

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

OPERATING MANUAL

MODEL 4292/4293

TMS320C6202/TMS320C6203
Quad/Octal Digital Signal Processor
for VMEbus with VIM and PMC I/O

Pentek, Inc.
One Park Way
Upper Saddle River, NJ 07458
(201) 818–5900
http://www.pentek.com/

Copyright © 2006

Manual Part No: 800.42920 Rev: A.7 – August 2, 2006

Manual Revision History

Date	Rev	Comments
7/17/01	Preliminary	Initial release of Operating Manual
12/9/02	A	Added Octal Model 4293 with 'C6x Slave DSP. Removed PCB jumpers for 'C6x boot options, i960 ID EEPROM, Sec 2.2. Moved Test Points from App D. to Sec 2.8. Added description of 'C6x internal resources, Sec 3.2. Added DSP register interrupt, Sec 3.6.8.2, to replace TOUT1 interrupt. Added caution for loading user code into first 64 kB of Local SDRAM, Sec. 3.5, per KBcase 1008. Corrected Slave DSP XBus HPI XBD/XBISA addresses, Sec. 3.5.6, per KBcase 1110. Added note that flag bits don't clear if status still set, Sec. 3.6, per KBcase 1026. Added two Misc. Int. Registers #2, Sec. 3.6. Updated 'C6x External Interrupt descriptions & added flows, Sec. 3.6.10 to 3.6.13, & 9.5. Changed Global Bus BIFO to Incoming (DSP) FIFO only, Ch 3 & Ch 9. Added note that DSPs will drop first word of each write to FIFOs, Sec 9.4.1, per KBcase 1055. Added Command Processor, Ch 12. Added data transfer benchmarks, App. C, & RACEway rates, App. D per KBcase 1072. Added RACEway restrictions, App D, per KBcase 1060. Added App F, list of Universe II Regs.
2/11/03	A.1	Corrected bit settings for VIM Module Clock, Sect. 3.6.8.2.
2/20/03	A.2	Corrected bit layout for MCR, Table 9–1, Sect. 9.2, per KBcase 1152.
12/2/03	A.3	Added Pentek Board Support Software, Sect. 1.6, & factory default jumpers, Sect. 2.2.9. Corrected footnote references, Table 3–1. Added RACEway references in memory maps, Sects. 3.5, 5.3, 7.3, & 7.3.2.
3/11/04	A.4	Added note for Slave SDRAM usage from Master, Sect. 4.4, per KBcase 1206
12/6/04	A.5	Corrected definition of jumper J21, System Reset Monitor, Sect 2.2.7.
7/21/05	A.6	Reversed default setting of jumper J5, Flash Write Enable, Sect 2.2.3.
8/2/06	A.7	Corrected definition of DSP Firmware Write (Set) & Read (Look) functions, Table 10–8, Sect 10.5.

Warranty

Pentek warrants that all products manufactured by Pentek conform to published Pentek specifications and are free from defects in materials and workmanship for a period of one year from the date of delivery when used under normal operating conditions and within the service conditions for which they were furnished. The obligation of Pentek arising from a warranty claim shall be limited to repairing or, at its option, replacing without charge any product that in Pentek's sole opinion proves to be defective within the scope of the warranty. Pentek must be notified in writing of the defect or nonconformity within the warranty period and the affected product returned to Pentek within thirty days after discovery of such defect or nonconformity.

Buyer shall prepay shipping charges, taxes, duties, and insurance for products returned to Pentek for warranty service. Pentek shall pay for the return of products to buyer except for products returned from another country.

Pentek shall have no responsibility for any defect or damage caused by improper installation, unauthorized modification, misuse, neglect, inadequate maintenance, accident, or for any product that has been repaired or altered by anyone other than Pentek or its authorized representatives.

