TAP controller, it enters TLRS when the TMS input is held high (0x1) for at least five rising edges of TCK. The TAP controller remains in this state while TMS is high. The TAP controller is also forced to enter this state when nTRST is asserted. If the TAP controller exits the TLRS as a result of an erroneous low signal on the TMS line at the time of a rising edge on TCK (for example, a glitch due to external interference), it returns to the test-logic reset state after three rising edges of TCK with the TMS line at the intended high logic level. Test logic operation is such that no disturbance is caused to internal system logic operation because of such an error.
Run-Test/Idle state	The TAP controller enters the run-test/idle state from TLRS, update-DR state, or update-IR state. The TAP controller remains in this state as long as TMS is held low. Instructions that do not call functions to execute in the run-test/idle state generate no activity in the test logic while the TAP controller is in this state. The Instruction register and all test data registers retain their current state. The TAP controller moves to the select-DR-scan state when TMS is high on the rising edge of TCK.

Copyright © 2009 Marvell

Table 184: TAP Controller States (Continued)

TAP Controller State	Description
Select-DR-Scan state	A temporary TAP controller state. The test-data registers selected by the current instruction retain their previous state. If TMS is held low on the rising edge of TCK when the TAP controller is in this state, the TAP controller moves into the capture-DR state and a scan sequence for the selected test-data register is initiated. If TMS is held high on the rising edge of TCK, the TAP controller moves into the select-IR-scan state. The instruction does not change while the TAP controller is in this state.
Capture-DR state	When the TAP controller is in the capture-DR state and the current instruction is **sample**/**preload**, the BSR captures input pin data on the rising edge of TCK. Test data registers that do not have parallel inputs are not changed. Also, the BSR retains it previous state if the **sample**/**preload** instruction is not selected in this state. The instruction does not change while the TAP controller is in this state. If TMS is high on the rising edge of TCK, the TAP controller enters the exit1-DR. The TAP controller enters the shift-DR state if TMS is low on the rising edge of TCK.
Shift-DR state	The test-data register, which is connected between TDI and TDO as a result of the current instruction, shifts data one bit position nearer to its serial output on each rising edge of TCK. Test-data registers that the current instruction selects but does not place in the serial path retain their previous value. The instruction does not change while the TAP controller is in this state. If TMS is high on the rising edge of TCK, the TAP controller enters the exit1-DR state. If TMS is low on the rising edge of TCK, the TAP controller remains in the shift-DR state.
Exit-DR state	A temporary controller state. When TMS is held high on the rising edge of TCK, the TAP controller enters the update-DR state, which terminates the scanning process. If TMS is held low on the rising edge of TCK, the TAP controller enters the pause-DR state. The instruction does not change while the TAP controller is in this state. All test-data registers selected by the current instruction retain their previous value during this state.
Pause-DR state	Allows the TAP controller to halt the shifting of data through the test-data register in the serial path between TDI and TDO. The test-data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state. The TAP controller remains in this state as long as TMS is low. The TAP controller moves to the exit2-DR state when TMS goes high on the rising edge of TCK.
Exit2-DR state	A temporary state. If TMS is held high on the rising edge of TCK, the TAP controller enters the update-DR state, which terminates the scanning process. If TMS is held low on the rising edge of TCK, the TAP controller enters the shift-DR state. The instruction does not change while the TAP controller is in this state. All test data registers selected by the current instruction retain their previous value during this state.
Update-DR state	The BSR has a parallel register that prevents changes at the parallel output while data is shifted in response to the **extest** sample/preload instructions. When the BSR is selected and the TAP controller is in the update-DR state, data is latched onto the BSR parallel output from the Shift register path on the falling edge of TCK. The data held at the latched parallel output does not change unless the TAP controller is in this state. All of the Test Data register shift register-bit positions selected by the current instruction retain their previous values. The instruction does not change while the TAP controller is in this state. If the TAP controller is in this state and TMS is held high on the rising edge of TCK, the TAP controller enters the select-DR-scan state. If TMS is held low on the rising edge of TCK, the TAP controller enters the run-test/idle state.

Copyright © 2009 Marvell

Table 184: TAP Controller States (Continued)

TAP Controller State	Description
Select-IR Scan state	A temporary controller state. The test data registers selected by the current instruction retain their previous state. If TMS is held low on the rising edge of TCK, the TAP controller moves into the capture-IR state and a scan sequence for the Instruction register is initiated. If TMS is held high on the rising edge of TCK, the TAP controller moves to the test-logic-reset state. The instruction does not change in this state.
Capture-IR state	The Shift register contained in the Instruction register loads the fixed value 0x00000000001 on the rising edge of TCK. The test-data register selected by the current instruction retains its previous value. The instruction does not change in this state. If TMS is held high on the rising edge of TCK, the TAP controller enters the exit1-IR state. If TMS is held low on the rising edge of TCK, the TAP controller enters the shift-IR state.
Shift-IR state	The Shift register contained in the Instruction register is connected between TDI and TDO and shifts data one bit position nearer to its serial output on each rising edge of TCK. The test-data register selected by the current instruction retains its previous value in this state. The instruction does not change. If TMS is held high on the rising edge of TCK, the TAP controller enters the exit1-IR state. If TMS is held low on the rising edge of TCK, the TAP controller remains in the shift-IR state.
Exit1-IR state	A temporary state. If TMS is held high on the rising edge of TCK, the TAP controller enters the update-IR state, which terminates the scanning process. If TMS is held low on the rising edge of TCK, the TAP controller enters the pause-IR state. The test-data register selected by the current instruction retains its previous value during the Exit1-IR state. The instruction does not change and the Instruction register retains its state.
Pause-IR state	Allows the TAP controller temporarily to halt the shifting of data through the Instruction register. The test-data registers selected by the current instruction retain their previous values. The instruction does not change and the Instruction register retains its state. The TAP controller remains in this state as long as TMS is held low. The TAP controller enters the exit2-IR state when TMS goes high on the rising edge of TCK.
Exit2-IR state	A temporary state. If TMS is held high on the rising edge of TCK, the TAP controller enters the update-IR state, which terminates the scanning process. If TMS is held low on the rising edge of TCK, the TAP controller enters the shift-IR state. This test-data register selected by the current instruction retains its previous value during this state. The instruction does not change and the Instruction register retains its state.
Update-IR state	The instruction shifted into the serial Instruction register is latched onto the parallel register from the Shift register path on the falling edge of TCK. When it is latched, the new instruction becomes the current instruction. Test-data registers selected by the current instruction retain their previous values. If TMS is held high on the rising edge of TCK, the TAP controller enters the select-DR-scan state. If TMS is held low on the rising edge of TCK, the TAP controller enters the run-test/idle state.

14.4 Register Descriptions

The JTAG logic has no memory-mapped associated registers.

Copyright © 2009 Marvell

15 Power Manager I2C Interface

This chapter describes the power manager integrated circuit (I^2C*) bus interface unit for the processor. This manual refers to this peripheral and bus as the PWR_I2C unit and the PWR_I2C bus. The PWR_I2C unit uses the PWR_I2C bus to communicate with the external voltage regulator.

The power-manager PWR_I2C bus is a full-function PWR_I2C bus capable of all normal operations, including master/slave and receive/transmit operation. The PWR_I2C bus allows the PWR_I2C unit to interface with a microcontroller for power-management functions.

**Future Marvell documentation will refer to this serial bus as a two-wire serial interface (TWSI).*

15.1 I/O Signals

The PWR_I2C bus is a serial bus (I^2C interface developed by Phillips Corporation) consisting of a two-pin interface. PWR_SDA is the data pin for input and output functions and PWR_SCL is the clock pin for reference and control of the PWR_I2C bus. Table 185 summarizes the PWR_I2C signals.

Table 185: I^2C Interface Signals Summary

Name	Type	Description
PWR_SDA	bidirectional	PWR_I2C Data signal
PWR_SCL	bidirectional	PWR_I2C Clock

15.2 Feature List

- I^2C bus interface unit of the power manager features are:
- I^2C compliant
- Multi-master and arbitration support
- Supports fast-mode operation of upto 380 Kbits/sec
- S3 entry due to battery fault operation with 32.768 kHz operation of 100 bits/sec

15.3 Functional Description

Each device on the PWR_I2C bus is recognized by a unique 7-bit address and can operate as a transmitter/receiver and master/slave. Table 186 lists the PWR_I2C device types.

Table 186: PWR_I2C Bus Definitions (Sheet 1 of 2)

I^2C Device	Definition
Transmitter	Sends data to the PWR_I2C bus
Receiver	Receives data from the PWR_I2C bus
Master	Initiates a transfer, generates the clock signal, and terminates the transactions

Copyright © 2009 Marvell

Table 186: PWR_I2C Bus Definitions (Sheet 2 of 2)

I^2C Device	Definition
Slave	The device addressed by a master

The PWR_I2C unit as a master transmitter addresses the external Power Manger Integrated Circuit (PMIC) as a slave receiver to receive data. When the PWR_I2C unit reads data, it is a master receiver, and the PMIC is a slave transmitter. In both cases, the master generates the clock, initiates the transaction, and terminates it. The PWR_I2C initiates transactions as a master or receives them as a slave. Under both conditions, the PWR_I2C unit performs reads, writes, or both to the PWR_I2C bus.

15.3.1 Operational Blocks

The PWR_I2C unit is a slave peripheral device that is connected to the power management unit within the services module. The application subsystem core interrupt mechanism can be used to notify the CPU of activity on the PWR_I2C bus. Polling can be used instead of interrupts, although it would be very cumbersome.

The PWR_I2C unit consists of the two-wire interface to the PWR_I2C bus, an 8-bit buffer for passing data to and from the processor, a set of control and status registers in the memory-mapped address space, and a Shift register for parallel/serial conversions.

The PWR_I2C unit can initiate an interrupt when it detects that the buffer is full or empty or when it detects a slave address or a bus-error condition. Software must explicitly clear all interrupt conditions.

The 8-bit PWR_I2C Data Buffer register (PIDBR) receives a byte of data from the Shift register interface of the PWR_I2C bus on one side and it receives parallel data from the processor internal bus on the other side. The Serial Shift register is not user accessible.

The PWR_I2C unit operates in fast-mode up to up to 380 Kbps. Fast-mode logic levels, formats, capacitive loading, and protocols are exactly the same as the 100 Kbps standard mode. In addition, the PWR_I2C unit supports 32.768 kHz operation of 100 bps. Because the data setup-and-hold times differ between the fast and standard mode, the PWR_I2C unit is designed to meet the slower standard mode requirements for these two specifications.

15.3.2 PWR_I2C Bus Interface Modes

The PWR_I2C bus interface unit can operate in different modes to accomplish a transfer. Table 187 summarizes the modes.

Copyright © 2009 Marvell

Table 187: PWR_I2C Bus Interface Operating Modes

Mode	Definition
Master - Transmit	PWR_I2C Unit acts as a master Used for a write operation PWR_I2C Unit sends the data PWR_I2C Unit is responsible for clocking Slave device will be in slave-receive mode
Master - Receive	PWR_I2C Unit acts as a master Used for a read operation PWR_I2C Unit receives the data PWR_I2C Unit is responsible for clocking Slave device will be in slave-transmit mode
Slave - Transmit	PWR_I2C Unit acts as a slave Used for a read (master) operation PWR_I2C Unit sends the data Master device will be in master-receive mode
Slave - Receive (default)	PWR_I2C Unit acts as a slave Used for a write (master) operation PWR_I2C Unit receives the data Master device will be in master-transmit mode

While the PWR_I2C unit is in Idle mode and neither receiving nor transmitting serial data, the unit defaults to Slave-Receive mode, which allows the interface to monitor the bus and receive any slave addresses directed to the PWR_I2C unit.

When the PWR_I2C unit receives an address that matches the 7-bit address in the PWR_I2C Slave Address register (PISAR) or the General Call Address (0x00), the interface either remains in Slave-Receive mode or transitions to Slave-Transmit mode as determined by the read/write (R/nW) bit (the least significant bit of the byte containing the slave address).

- If the R/nW bit is cleared, the master is initiating the transaction to perform a Write and the PWR_I2C unit remains in Slave-Receive mode.
- If the R/nW is set, the initiating master wants to read data and the slave transitions to Slave-Transmit mode. Slave operation is further defined in Section 15.3.5.

When the processor must initiate a Read or Write on the PWR_I2C bus, the PWR_I2C unit transitions from the default Slave-Receive mode to Master-Transmit mode.

- If the PWR_I2C unit is to write data, the interface remains in Master-Transmit mode after the address transfer completes.
- If the PWR_I2C unit is to read data, the PWR_I2C Unit transmits the start address and transitions to Master-Receive mode.

15.3.3 Start and Stop Bus States

The PWR_I2C bus defines a transaction START and a transaction STOP bus state, where each is used at the beginning and end of the transfer of one to an unlimited number of bytes on the bus. The PWR_I2C unit uses the start and stop bits in the PWR_I2C Control register (PICR) to perform the following tasks:

- Initiate an additional byte transfer.
- Initiate a START condition on the PWR_I2C bus.
- Enable data chaining (repeated START).
- Initiate a STOP condition on the PWR_I2C bus.

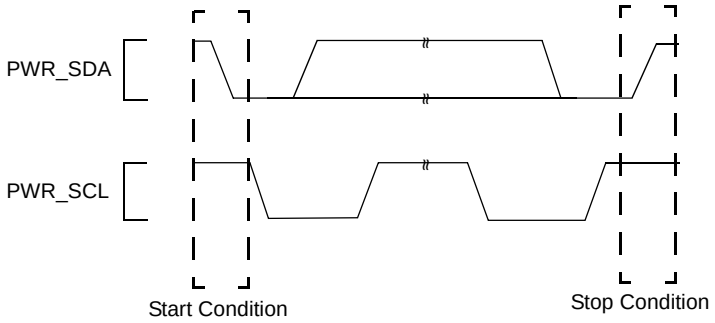
Table 188 summarizes the definition of the *start* and *stop* bits in the PICR.

Table 188: START and STOP Bit Definitions

STOP Bit	START Bit	Condition	Notes
0	0	No START or STOP	No START or STOP condition is sent by the PWR_I2C unit. This is used when multiple data bytes need to be transferred.
0	1	START Condition and Repeated START	The PWR_I2C unit sends a START condition and transmits the contents of the 8-bit PIDBR after the START. The PIDBR must contain the 7-bit address and the *R/nW* bit before a START is initiated. For a repeated start, the PIDBR contains the target slave address and the *R/nW* bit. This enables multiple transfers to different slaves without surrendering the bus. The interface remains in Master-Transmit mode for a write or it transitions to master-receive mode if a read is requested.
1	X	STOP Condition	In Master-Transmit mode, the PWR_I2C unit transmits the 8-bit PIDBR and then sends a STOP condition on the PWR_I2C bus. In Master-Receive mode, the PICR[ACKNAK] must be set, which defines a negative acknowledge (NAK) pulse (see Section 15.3.4.2). The PWR_I2C bus transmits the *NAK* pulse, places the received data byte into the PIDBR, and sends a STOP on the PWR_I2C bus.

Figure 30 shows the relationship between the PWR_SDA and PWR_SCL lines for a START and STOP condition.

Figure 30: Start and Stop Conditions



15.3.3.1 START Condition

The START condition (PICR[START] = 1, PICR[STOP] = 0) initiates a master transaction or repeated START.

1. Software must load the target slave address and the R/nW bit in the Power Manager I2C Data Buffer Register (PIDBR) before setting the PICR[CTL0] (START) bit.
2. The START and the PIDBR contents are transmitted on the PWR_I2C bus when the PICR Transfer Byte bit is set.

Copyright © 2009 Marvell

The PWR_I2C bus remains in Master-Transmit mode when a Write is requested or enters Master-Receive mode when a Read is requested.

3. For a repeated start (a change in Read or Write or a change in the target slave address), the PIDBR contains the updated target slave address and the R/nW bit. A repeated start enables multiple transfers to different slaves without surrendering the bus.

15.3.3.2 No START or STOP Condition

No START or STOP condition (PICR[START] = 0, PICR[STOP] = 0) is used in Master-Transmit mode while the PWR_I2C unit is transmitting multiple data bytes:

1. Software writes the data byte, sets the PICR[ITEIE] bit, and clears the PISR[ITE] bit.
2. The software then writes a new byte to the PIDBR and sets the PICR[TB] bit, which initiates the new byte transmission.

   This process continues until software sets the start or stop bit. The PWR_I2C unit does not automatically clear the start and stop bits in the PICR after the transmission of a START, STOP, or repeated START.

After each byte transfer, including the acknowledge pulse defined by the ICR[ACKNAK] control bit, the PWR_I2C unit holds the PWR_SCL line low to insert wait states until the PICR[TB] is set. This action notifies the PWR_I2C unit to release the PWR_SCL line and allow the next data transfer to proceed.

15.3.3.3 STOP Condition

The STOP condition (PICR[START] = X, PICR[STOP] = 1) terminates a data transfer:

- In Master-Transmit mode, the PICR[STOP] and the PICR[TB] must be set to initiate the last byte transfer.
- In Master-Receive mode, the PWR_I2C interface must set PICR[ACKNAK], PICR[STOP], and PICR[TB] to initiate the last transfer.
- Software must clear PICR[STOP] after the Stop condition is transmitted.

15.3.4 Data Transfer Sequence

The PWR_I2C bus interface unit transfers in 1-byte increments:

1. START
2. 7-bit slave address
3. R/nW Bit
4. Acknowledge pulse
5. 8 bits of data
6. ACK/NAK pulse
7. Repeat of steps 5 and 6 for the required number of bytes
8. Repeated START (repeat step 1) or STOP

15.3.4.1 Data and Addressing Management

Data and slave addressing is managed through the PI2C Data Buffer register (PIDBR) and the I2C Slave Address register (PISAR). The PIDBR (see Section 15.4.4, “Power Manager I2C Data Buffer Register (PIDBR)”) contains data or a slave address and R/nW bit. The PISAR contains the PWR_I2C unit programmable slave address. Data entering the PWR_I2C unit is received into the PIDBR after a full byte is received and acknowledged. To transmit data, the CPU writes to the PIDBR, and the PWR_I2C unit passes this data onto the serial bus when the PICR[TB] bit is set. See Section 15.4.1, “Power Manager I2C Control Register (PICR)”.

The following process occurs when the PWR_I2C unit is in Master or Slave Transmit mode:

1. Software writes data to the PIDBR, which initiates a master transaction or sends the next data byte after the PISR[ITE] bit is set.
2. The PWR_I2C unit transmits the data from the PIDBR when the PICR[TB] is set.
3. When enabled with the PICR[ITEIE] set, a PIDBR transmit-empty interrupt is signalled when a byte is transferred on the PWR_I2C bus and the acknowledge cycle is complete.
4. When the PWR_I2C bus is ready to transfer the next byte before the CPU writes to PIDBR and a STOP condition is not in place, the PWR_I2C unit inserts wait states until software writes a new value into the PIDBR and sets the PICR[TB].

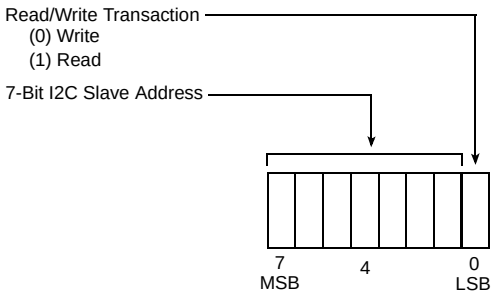
The following process occurs when the PWR_I2C unit is in Master or Slave Receive mode:

1. Software reads the PIDBR data after the PIDBR Receive-Full interrupt is signalled (if enabled with the PICR[DRFIE] bit set).
2. The PWR_I2C unit transfers data from the Shift register to the PIDBR after the ACK cycle completes.
3. The PWR_I2C unit inserts wait states until the PIDBR is read. Refer to Section 15.3.4.2 for acknowledge-pulse information in Receiver mode.
4. After the CPU reads the PIDBR, the PWR_I2C unit writes the PICR[ACKNAK] and the PICR[TB] bits, allowing the next byte transfer to proceed.