The warranty described above is buyer's sole and exclusive remedy and no other warranty, whether written or oral, is expressed or implied. Pentek specifically disclaims fitness for a particular purpose. Under no circumstances shall Pentek be liable for any direct, indirect, special, incidental, or consequential damages, expenses, losses, or delays (including loss of profits) based on contract, tort, or any other legal theory.

The software in this product was in part provided by Red Hat and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Copyrights

With the exception of those items listed below, the contents of this publication are copyright © 2006, Pentek, Inc. All Rights Reserved. Contents of this publication may not be reproduced in any form without written permission.

Part of the software embedded in this product is eCos, embedded Configurable operating system, from Red Hat, Inc. Portions created by Red Hat are Copyright © 1998, 1999, 2000 Red Hat, Inc. (http://www.redhat.com/). All Rights Reserved.

Trademarks

Pentek, ReadyFlow, SwiftNet, and VIM are trademarks or registered trademarks of Pentek, Inc.

DuPont is a trademark of E.I. du Pont de Nemours and Company. eCos and RedBoot are trademarks of Red Hat, Inc. Intel and i960 are registered trademarks of Intel Corporation. PLX Technology and the PLX logo are registered trademarks of PLX Technology, Inc. RACE, RACE++, and RACEway are registered trademarks or trademarks of Mercury Computer Systems, Inc. Tundra and Universe II are registered trademarks or trademarks of Tundra Semiconductor Corporation.

Printed in the United States of America.

Table of Contents

Page

Chapter 1: Introduction

1.1 General Description....17
1.2 Digital Signal Processing Nodes....17
1.2.1 Expansion Bus (XBus) Interface....17
Figure 1–1: Model 4292 Block Diagram (Quad Version)....18
Figure 1–2: Model 4293 Block Diagram (Octal Version)....19
1.2.2 External Memory (EMIF) Interface....20
1.2.3 Serial Port Interfaces....20
1.2.4 Velocity Interface Mezzanine (VIM) Interfaces....20
1.3 I/O Processor....21
1.4 Global Resources....21
1.4.1 Global Bus FIFOs....21
1.4.2 Global SDRAM....21
1.4.3 VMEbus Interface....22
1.4.4 PMC Site....22
1.4.5 RS–232/Ethernet Interface....22
1.4.6 Optional RACEway Interface (Option 005)....22
1.4.7 JTAG Test Bus Controller....23
Figure 1–3: Model 4292/4293 JTAG/XDS Connections....23
1.5 Theory of Operation....24
1.6 Board Support Software....25
1.7 SwiftNet Software....25
1.8 Supporting Documentation....26
1.9 Specifications....27

Chapter 2: Installation and Connections

2.1 Inspection....31
2.2 Jumper Settings....31
Figure 2–1: Model 4292/4293 PCB Assembly, Component Side....32
2.2.1 Board Power–Up Mode Jumpers—J1 & J2....33
Table 2–1: Jumpers J1 & J2 – Board Power–Up Mode....33
2.2.2 Endian Mode Jumper—J3....33
Table 2–2: Jumper J3 – Endian Mode....33
2.2.3 DSP FLASH Memory Write Enable Jumpers—J4 & J5....34
Table 2–3: Jumpers J4 & J5 – DSP FLASH Memory Write Enable....34
2.2.4 i960 Initialization Mode Jumpers—J10, J11, & J12....35
Table 2–4: Jumpers J10, J11, & J12 – i960 Initialization Mode....35
2.2.5 i960 FLASH Memory Write Enable Jumper—J19....35
Table 2–5: Jumper J19 – i960 FLASH Memory Write Enable....35