15.3.4.1.1 Addressing a Slave Device

As a master device, the PWR_I2C unit must compose and send the first byte of a transaction. This byte consists of the slave address for the intended device and a R/nW bit for transaction definition. The slave address and the R/nW bit are written to the PIDBR (see Figure 31).

Figure 31: Data Format of First Byte in Master Transaction



The first byte transmission must be followed by an ACK pulse from the addressed slave. When the transaction is a write:

1. The PWR_I2C unit remains in Master-Transmit mode.
2. The addressed slave device stays in Slave-Receive mode.

When the transaction is a read:

1. The PWR_I2C bus transitions to Master-Receive mode immediately after the ACK.
2. The addressed slave device transitions to Slave-Transmit mode.

When a NAK is returned, the PWR_I2C unit aborts the transaction by automatically sending a STOP and setting the PISR[BED] bit.

When the PWR_I2C unit is enabled and idle:

Copyright © 2009 Marvell

1. It remains in Slave-Receive mode and monitors the PWR_I2C bus for a START signal.
2. Upon detecting a START pulse, the PWR_I2C unit reads the first seven bits and compares them to those in the PWR_I2C Slave Address register (PISAR) and the general call address (0X00).
3. When the bits match those of the PISAR register, the PWR_I2C unit reads the eighth bit (R/nW bit) and transmits an ACK pulse.
4. The PWR_I2C unit either remains in Slave-Receive mode (R/nW = 0) or transitions to Slave-Transmit mode (R/nW = 1).

See Section 15.3.6.4 for actions when a general call address is detected.

15.3.4.2 PWR_I2C Acknowledge

Every PWR_I2C byte transfer must be accompanied by an acknowledge pulse, which the receiver (master or slave) always generates. The transmitter must release the PWR_SDA line for the receiver to transmit the acknowledge pulse.

When the target slave receiver device (in Master-Transmit mode) cannot generate the acknowledge pulse:

1. The PWR_SDA line remains high, which indicates a negative-acknowledge (NAK).
2. This lack of acknowledge causes the PWR_I2C unit to set the PISR[BED] and generate the associated interrupt when enabled. Software must manage the interrupt.
3. The master can then either generate a STOP condition to abort the transfer or generate a repeated START condition to start a new transfer.

In Master-Receive mode:

1. The PWR_I2C interface sends a negative-acknowledge (NAK) pulse to signal the slave transmitter to stop sending data.
2. The ICR[ACKNAK] bit controls the ACK/NAK pulse value driven onto the PWR_I2C bus. As required by the I^2C protocol, PISR[BED] is not set for Master-Receive mode NAK.
3. The PWR_I2C interface automatically transmits the ACK pulse after it receives each byte from the serial bus.
4. Software must set the PICR[ACKNAK] to generate a NAK before the unit receives the last byte.
5. The NAK pulse, sent after the last byte, signals that the last byte has been sent.

In Slave mode:

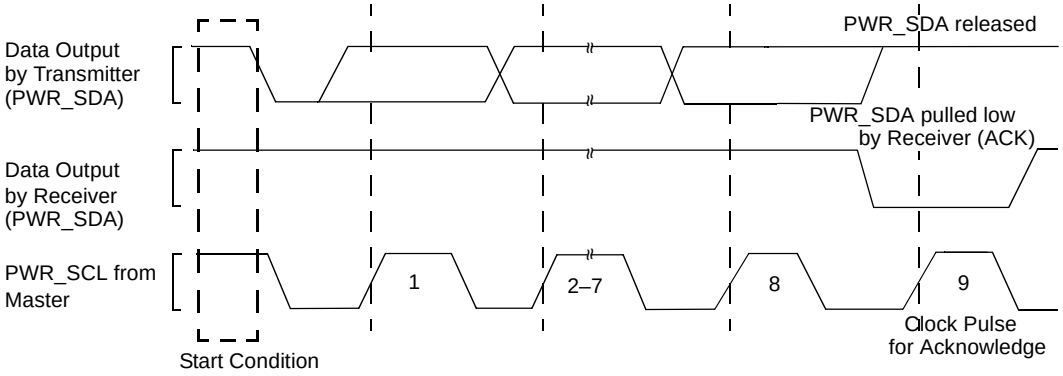
1. The PWR_I2C interface automatically acknowledges its own slave address, regardless of the ICR[ACKNAK] setting.
2. In Slave-Receive mode, an ACK response automatically follows a data byte, regardless of the PICR[ACKNAK] setting. The PWR_I2C unit sends the ACK value after receiving the eighth data bit of the byte.

In Slave-Transmit mode:

1. The last byte is transferred when a NAK is received from the master.
2. The master sends either a STOP or repeated START.
3. The PISR[UB] remains set until a STOP or repeated START is received.

Copyright © 2009 Marvell

Figure 32: Acknowledge on the PWR_I2C Bus



15.3.4.3 Master Operations

When software initiates a Read or Write on the PWR_I2C bus, it transitions from the default Slave-Receive mode to Master-Transmit mode. The start pulse is sent and then the 7-bit slave address and the R/nW bit are sent. After the master receives an acknowledge, the PI2C interface enters one of two master modes:

- Master-Transmit — PWR_I2C interface writes data
- Master-Receive — PWR_I2C interface reads data

The CPU initiates a master transaction by writing to the PICR register. Data is read and written from the PWR_I2C unit through the memory-mapped registers. Table 189 describes the actions of the PWR_I2C unit as a master device.

Table 189: Master Transactions

PWR_I2C Master Action	Operating Mode	Description
Generate clock output	Master-transmit Master-receive	• The master always drives the SCL line. • The PICR[SCLE] and PICR[IUE] must be set.
Write target slave address to PIDBR	Master-transmit Master-receive	• The CPU writes to PIDBR bits [7:1] before a START condition is enabled. • The first 7 bits are sent on the PWR_I2C bus after START. See Section 15.3.3.
Write R/nW Bit to PIDBR	Master-transmit Master-receive	• The CPU writes to the least significant PIDBR bit with the target slave address • If the R/nW bit is cleared, the master remains a master transmitter; if it is set, the master transitions to a master receiver. See Section 15.3.4.1

Copyright © 2009 Marvell

Table 189: Master Transactions (Continued)

PWR_I2C Master Action	Operating Mode	Description
Signal START Condition	Master-transmit Master-receive	See “Generate clock output.” After the target slave address and the R/nW bit are in the PIDBR: • Software sets PICR[START]. • Software sets PICR[TB] to initiate the START condition. See Section 15.3.3
Initiate first data byte transfer	Master-transmit Master-receive	CPU writes a byte to the PIDBR: • PWR_I2C unit transmits the byte when PICR[TB] is set. • PWR_I2C bus interface unit clears the PICR[TB] and sets the PISR[ITE] bit when the transfer completes.
Write one data byte to the PIDBR	Master-transmit only	• Occurs when the PISR[ITE] is set and the PICR[TB] is cleared if the PIDBR Transmit-Empty Interrupt is signalled to the processor. • CPU writes one data byte to the PIDBR, sets the appropriate START/STOP bit combination, and then sets PICR[TB] to send the data; eight bits are taken from the shift registers and written on the serial bus; the eight bits are followed by a STOP if the PICR[STOP] bit is set.
Wait for Acknowledge from slave-receiver	Master-transmit only	• As a master-transmitter, the PWR_I2C unit generates the clock for the acknowledge pulse; the PWR_I2C Unit is responsible for releasing the SDA line to allow slave-receiver ACK transmission. • See Section 15.3.4.2.
Read one byte of PWR_I2C Data from the PIDBR	Master-receive only	• Eight bits are read from the serial bus, collected in the shift register, then transferred to the PIDBR after the PICR[ACKNAK] bit is read. • The CPU reads the PIDBR when PISR[IRF] is set and PICR[TB] is cleared. If enabled, a PIDBR Receive Full Interrupt is signalled to the CPU. • When the PIDBR is read, if PISR[ACKNAK] is cleared (indicating ACK), software must clear the PICR[ACKNAK] bit and set PICR[TB] to initiate the next byte read. • If PISR[ACKNAK] is set (indicating NAK), PICR[TB] is cleared, PICR[STOP] is set, and PISR[UB] is set, then the last data byte has been read into the PIDBR and the PWR_I2C bus interface unit is sending the STOP. • If PISR[ACKNAK] is set (indicating NAK), ICR[TB] is clear, but PICR[STOP] is clear, the software has two options: 1. Set PICR[START], write a new target address to the PIDBR, and set PICR[TB], which sends a repeated START. 2. Set PICR[MA] and leave PICR[TB] clear, which sends a STOP only. See Figure 33.
Transmit Acknowledge to slave-transmitter	Master-receive only	• As a master receiver, the PWR_I2C unit generates the clock for the acknowledge pulse. It also drives the PWR_SDA line during the ACK cycle. • If the next data byte is to be the last transaction, software sets PICR[ACKNAK] for NAK generation. See Section 15.3.4.2.

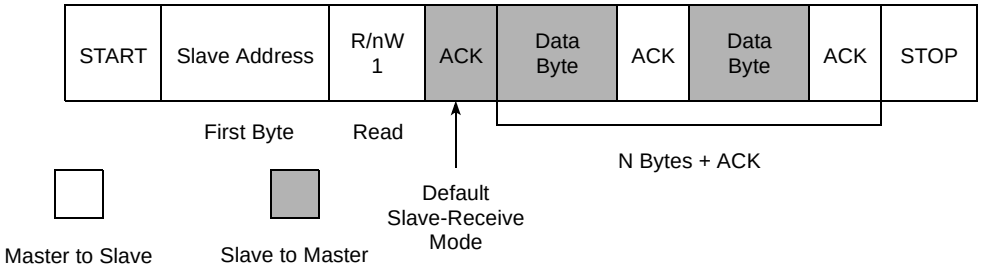
Table 189: Master Transactions (Continued)

PWR_I2C Master Action	Operating Mode	Description
Generate a Repeated START to chain PWR_I2C transactions	Master-transmit Master-receive	• If data chaining is preferred, use a repeated START condition instead of a STOP condition. • The repeated START is generated after the last data byte of a transaction is transmitted on the PWR_I2C bus. • Software must write the next target slave address and the R/nW bit to the PIDBR, set PICR[START], and set PICR[TB]. See Section 15.3.3 and Figure 34.
Generate a STOP	Master-transmit Master-receive	• A STOP is generated after the last data byte of a transaction is transmitted on the PWR_I2C bus. • PICR[STOP] must be set to generate the STOP condition. See Section 15.3.3 and Figure 35.

When the software must read data:

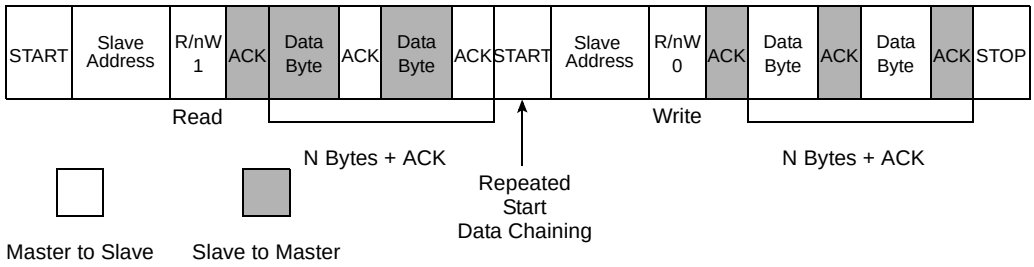
1. The PWR_I2C unit transitions from Slave-Receive mode to Master-Transmit mode to transmit the start address.
2. The PWR_I2C unit transitions to Master-Receive mode immediately after the ACK pulse to wait for the Read data from the slave device. Multiple transactions can occur during an PWR_I2C operation, such as transitioning from Master-Receive to Master-Transmit through a repeated start.

Figure 33: Master-Receiver Read from Slave-Transmitter



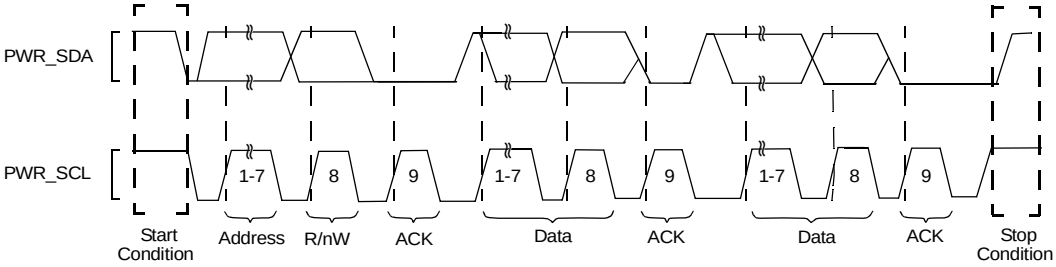
Copyright © 2009 Marvell

Figure 34: Master-Receiver Read from Slave-Transmitter/Repeated Start*



*Full title: Master-Receiver Read from Slave Transmitter/Repeated Start/Master-Transmitter Write to Slave-Receiver

Figure 35: A Complete Data Transfer



15.3.4.4 Master Programming Examples

The examples in this section illustrate how to initialize the PWR_I2C unit, write and read one byte as a master, write two bytes and repeated start read one byte, and read two bytes and send a STOP using the abort.

15.3.4.5 Initialize the Unit

1. Write to the PISAR slave address field.
2. Write PICR: enable all interrupts (except Arb Loss) and set the IUE and SCLE bits to enable the PWR_I2C interface and SCL.

15.3.4.6 Write 1 Byte as a Master

1. Load the target slave address and R/nW bit (0 for write) into the PIDBR.
2. Initiate the Write in the PICR: set START, clear STOP, clear ALDIE, and set TB.
3. When a PIDBR Transmit Empty interrupt occurs, read the PISR register: ITE = 1, UB = 1, RWM = 0.
4. Write a 1 to the PISR[ITE] bit to clear the interrupt.
5. Write a 1 to the PISR[ALD] bit if set.
6. Load the data byte to be transferred into the PIDBR

7. Initiate the Write in the PICR: clear START, set STOP, set ALDIE, and set TB.
8. When a PIDBR Transmit-Empty interrupt occurs (PWR_I2C is sending STOP), read the PISR: ITE = 1, UB = x, and RWM = 0.
9. Write a 1 to the PISR[ITE] bit to clear the interrupt.
10. Clear the PICR[STOP] bit.

15.3.4.7 Read 1 Byte as a Master

1. Load the target slave address and the R/nW bit into the PIDBR. R/nW must be 1 for a Read.
2. Initiate the Write in the PICR: set START, clear STOP, ALDIE, and set TB.
3. When a PIDBR Transmit-Empty interrupt occurs, read PISR: ITE = 1, UB = 1, and RWM = 1.
4. Write a 1 to the PISR[ITE] bit to clear the interrupt.
5. Initiate the Read in the PICR: clear START, set STOP, set ALDIE, set ACKNAK, and set TB.
6. When a PIDBR Receive-Full interrupt occurs (unit is sending Stop), read the PISR: IDBR receive full (1), PISR[UB] = x, PISR[RWM] = 1, ACK/NAK bit (1)
7. Write a 1 to the PISR[IRF] bit to clear the interrupt.
8. Read PIDBR data.
9. Clear PICR[STOP] and PICR[ACKNAK] bits.

15.3.4.8 Write 2 Bytes and Repeated Start Read 1 Byte as a Master

1. Load target slave address and R/nW bit in the PIDBR. R/nW must be 0 for a Write.
2. Initiate the Write.
   Set PICR[START], clear PICR[STOP], clear PICR[ALDIE], set PICR[TB]
3. When an PIDBR Transmit-Empty interrupt occurs:
   Read PISR register: PISR[ITE] = 1, PISR[UB] = 1, PISR[RWM] = 0
4. Write a 1 to the PISR[ITE] bit to clear interrupt.
5. Load data byte to be transferred in the PIDBR.
6. Initiate the Write.
   Clear PICR[START], clear PICR[STOP], set PICR[ALDIE], set PICR[TB]
7. When an PIDBR Transmit-Empty interrupt occurs:
   Read PISR register: PISR[ITE] = 1, PISR[UB] = 1, PISR[RWM] = 0
8. Write a 1 to the PISR[ITE] bit to clear interrupt.
9. Repeat steps 5-8 one time.
10. Load target slave address and R/nW bit in the PIDBR. R/nW must be 1 for a Read.
11. Send repeated Start as a master.
    Set PICR[START], clear PICR[STOP], clear PICR[ALDIE], set PICR[TB]
12. When an PIDBR Transmit-Empty interrupt occurs:
    Read PISR register: PISR[ITE] = 1, PISR[UB] = 1, PISR[RWM] = 1
13. Write a 1 to the PISR[ITE] bit to clear interrupt.
14. Initiate the Read.
    Clear PICR[START], set PICR[STOP], set PICR[ALDIE], set PICR[ACKNAK], set PICR[TB]
15. When an PIDBR Receive-Full interrupt occurs (unit is sending Stop):
    Read PISR register: PISR[IRF] = 1, PISR[UB] = x, PISR[RWM] = 1, PISR[ACKNAK] = 1.
16. Write a 1 to the PISR[IRF] bit to clear the interrupt.
17. Read PIDBR data.
18. Clear PICR[STOP] and PICR[ACKNAK] bits.

15.3.4.9 Read 2 Bytes as a Master - Send STOP Using the Abort

1. Load target slave address and R/nW bit in the PIDBR. R/nW must be 1 for a Read.

Copyright © 2009 Marvell

2. Initiate the Write.
   Set PICR[START], clear PICR[STOP], clear PICR[ALDIE], set PICR[TB]
3. When an PIDBR Transmit-Empty interrupt occurs:
4. Read PISR register: PISR[ITE] = 1, PISR[UB] = 1, PISR[RWM] = 1
5. Write a 1 to the PISR[ITE] bit to clear interrupt.
6. Initiate the Read.
   Clear PICR[START], clear PICR[STOP], set PICR[ALDIE], clear PICR[ACKNAK], set PICR[TB]
7. When an PIDBR Receive-Full interrupt occurs:
   Read PISR register: PISR[IRF] = 1, PISR[UB] = 1, PISR[RWM] = 1, ACK/NAK bit (0)
8. Write a 1 to the PISR[IRF] bit to clear the interrupt.
9. Read PIDBR data.
10. Clear PICR[STOP] and PICR[ACKNAK] bits.
11. Initiate the Read.
    Clear PICR[START], clear PICR[STOP], set PICR[ALDIE], set PICR[ACKNAK], set PICR[TB]
    PICR[STOP] is not set because Stop or repeated Start is determined on the byte read.
12. When an PIDBR Receive-Full interrupt occurs:
    Read PISR register: PISR[IRF] = 1, PISR[UB] = 1, PISR[RWM] = 1, PISR[ACKNAK] = 1
13. Write a 1 to the PISR[IRF] bit to clear the interrupt.
14. Read PIDBR data.
15. Initiate Stop abort condition (Stop with no data transfer).
    Set PICR[MA].

Note

If a NAK is not sent in Step 11, the next transaction must involve another data byte read.

15.3.5 Slave Operations

Table 190 describes the PWR_I2C Unit responsibilities as a slave device.