Table of Contents

Page

Chapter 2: Installation and Connections *(continued)*

2.2.6 VMEbus A16/A24 Slave Select Jumper—J20 36
Table 2–6: Jumper J20 – VMEbus A16/A24 Slave Select 36
2.2.7 VMEbus System Reset Monitor Jumper—J21 36
Table 2–7: Jumper J21 – System Reset Monitor 36
2.2.8 VMEbus Slave Base Address Jumpers—JB1 37
Table 2–8: Jumper Block JB1 – VMEbus A16/A24 Slave Base Address 37
2.2.9 Factory Default Jumpers 38
Table 2–9: Factory Default Jumpers 38
2.3 Signal Connectors 39
2.3.1 VIM Connectors 39
2.3.2 PMC Connectors 39
2.3.3 JTAG/XDS Connector 40
Table 2–10: JTAG/XDS Connector (P7) 40
2.3.4 i960 Emulator Connector 40
Table 2–11: i960 Emulator Connector (P15) 40
2.4 DSP Timer Test Points 41
Table 2–12: 'C6x Timer Test Points 41
Figure 2–2: Model 4292/4293 PCB Assembly, Solder Side 42
2.5 Installation 43
2.5.1 Installing VIM and PMC Modules 43
2.5.2 Installing the Model 4292/4293 in the VME Card Cage 43
2.5.3 Removing the Model 4292/4293 from the VME Card Cage 43
2.6 Front Panel 44
Figure 2–3: Model 4292/4293 Front Panel 44
2.6.1 Reset Button 44
2.6.2 Power LED 44
2.6.3 Over–Temperature LED 44
2.6.4 I/O LEDs 45
2.6.5 'C6x Processor LEDs 45
2.6.6 RS–232/Ethernet Connector 45
Table 2–13: RS–232/Ethernet Connector 45
Figure 2–4: RS–232/Ethernet Cable 46
Table 2–14: RS–232/Ethernet Cable Pin Connections 46
2.7 Board Reset 47
Figure 2–5: Model 4292/4293 Reset Logic 47

Table of Contents

Page

Chapter 3: Master 'C6x Processor Memory Maps and Resources

3.1 Overview....49
3.2 'C6x Processor Internal Resources....49
3.3 Master DSP Memory Map....50
Table 3–1: Master DSP Memory Map....50
3.4 Master DSP Local SDRAM Operation....51
3.5 Master DSP XBus Host Port Interface Operation....51
Table 3–2: Master DSP XBus Host Port Memory Map....51
3.5.1 Global SDRAM....52
3.5.2 Universe II Registers....52
3.5.3 PLX PCI 9054 Registers....52
3.5.4 VMEbus Master Access....52
Table 3–3: Master DSP VMEbus Master Memory Map....52
3.5.5 i960 Messaging Unit....53
3.5.6 Slave DSP XBus Host Port (Model 4293 only)....53
3.5.7 PMC Module Access....53
3.6 Master DSP Control Registers....54
Table 3–4: Master DSP Register Memory Map....54
3.6.1 FIFO Interrupt Enable Registers....55
Figure 3–1: FIFO Flags and Interrupts....55
Table 3–5: Global Bus FIFO Interrupt Enable Register (BIER0)....56
Table 3–6: XBus Inter–Processor XX Bi–FIFO Inter. Enable Register (BIER1)....56
Table 3–7: XBus Inter–Processor YY Bi–FIFO Inter. Enable Register (BIER2)....57
Table 3–8: Mezzanine Bi–FIFO Interrupt Enable Register (BIER3)....57
3.6.1.1 Incoming Mailbox....57
3.6.1.2 Incoming Almost Full....57
3.6.1.3 Incoming Not Almost Empty....58
3.6.1.4 Incoming Not Empty....58
3.6.1.5 Outgoing Mailbox....58
3.6.1.6 Outgoing Not Almost Full....58
3.6.1.7 Outgoing Almost Empty....58
3.6.1.8 Outgoing Empty....58
3.6.2 FIFO Flag Registers....59
Table 3–9: Global Bus FIFO Flag Register (BIFR0)....59
Table 3–10: XBus Inter–Processor XX Bi–FIFO Flag Register (BIFR1)....60
Table 3–11: XBus Inter–Processor YY Bi–FIFO Flag Register (BIFR2)....60
Table 3–12: Mezzanine Bi–FIFO Flag Register (BIFR3)....60
3.6.2.1 Incoming Mailbox....61
3.6.2.2 Incoming Almost Full....61
3.6.2.3 Incoming Not Almost Empty....61
3.6.2.4 Incoming Not Empty....61
3.6.2.5 Outgoing Mailbox....61