Table 190: Slave Transactions

PWR_I2C **Slave Action**	**Mode of Operation**	**Definition**
Slave-receive (default mode)	Slave-receive only	• PWR_I2C Unit monitors all slave address transactions • PICR[IUE] bit must be set • PWR_I2C Unit monitors bus for START conditions; when a START is detected, the interface reads the first 8 bits and compares the most significant 7 bits with the 7 bit I^2C Slave Address register and the General Call address (0X00); if there is a match, the PWR_I2C Unit sends an ACK • If the first 8 bits are all zeroes, this indicates a general call address; if PICR[GCD] is clear, both the PISR[GCAD] and PISR[SAD] are set; see Section 15.3.6.4. • If the 8th bit of the first byte (R/nW bit) is low, the PWR_I2C Unit remains in Slave-Receive mode and the PISR[SAD] is cleared. If the R/nW bit is high, the PWR_I2C Unit transitions to Slave-Transmit mode and ISR[SAD] is set
Setting the Slave Address Detected bit	Slave-receive Slave-transmit	• Indicates the interface has detected a PWR_I2C operation that addresses the processor (this includes general call address); the processor can distinguish a PISAR match from a General Call by reading tPISR[GCAD] • An interrupt is signalled (if enabled) after the matching slave address is received and acknowledged
Read one byte of PWR_I2C Data from the PIDBR	Slave-receive only	• Data Receive mode of PWR_I2C slave operation • This operation occurs when PISR[IRF] is set and PICR[TB] is clear; if enabled, the PIDBR receive-full interrupt is generated • Eight bits are read from the serial bus into the Shift register; when a full byte has been received and the ACK/NAK bit has completed, the byte is transferred from the shift register to the PIDBR • Occurs when the PIDBR receive full bit in the PISR is set and the transfer byte bit is clear; if enabled, the PIDBR Receive Full Interrupt is signalled to the CPU • CPU reads 1 data byte from the PIDBR. When the PIDBR is read, the processor writes the preferred PICR[ACKNAK] bit and sets the PICR[TB], which causes the PWR_I2C Unit to stop inserting Wait states and let the master transmitter write the next chunk of information
Transmit Acknowledge to master-transmitter	Slave-receive only	• As a slave-receiver, the PWR_I2C Unit is responsible for pulling the PWR_SDA line low to generate the ACK pulse during the high PWR_SCL period • The PICR[ACKNAK] controls the ACK data the PWR_I2C Unit drives; see Section 15.3.4.2
Write one byte of PWR_I2C data to the PIDBR	Slave-transmit only	• Data Transmit mode of PWR_I2C slave operation • This operation occurs when ISR[ITE] is set and ICR[TB] is clear; if enabled, the IDBR transmit-empty interrupt is generated • writes a data byte to the PIDBR and sets the PICR[TB] to initiate the transfer
Wait for Acknowledge from master-receiver	Slave-transmit only	• As a slave-transmitter, the PWR_I2C Unit is responsible for releasing the PWR_SDA line to allow the master-receiver to pull the line low for the ACK • See Section 15.3.4.2.

Copyright © 2009 Marvell

Figure 36 through Figure 38 are examples of PWR_I2C transactions that show the relationships between master and slave devices.

Figure 36: Master-Transmitter Write to Slave-Receiver

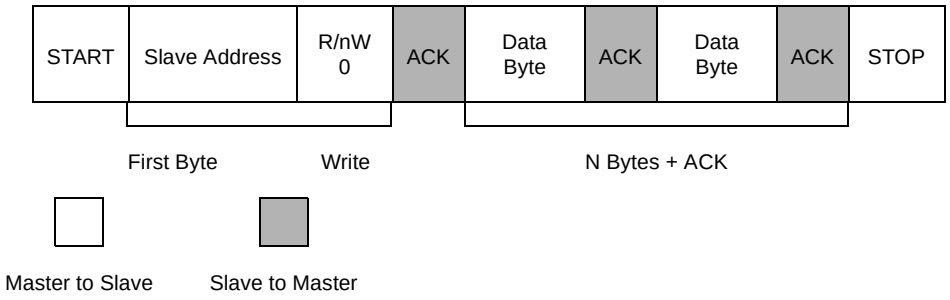


Figure 37: Master-Receiver Read to Slave-Transmitter

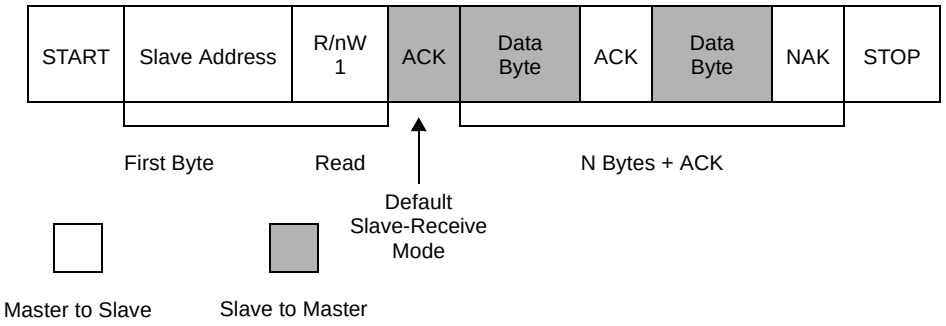
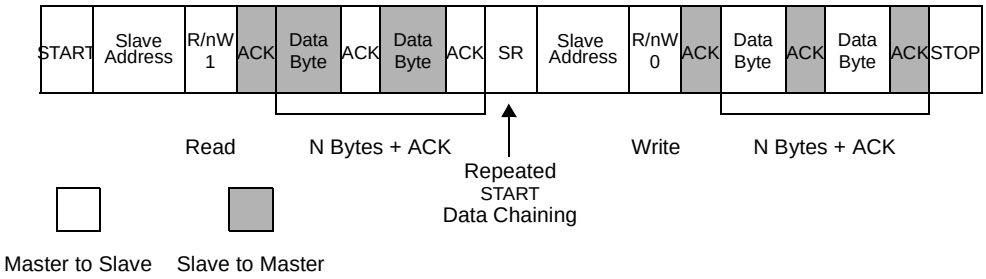


Figure 38: Master-Receiver Read to Slave-Transmitter, Repeated START**



***full title: Master-Receiver Read to Slave Transmitter, Repeated START, Master-Transmitter Write to Slave Receiver*

15.3.6 Slave Mode Programming Examples

15.3.6.1 Initialize Unit

1. Set the slave address in the PISAR.
2. Enable preferred interrupts in the PICR.
3. If the I^2C unit is a Hs-mode slave, set PICR[MODE] = 0b10 or 0b11
4. Set the PICR[IUE] bit to enable the I^2C interface.

15.3.6.2 Write *n* Bytes as a Slave

1. When a slave-address-detected interrupt occurs:
   Read PISR register: PISR[SAD] = 1, PISR[UB] = 1, PISR[RWM] = 1, PISR[ACKNAK] = 0
2. Write a 1 to the PISR[SAD] bit to clear the interrupt.
3. Return from interrupt.
4. Load data byte to transfer in the PIDBR.
5. Set PICR[TB] bit.
6. When a PIDBR Transmit-Empty interrupt occurs:
   Read PISR register: PISR[ITE] = 1, PISR[ACKNAK] = 0, PISR[RWM] = 0
7. Load data byte to transfer in the PIDBR.
8. Set the PICR[TB] bit.
9. Write a 1 to the PISR[ITE] bit to clear interrupt.
10. Return from interrupt.
11. Repeat steps 6 to 10 for *n*-1 times. If, at any time, the slave does not have data, the I^2C interface keeps SCL low until data is available.
12. When a PIDBR Transmit-Empty interrupt occurs:
    Read PISR register: PISR[ITE] = 1, PISR[ACKNAK] = 1, PISR[RWM] = 0
13. Write a 1 to the PISR[ITE] bit to clear interrupt.
14. Return from interrupt
15. When a Slave-Stop-Detected interrupt occurs:
    Read PISR register: PISR[UB] = 0, PISR[SSD] = 1
16. Write a 1 to the PISR[SSD] bit to clear interrupt.

Copyright © 2009 Marvell

15.3.6.3 Read *n* Bytes as a Slave

1. When a slave-address-detected interrupt occurs:
   Read PISR register: PISR[SAD] = 1, PISR[UB] = 1, PISR[RWM] = 0
2. Write a 1 to the PISR[SAD] bit to clear the interrupt.
3. Return from interrupt.
4. Set PICR[TB] bit to initiate the transfer.
5. When an PIDBR Receive-Full interrupt occurs:
   Read PISR register: PISR[IRF] = 1, PISR[ACKNAK] = 0, PISR[RWM] = 0
6. Read PIDBR to get the received byte.
7. Write a 1 to the PISR[IRF] bit to clear interrupt.
8. Return from interrupt.
9. Repeat steps 4 to 8 for *n*-1 times. Once the PIDBR is full, the I^2C interface keeps SCL low until the data is read.
10. Set PICR[TB] bit to release PWR_I2C bus and allow next transfer.
11. When a Slave-Stop-Detected interrupt occurs:
    Read PISR register: PISR[UB] = 0, PISR[SSD] = 1
12. Write a 1 to the PISR[SSD] bit to clear interrupt.

15.3.6.4 General Call Address

A general call address is defined as a transaction with a slave address of 00H. When a device requires the data from a general call address, it acknowledges the transaction and remains in Slave-Receiver mode; otherwise, the device can ignore the general call address. The second and subsequent bytes of a general call transaction are acknowledged by every device using it on the bus. Any device not using these bytes must not Ack. The meaning of a general call address is defined in the second byte sent by the master transmitter. Figure 39 shows a general call address transaction. The least significant bit (B) of the second byte defines the transaction. Table 191 shows the valid values and definitions when B=0.

The PWR_I2C unit supports both sending and receiving general call-address transfers on the PWR_I2C bus. When sending a general call message from the PWR_I2C unit, software must set the PICR[GCD] to keep the PWR_I2C unit from responding as a slave. Failure to set this bit causes the PWR_I2C bus to enter an indeterminate state.

The PWR_I2C unit performs the following operations when the PWR_I2C unit acts as a slave and receives a general call address while the PICR[GCD] bit is clear:

- Sets the PISR[GCAD] bit
- Sets the PISR[SAD] bit
- Interrupts (when enabled) the processor.

The PWR_I2C unit ignores the general call address when the PWR_I2C unit receives a general call address and the PICR[GCD] bit is set.

Figure 39: General Call Address

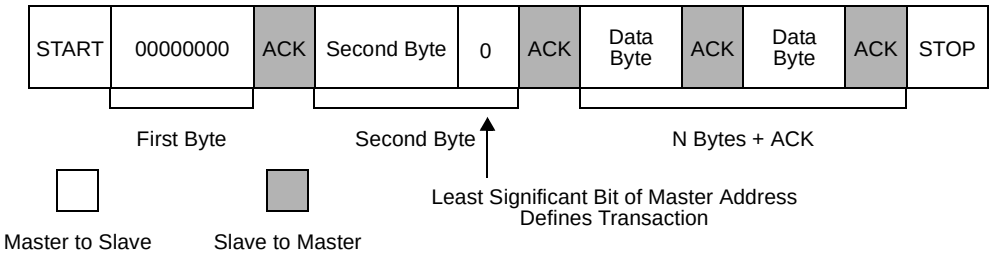


Table 191: General Call Address Second Byte Definitions

Least Significant Bit of Second Byte (B)	Second Byte Value	Definition
0	06H	2-byte transaction where the second byte tells the slave to reset and then store this value in the programmable part of its slave address
0	04H	2-byte transaction where the second byte tells the slave to store this value in the programmable part of its slave address; no reset
0	00H	Not allowed as a second byte

The PWR_I2C unit returns to its default reset condition when directed to reset, with the exception of the PISAR. The CPU is responsible for ensuring this reset occurs, not the PWR_I2C unit hardware.

When B=1, the sequence is used as a hardware general call by hardware masters. They cannot transmit a slave address, only their own address. The PWR_I2C unit does not support this mode of operation.

PWR_I2C 10-bit addressing and CBUS compatibility are not supported.

15.3.7 Glitch Suppression Logic

The PWR_I2C unit has built-in glitch-suppression logic. Glitches are suppressed for a length of time given by: 2 * (1 / PWR_I2C input clock frequency). For example, with a 13 MHz input clock frequency, glitches of 154 ns or less are suppressed, which is within the 50 ns glitch suppression specification. For glitch suppression to be within specification, (1 / PWR_I2C input clock frequency) > 25 ns.

15.3.8 Reset Conditions

Software must ensure that the PWR_I2C unit is not busy (PISR[UB] = 0) before it asserts reset. Software must also ensure that the PWR_I2C bus is idle (PISR[IBB] = 0) when the unit is enabled after reset. When directed to reset, the PWR_I2C interface, except for the PISAR, returns to its default reset condition, the PISAR as it is not affected by a reset.

When the PICR[UR] is set, the PWR_I2C interface resets but the associated PWR_I2C MMRs remain intact. Use the following guidelines when resetting the PWR_I2C interface with the PICR unit reset:

Copyright © 2009 Marvell

1. In the PICR register, set the reset bit and clear the remainder of the register.
2. Clear the PISR register.
3. Clear reset in the PICR.

Note

The I^2C specification does not account for a situation where one device (master) is reset and the other device (slave) is not. The result of such a situation occurring is that the slave device will be waiting to finish a transfer when the master comes out of reset. When a processor reset occurs (GPIO or hardware), software must be made aware of this condition to avoid a processor (master) reset while the external device (slave) is not reset. If the external device cannot be reset along with the processor, software must be programmed to stop the current transfer when it resumes from reset.

15.4 Register Descriptions

The following PWR_I2C registers are located within the peripheral memory-mapped address space of the processor. The bit definitions for the Power Manager PI2C register are identical to those for the standard I2C unit.

Table 192: Power Manager PWR_I2C Register Summary

Physical Addresses	Name	Description	Page
0x40F5_00C8	PICR	Power Manager I2C Control register	
0x40F5_00CC	PISR	Power Manager I2C Status register	
0x40F5_00D0	PISAR	Power Manager I2C Slave Address register	
0x40F5_00C4	PIDBR	Power Manager I2C Data Buffer register	
0x40F5_00C0	PIBMR	Power Manager I2C Bus Monitor register	

15.4.1 Power Manager I2C Control Register (PICR)

PICR, shown in Table 193, provides the means for the processor to control the PWR_I2C unit.

Table 193: PICR Bit Definitions

	Physical Address 0x40F5_00C8																															
	PICR																Power Manager PWR_I2C															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																CTL[15:0]															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	Reserved
CTL15	R/W	MODE	0 = Reserved. 1 = Up to 380 Kbps. operation
CTL14	R/W	UR	0 = No reset. 1 = Reset the PWR_I2C unit only.
CTL13	R/W	SADIE	Slave Address Interrupt Enable 0 = Disable interrupt. 1 = Enables the PWR_I2C unit to interrupt the processor upon detecting a slave address match or a general call address.
CTL12	R/W	ALDIE	Arbitration Lost Interrupt Enable 0 = Disable interrupt. 1 = Enables the PWR_I2C unit to interrupt the processor upon losing arbitration while in master mode.
CTL11	R/W	SSDIE	0 = Disable interrupt. 1 = Enables the PWR_I2C unit to interrupt the processor when it detects a STOP condition while in Slave mode.
CTL10	R/W	BEIE	Enables the PWR_I2C unit to interrupt the processor for the following PI2C bus errors: • As a master transmitter, no ACK is detected after a byte is sent. • As a slave receiver, the PWR_I2C unit generates a NAK pulse. **NOTE:** Software is responsible for guaranteeing that misplaced START and STOP conditions do not occur. See Section 15.3.7. 0 = Disable interrupt. 1 = Enable interrupt.
CTL9	R/W	DRFIE	0 = Disable interrupt. 1 = Enables the PWR_I2C unit to interrupt the processor when the PIDBR receives a data byte from the PWR_I2C bus.
CTL8	R/W	ITEIE	0 = Disable interrupt. 1 = Enables the PWR_I2C unit to interrupt the processor after transmitting a byte onto the PWR_I2C bus.
CTL7	R/W	GCD	General Call Disable 0 = Enables the PWR_I2C unit to respond to general call messages. 1 = Disables PWR_I2C unit response to general call messages as a slave. This bit must be set when sending a Master mode general call message from the PWR_I2C unit.

Copyright © 2009 Marvell

Table 193: PICR Bit Definitions (Continued)

Physical Address **0x40F5_00C8** **PICR** **Power Manager** PWR_I2C

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																**CTL[15:0]**															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
CTL6	R/W	IUE	Software must guarantee the PWR_I2C bus is idle before setting this bit. This can be done by verifying that PVCR[VSCA] is clear. 0 = Disables the unit and does not master any transactions or respond to any slave transactions. 1 = Enables the PWR_I2C unit (defaults to Slave-Receive mode).
CTL5	R/W	SCLE	0 = Disables the PWR_I2C unit from driving the PWR_SCL line. 1 = Enables the PWR_I2C clock output for Master mode operation.
CTL4	R/W	MA	Master Abort Used by the PWR_I2C unit when in Master mode to generate a STOP without transmitting another data byte. 0 = The PWR_I2C unit transmits STOP using the stop PICR bit only. 1 = The PWR_I2C unit sends STOP without data transmission. When in Master Transmit mode, after transmitting a data byte, the PICR transfer byte bit is cleared and PIDBR transmit empty bit is set. When no more data bytes need to be sent, setting master abort bit sends the STOP. The PICR[transfer byte] bit must remain clear. In Master-Receive mode, when a NAK is sent without a STOP (stopICR bit was not set) and the processor does not send a repeated START, setting this bit sends the STOP. Once again, the PICR [transfer byte] bit must remain clear.
CTL3	R/W	TB	Transfer Byte Used to send/receive a byte on the PWR_I2C bus. 0 = Cleared by PWR_I2C unit when the byte is sent/received. 1 = Send/receive a byte. The processor can monitor this bit to determine when the byte transfer has completed. In Master or Slave mode, after each byte transfer including ACK/NAK bit, the I^2C unit holds the PWR_SCL line low (inserting Wait states) until the transfer byte bit is set.
CTL2	R/W	ACKNAK	Defines the type of ACK pulse sent by the PWR_I2C unit when in Master Receive mode. 0 = The PWR_I2C unit sends an ACK pulse after receiving a data byte. 1 = The PWR_I2C unit sends a negative ACK (NAK) after receiving a data byte. The PWR_I2C unit automatically sends an ACK pulse when responding to its slave address or when responding in Slave-Receive mode, independent of the ACK/NAK control bit setting.
CTL1	R/W	STOP	Used to initiate a STOP condition after transferring the next data byte on the PWR_I2C bus when in Master mode. In Master-Receive mode, the ACK/NAK control bit must be set in conjunction with this bit. See Section 15.3.3.3 for more details on the STOP state. 0 = Do not send a STOP. 1 = Send a STOP.

Table 193: PICR Bit Definitions (Continued)

Physical Address 0x40F5_00C8 **PICR** **Power Manager** PWR_I2C

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																**CTL[15:0]**															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
CTL0	R/W	START	Used to initiate a START condition to the PWR_I2C unit when in Master mode. See Section 15.3.3.1 for more details on the START state. 0 = Do not send a START. 1 = Send a START.

15.4.2 Power Manager I2C Status Register (PISR)

PISR signals PI2C interrupts to the processor interrupt controller. Software uses the PISR bits to check the status of the PI2C unit and bus. PISR bits (bits 9-5) are updated after the ACK/NAK signal is de-asserted on the PI2C bus. See Table 194.