Table of Contents

Page

Chapter 3: Master 'C6x Processor Memory Maps and Resources *(continued)*

3.6.2.6 Outgoing Not Almost Full 61
3.6.2.7 Outgoing Almost Empty 61
3.6.2.8 Outgoing Empty 61
3.6.3 FIFO Status Registers 62
Table 3–13: Global Bus FIFO Status Register (BISR0) 62
Table 3–14: XBus Inter–Processor XX Bi–FIFO Status Register (BISR1) 63
Table 3–15: XBus Inter–Processor YY Bi–FIFO Status Register (BISR2) 63
Table 3–16: Mezzanine Bi–FIFO Status Register (BISR3) 63
3.6.4 Miscellaneous Interrupt Flag Register 64
Table 3–17: Miscellaneous Interrupt Flag Register (MIFR) 64
3.6.4.1 Slave DSP (Model 4293 only) 65
3.6.4.2 Semaphore 65
3.6.4.3 i960 PINT 65
3.6.4.4 i960 SwiftNet 65
3.6.4.5 VME 65
3.6.4.6 PLX 65
3.6.4.7 VIM 65
3.6.5 Miscellaneous Interrupt Status Register 66
Table 3–18: Miscellaneous Interrupt Status Register (MISR) 66
3.6.6 'C6x External Interrupt Enable Registers 67
Table 3–19: External Interrupt #4 Enable Register (EIER4) 67
Table 3–20: External Interrupt #5 Enable Register (EIER5) 68
Table 3–21: External Interrupt #6 Enable Register (EIER6) 68
Table 3–22: External Interrupt #7 Enable Register (EIER7) 68
3.6.6.1 Slave DSP (Model 4293 only) 69
3.6.6.2 Semaphore 69
3.6.6.3 i960 PINT 69
3.6.6.4 i960 SwiftNet 69
3.6.6.5 VME 69
3.6.6.6 PLX 69
3.6.6.7 VIM 70
3.6.6.8 Mezzanine BI–FIFO 70
3.6.6.9 XBus I/P YY BI–FIFO 70
3.6.6.10 XBus I/P XX BI–FIFO 70
3.6.6.11 Global Bus FIFO 70
3.6.7 Board Status Register 71
Table 3–23: Board Status Register (BSR) 71
3.6.7.1 FPGA Revision 71
3.6.7.2 Mezzanine Bi–FIFO Reset 71
3.6.7.3 Board Speed 72
3.6.7.4 VIM Module Present 72
3.6.7.5 Processor ID 72