PISR is also used to clear interrupts signalled from the PWR_I2C Bus Interface unit and includes the following:

- PIDBR Receive Full
- PIDBR Transmit Empty
- Slave Address Detected
- Bus Error Detected
- STOP Condition Detect
- Arbitration Lost

Table 194: PISR Bit Definitions

Physical Address 0x40F5_00CC **PISR** **Power Manager** PWR_I2C

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved																					**Stat[10:0]**										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	—	reserved

Copyright © 2009 Marvell

Table 194: PISR Bit Definitions (Continued)

	Physical Address 0x40F5_00CC																															
	PISR																															
	Power Manager PWR_I2C																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved																					Stat[10:0]										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
Stat10	R/Write 1 to clear	BED	0 = :No error detected. 1 = The PWR_I2C unit sets this bit when it detects one of the following error conditions: • As a master transmitter, no ACK was detected on the interface after a byte was sent. • As a slave receiver, the PWR_I2C unit generates a NAK pulse. **NOTE:** When an error occurs, PWR_I2C bus transactions continue. Software must guarantee that misplaced START and STOP conditions do not occur. See Section 15.3.3.2.
Stat9	R/Write 1 to clear	SAD	0 = No slave address detected. 1 = PWR_I2C unit detected a 7-bit address that matches the general call address or PISAR. An interrupt is signalled when enabled in the PICR.
Stat8	R/Write 1 to clear	GCAD	0 = No general call address received. 1 = PWR_I2C unit received a general call address.
Stat7	R/Write 1 to clear	IRF	0 = :The PIDBR has not received a new data byte or the PWR_I2C unit is idle. 1 = The PIDBR register received a new data byte from the PWR_I2C bus. An interrupt is signalled when enabled in the PICR.
Stat6	R/Write 1 to clear	**ITE**	0 = The data byte is still being transmitted. 1 = The PWR_I2C unit has finished transmitting a data byte on the PWR_I2C bus. An interrupt is signalled when enabled in the PICR.
Stat5	R/Write 1 to clear	ALD	Arbitration Loss Detected: Used during multi-master operation. 0 = Cleared when arbitration is won or never took place. 1 = Set when the PWR_I2C unit loses arbitration.
Stat4	R/Write 1 to clear	SSD	0 = No STOP detected. 1 = The PWR_I2C unit detects a STOP in Slave-Receive or Slave-Transmit mode.
Stat3	R	IBB	0 = PWR_I2C bus is idle or the PWR_I2C unit is using the bus. 1 = The PI2C bus is busy but the processor PWR_I2C unit is not involved in the transaction.
Stat2	R	**UB**	0 = PWR_I2C unit not busy. 1 = The processor PWR_I2C unit is busy. This is defined as the time between the first START and STOP.

Table 194: PISR Bit Definitions (Continued)

Physical Address 0x40F5_00CC	PISR	Power Manager PWR_I2C

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved																					Stat[10:0]										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
Stat1	R	ACKNAK	This bit is used in Slave-Transmit mode to determine when the byte transferred is the last one. This bit is updated after each byte and ACK/NAK information is received. 0 = The PWR_I2C unit received or sent an ACK on the bus. 1 = The PWR_I2C unit received or sent a NAK.
Stat0	R	RWM	This is the R/nW bit of the slave address. Hardware automatically clears it after a stop state. 0 = The PWR_I2C unit is in Master-Transmit or Slave-Receive mode. 1 = The PWR_I2C unit is in Master-Receive or Slave-Transmit mode.

15.4.3 Power Manager I2C Slave Address Register (PISAR)

PISAR, shown in Table 195, defines the PWR_I2C unit 7-bit slave address to which the processor responds in Slave-Receive mode. This register, which is written by the processor before PWR_I2C operations are enabled, is fully programmable (no address is assigned to the PWR_I2C unit) so it can be set to a value other than those of any hard-wired PWR_I2C slave peripherals in the system. The PISAR is not affected by a processor reset.

Table 195: PISAR Bit Definitions

Physical Address 0x34F5_00D0	PISAR	Power Manager I^2C

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																									Slave Address						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	Reserved
6:0	R/W	Slave Address	The 7-bit address to which the PWR_I2C unit responds in Slave-Receive mode.

15.4.4 Power Manager I2C Data Buffer Register (PIDBR)

PIDBR, shown in Table 196, enables the processor to transmit and receive data from the PWR_I2C bus. The accesses come from the PIDBR by the processor on one side and by the PWR_I2C Shift register on the other. Data entering the PWR_I2C bus interface unit is received into the PIDBR after a full byte is received and acknowledged. The processor writes data going out of the PWR_I2C bus interface unit to PIDBR, and it is sent to the serial bus.

Copyright © 2009 Marvell

When the PWR_I2C Bus interface unit is in Transmit mode (master or slave):

1. The processor writes data to the PIDBR over the internal bus. This data Write occurs when a master transaction is initiated or when the PIDBR Transmit Empty Interrupt is signalled.
2. Data is moved from the PIDBR to the Shift register when the transfer byte bit is set. The PIDBR Transmit Empty Interrupt is signalled (if enabled) when a byte is transferred on the PWR_I2C bus and the acknowledge cycle is complete.
3. If the processor does not write to the PIDBR (and a STOP condition is not in place) before the PWR_I2C bus is ready to transfer the next byte packet, the PWR_I2C unit inserts wait states until the processor writes the PIDBR and sets the transfer byte bit.

When the PWR_I2C unit is in Receive mode (master or slave):

1. The processor reads PIDBR data over the internal bus. This data Read occurs when the PIDBR Receive Full Interrupt is signalled.
2. The data is moved from the shift register to the PIDBR when the ACK cycle completes.
3. The PWR_I2C unit inserts wait states until the PIDBR is read. Refer to Section 15.3.4.2 for acknowledge-pulse information in Receiver mode.
4. After the processor reads the PIDBR, the ACK/NAK control bit is written and the transfer byte bit is written, allowing the next byte transfer to proceed on the PWR_I2C bus.

Table 196 shows the format of the PIDBR.

Table 196: PIDBR Bit Definitions

	Physical Address 0x40F5_00C4																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								**Data Buffer**							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	**Access**	**Name**	**Description**
31:8	—	—	Reserved
7:0	R/W	Data Buffer	Buffer for PWR_I2C bus send/receive data.

15.4.5 Power Manager I2C Bus Monitor Register (PIBMR)

PIBMR, shown in Table 197, tracks the status of the PWR_SCL and PWR_SDA pins. The values of these pins are recorded in this read-only register so that software can determine whether the PWR_I2C bus is hung up and the I^2C unit must be reset.

Table 197: PIBMR Bit Definitions

	Physical Address 0x40F5_00C0										PIBMR										Power Manager PWR_I2C											
User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved																														PWR_SCL	PWR_SDA
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1

Bits	Access	Name	Description
31:2	—	—	Reserved
1	R	SCL	PIBMR[SCL]: This bit continuously reflects the value of the PWR_SCL pin.
0	R	SDA	PIBMR[SDA] Status: This bit continuously reflects the value of the PWR_SDA pin.

Copyright © 2009 Marvell

16 Memory Map

16.1 Overview

This chapter presents the physical address map for the processor. Software can use the memory management unit in the processor to map these registers into a virtual address map, as required. Figure 40 shows the physical address map decode regions, Table 201 shows the summary memory map for region 0x0000_0000 to 0x7FFF_FFFF, and Table 200 shows the summary memory map for region 0x8000_0000 to 0xFFFF_FFFF.

Accessing reserved portions of the memory map shown in Table 199 or Table 200 results in a data-abort exception. Accessing reserved portions of a particular peripheral's address space does not cause a data-abort exception, but the data returned is undefined. The internal boot ROM is mapped to two different address spaces, virtually to 0x0000_0000– 0x0000_BFFF and physically to 0x5E00_0000 - 0x5E00_BFFF (for details, see the chapter on internal memory). At all times it appears in the 0x5E00_0000 - 0x5E00_BFFF region. Before handing control to any OBM image, the boot ROM remaps the 0x0000_0000 - 0x0000_BFFF address space to an external flash memory.

Figure 40: Physical Address Map Decode Regions

processor Physical Address Mapping - Decode Regions																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Block Decode (64 Blocks, 64 Mbyte each)						Unit Decode (64 Units, 1 Mbyte each) processor.						Sub-Unit Decode (1 Mbyte); for processor.																			

16.2 PXA3xx Processor Differences

Table 198 shows the Memory Map differences among the PXA32x, PXA31x and PXA30x processors.

Table 198: PXA3xx Processors Feature Differences

Address Range	PXA30x	PXA31x	PXA32x
0x0000_0000 - 0x0FFF_FFFF	Static Memory Controller Chip Select 0	Static Memory Controller Chip Select 0	Reserved
0x2000_0000 - 0x23FF_FFFF	Reserved	Reserved	CF - I/O Space
0x2800_0000 - 0x2BFF_FFFF	Reserved	Reserved	CF - Attribute Memory Space
0x2C00_0000 - 0x2FFF_FFFF	Reserved	Reserved	CF - Common Memory Space
0x3000_0000 - 0x3FFF_FFFF	Static Memory Controller Chip Select 1	Static Memory Controller Chip Select 1	Reserved

Copyright © 2009 Marvell

Table 198: PXA3xx Processors Feature Differences (Continued)

Address Range	PXA30x	PXA31x	PXA32x
0x41C0_0000–0x41A_FFFF	Reserved	Reserved	Touch Screen Interface
0x4250_0000–0x4250_004F	Reserved	MMC/SD/SDIO Controller #3 Registers	Reserved
0x5600_0000 –0x562F_FFFF	Reserved	Hardware Video Accelerator Registers	Reserved
0xA000_0000 -0xBFFF_FFFF	Reserved	Reserved	Dynamic Memory Controller Chip Select 0[1]
0xE000_0000 -0xFFFF_FFFF	Reserved	Reserved	Dynamic Memory Controller Chip Select 1[1]
NOTE: 1. Requires a 32-bit bus			

Table 199: Memory Map (Part 1) — From 0x0000_0000 to 0x7FFF_FFFF

	Memory Map
0x7C00_0000	Reserved (64 MB)
0x7800_0000	Reserved (64 MB)
0x7400_0000	Reserved (64 MB)
0x7000_0000	Reserved (64 MB)
0x6C00_0000	Reserved (64 MB)
0x6800_0000	Reserved (64 MB)
0x6400_0000	Reserved (64 MB)
0x6000_0000	Reserved (64 MB)
0x5C00_0000	Internal SRAM (256 KB of 64 MB) and Internal Boot ROM (48K)[1]
0x5800_0000	System Bus 1 (Internal Memory) Memory-Mapped registers[1]
0x5400_0000	System Bus 2 Memory-Mapped registers[1]
0x5000_0000	System Bus 1 (Camera Interface) Memory-Mapped registers[1]
0x4C00_0000	System Bus 1 (USB Host) Memory-Mapped registers[1]
0x4800_0000	System Bus 1 (DMEMC and SMEMC) Memory-Mapped registers[1]
0x4400_0000	System Bus 1 (LCD and Mini-LCD) Memory-Mapped registers[1]

Copyright © 2009 Marvell

Table 199: Memory Map (Part 1) — From 0x0000_0000 to 0x7FFF_FFFF (Continued)

	Memory Map
0x4000_0000	Peripheral Bus/System Bus 1 (Peripherals) Memory-Mapped registers[1]
0x3C00_0000	Static Chip Select 1 (256 MB) (PXA30x and PXA31x Only) Reserved (PXA32x Only)
0x3800_0000	
0x3400_0000	
0x3000_0000	
0x2C00_0000	Reserved (PXA30x and PXA31x Only) CF - Common Memory Space (64 MB) (PXA32x Only)
0x2800_0000	Reserved (PXA30x and PXA31x Only) CF - Attribute Memory Space (64 MB) (PXA32x Only)
0x2400_0000	Reserved (64 MB)
0x2000_0000	Reserved (PXA30x and PXA31x Only) CF - I/O Space (64 MB) (PXA32x Only)
0x2C00_0000	Reserved (64 MB)
0x2800_0000	Reserved (64 MB)
0x2400_0000	Reserved (64 MB)
0x2000_0000	Reserved (64 MB)
0x1C00_0000	Reserved (64 MB)
0x1800_0000	Reserved (64 MB)
0x1400_0000	Static Chip Select 3 (64 MB)
0x1000_0000	Static Chip Select 2 (64 MB)
0x0C00_0000	Static Chip Select 0 (256 MB) (PXA30x and PXA31x Only) Reserved (PXA32x Only)
0x0800_0000	
0x0400_0000	
0x0000_0000	

Table 200: Memory Map (Part Two) — From 0x8000_0000 to 0xFFFF FFFF

	Memory Map
0xFC00_0000	Reserved (PXA30x and PXA31x Only) DDR SDRAM Chip Select 1 (512 MB) (PXA32x Only)
0xF800_0000	
0xF400_0000	
0xF000_0000	
0xEC00_0000	
0xE800_0000	
0xE400_0000	
0xE000_0000	
0xDC00_0000	DDR SDRAM Chip Select 1 (512 MB)
0xD800_0000	
0xD400_0000	
0xD000_0000	
0xCC00_0000	
0xC800_0000	
0xC400_0000	
0xC000_0000	
0xBC00_0000	Reserved (PXA30x and PXA31x) DDR SDRAM Chip Select 0 (512 MB) (PXA32x Only)
0xB800_0000	
0xB400_0000	
0xB000_0000	
0xAC00_0000	
0xA800_0000	
0xA400_0000	
0xA000_0000	

Copyright © 2009 Marvell

Table 200: Memory Map (Part Two) — From 0x8000_0000 to 0xFFFF FFFF

	Memory Map
0x9C00_0000	DDR SDRAM Chip Select 0 (512 MB)
0x9800_0000	
0x9400_0000	
0x9000_0000	
0x8C00_0000	
0x8800_0000	
0x8400_0000	
0x8000_0000	

Note:
1. See Table 201 for further information.

16.3 Memory-Mapped Registers Summary

Table 201 summarizes the memory map area from 0x4000_0000–0x5BFF_FFFF. This area contains memory-mapped registers stored within the various units and peripherals in the processor.

Table 201: Register Address Summary

Unit	Address
Peripheral Bus (Through System Bus 1)	
DMA Controller	0x4000_0000 - 0x400F_FFFF
UART 1 - Full Function UART	0x4010_0000 - 0x401F_FFFF
UART 2 - Bluetooth UART	0x4020_0000 - 0x402F_FFFF
I^2C	0x4030_0000 - 0x403F_FFFF
Reserved	0x4040_0000 - 0x404F_FFFF
AC'97	0x4050_0000 - 0x405F_FFFF
USB Client	0x4060_0000 - 0x406F_FFFF
UART 3 - SIR UART	0x4070_0000 - 0x407F_FFFF
Reserved	0x4080_0000 - 0x408F_FFFF
RTC	0x4090_0000 - 0x409F_FFFF
OS Timer	0x40A0_0000 - 0x40AF_FFFF
PWM0 and PWM2	0x40B0_0000 - 0x40BF_FFFF
PWM1 and PWM3	0x40C0_0000 - 0x40CF_FFFF

Copyright © 2009 Marvell

Table 201: Register Address Summary (Continued)

Unit	Address
Interrupt Control	0x40D0_0000 - 0x40DF_FFFF
GPIO Controller	0x40E0_0000 - 0x40EF_FFFF
Slave Power Manager (APMU), Services Power Manager (SPMU), and PWR_1^{2}C	0x40F0_0000 - 0x40FF_FFFF
SSP1	0x4100_0000 - 0x410F_FFFF
MMC/SD/SDIO 1	0x4110_0000- 0x411F_FFFF
Reserved	0x4120_0000 - 0x412F_FFFF
Services and Slave Clocks Controller (ACCU and SCCU)	0x4130_0000 - 0x413F_FFFF
Reserved	0x4140_0000 - 0x414F_FFFF
KeyPad Interface	0x4150_0000 - 0x415F_FFFF
USIM 1	0x4160_0000 - 0x416F_FFFF
SSP2	0x4170_0000 - 0x417F_FFFF
Reserved	0x4180_0000 - 0x418F_FFFF
SSP3	0x4190_0000 - 0x419F_FFFF
SSP4	0x41A0_0000 - 0x41AF_FFFF
1-Wire	0x41B0_0000 - 0x41BF_FFFF
Reserved (PXA30x and PXA31x Only)	0x41C0_0000 - 0x41CF_FFFF
Touch Screen Interface	
Consumer Infrared	0x41D0_0000 - 0x41DF_FFFF
Reserved	0x41E0_0000 - 0x41FF_FFFF
MMC/SD/SDIO 2	0x4200_0000 - 0x420F_FFFF
USIM 2	0x4210_0000 - 0x421F_FFFF
Reserved	0x4220_0000 – 0x4249_FFFF
Reserved (PXA32x and PXA30x Only)	0x4250_0000 – 0x4250_004F
MMC/SD/SDIO 3 (PXA31x Only)	
Reserved	0x4250_0050 – 0x430F_FFFF
System Bus 1	
NAND Flash Controller	0x4310_0000 - 0x431F_FFFF
Reserved	0x4320_0000 – 0x43FF_FFFF

Copyright © 2009 Marvell

Table 201: Register Address Summary (Continued)

Unit	Address
LCD Controller	0x4400_0000 - 0x440F_FFFF
Reserved	0x4410_0000 – 0x45FF_FFFF
Mini-LCD	0x4600_0000 – 0x4600_FDFF
System Bus Arbiters	0x4600_FE00 – 0x4600_FEFF
Performance Monitoring and Debug	0x4600_FF00 – 0x4600_FFFF
Reserved	0x4601_0000 – 0x480F_FFFF
Dynamic Memory Controller	0x4810_0000 - 0x4B1F_FFFF
Reserved	0x4820_0000 – 0x49FF_FFFF
Static Memory Controller	0x4A00_0000 - 0x4A0F_FFFF
Reserved	0x4A10_0000 – 0x4BFF_FFFF
USB Host 1.1	0x4C00_0000 - 0x4C0F_FFFF
Reserved	0x4C10_0000 – 0x4FFF_FFFF
Quick Capture Interface	0x5000_0000 - 0x500F_FFFF
Reserved	0x5010_0000 – 0x53FF_FFFF
System Bus 2	
2D Graphics Controller	0x5400_0000 - 0x540F_FFFF
USB Host 2.0	0x5410_0000 - 0x541F_FFFF
Reserved	0x5420_0000 – 0x55FF FFFF
Reserved (PXA32x and PXA30x Only)	0x5600_0000 – 0x562F_FFFF
Video Accelerator (PXA31x Only)	
Reserved	0x5630_0000 – 0x57FF FFFF
System Bus 1	
Internal SRAM control space	0x5800_0000
Reserved	0x5810_0000 – 0x5BFF_FFFF

16.4 Boot ROM Space

The internal Boot ROM appears in up to two places in the address map. At all times it appears in the 0x5E00_0000 – 0x5E00_FFFF region; however it can also be virtually mapped to address 0x0000_0000 – 0x0000_FFFF. The functionality of both mappings is identical. This Boot ROM is used to execute the initial start-up code in all cases of reset and to provide various options for booting the device. Additional information is provided in Section 2.15, Power-On Reset and Boot Operation.

For more details on Boot ROM addresses and requirements refer to the *PXA3xx Processor Boot ROM Reference Manual*.

Table 202: Boot ROM Locations

Unit	Address
Boot position	
Boot ROM	0x5E00_0000 – 0x5E00_FFFF
IM position	
Boot ROM	0x0000_0000 – 0x0000_FFFF

Copyright © 2009 Marvell

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

Marvell. Moving Forward Faster