Table of Contents

Page

Chapter 3: Master 'C6x Processor Memory Maps and Resources *(continued)*

3.6.8 Miscellaneous Control Register 73
Table 3–24: Miscellaneous Control Register (MCR) 73
3.6.8.1 Slave Interrupt Request (Model 4293 only) 73
3.6.8.2 VIM Module Clock 73
3.6.8.3 i960 Interrupt Request 74
3.6.8.4 VIM Module Reset 74
3.6.8.5 Mezzanine Bi–FIFO Reset 74
3.6.8.6 XBus XX Bi–FIFO Reset 74
3.6.8.7 Global Bus FIFO Reset 74
3.6.9 General Purpose Semaphore Register 75
Table 3–25: General Purpose Semaphore Register (GPSR) 75
3.6.9.1 Grant2, Grant1, Grant0 75
3.6.9.2 Request2, Request1, Request0 75
3.6.10 'C6x External Interrupt Mode Register 77
Table 3–26: External Interrupt Mode Register (EIMR) 77
Figure 3–2: 'C6x External Interrupt Flow 78
3.6.11 'C6x External Interrupt Status Register 78
Table 3–27: External Interrupt Status Register (EISR) 78
3.6.12 Miscellaneous Interrupt Flag Register #2 79
Table 3–28: Miscellaneous Interrupt Flag Register 2 (MIFR2) 79
Figure 3–3: FIFO Interrupt Flags Flow 80
3.6.12.1 Slave DSP (Model 4293 only) 81
3.6.12.2 Semaphore 81
3.6.12.3 i960 PINT 81
3.6.12.4 i960 SwiftNet 81
3.6.12.5 VME 81
3.6.12.6 PLX 82
3.6.12.7 VIM 82
3.6.12.8 Mezzanine BI–FIFO 82
3.6.12.9 XBus I/P YY BI–FIFO 82
3.6.12.10 XBus I/P XX BI–FIFO 82
3.6.12.11 Global Bus FIFO 82
3.6.13 Miscellaneous Interrupt Status Register #2 83
Table 3–29: Miscellaneous Interrupt Status Register 2 (MISR2) 83
Figure 3–4: FIFO Interrupt Status Flow 84

Table of Contents

Page

Chapter 4: Slave 'C6x Processor Memory Maps and Resources

4.1 Overview ... 85
4.2 'C6x Processor Internal Resources ... 85
4.3 Slave DSP Memory Map ... 86
Table 4–1: Slave DSP Memory Map ... 86
4.4 Slave DSP Local SDRAM Operation ... 87
4.5 Slave DSP Interrupt Operation ... 87
Figure 4–1: Model 4293 Slave DSP Interrupt Paths ... 87
4.5.1 Responding to Master DSP Interrupt ... 88
4.5.2 Responding to i960 Interrupt ... 88
4.5.3 Asserting Slave DSP Interrupt ... 88
4.5.3.1 Slave Interrupt to Master DSP ... 88
4.5.3.2 Slave Interrupt to i960 ... 88

Chapter 5: I/O Processor Memory Maps and Resources

5.1 Overview ... 89
5.2 i960 Processor Internal Resources ... 89
5.3 i960 Processor Memory Maps ... 90
Table 5–1: i960 Processor Memory Map ... 90
5.3.1 i960 PCI Bus I/O Address Map ... 91
Table 5–2: i960 PCI Bus I/O Map ... 91
5.3.2 i960 I^2C Bus Memory Map ... 91
Table 5–3: i960 I^2C Bus Memory Map ... 91
5.4 i960 Resources Operation ... 92
5.4.1 VMEbus Master Access ... 92
Table 5–4: VMEbus Master Memory Map ... 92
5.4.2 RS–232/Ethernet Interface ... 92
5.4.3 Universe II Registers ... 92
5.4.4 PMC Module Access ... 93
5.4.5 Global SDRAM ... 93
5.4.6 Test Bus Controller ... 93
5.4.7 DSP Expansion Bus Host Port ... 94
Table 5–5: DSP XBus Host Port Registers ... 94
5.4.8 PLX PCI 9054 Registers ... 95
Table 5–6: PLX PCI 9054 Registers ... 95
5.4.9 Voltage/Temperature Monitor ... 95
5.4.10 Board ID EEPROM ... 95

Table of Contents

Page

Chapter 5: I/O Processor Memory Maps and Resources (continued)

5.5 i960 Control Registers 96
Table 5–7: i960 Register Memory Map 96
5.5.1 DSP (SwiftNet) Interrupt Request Register 96
Table 5–8: DSP (SwiftNet) Interrupt Request Register 96
5.5.2 PLX PCI 9054 Interrupt Enable Register 97
Table 5–9: PLX PCI 9054 Interrupt Enable Register 97
5.5.3 DSP (SwiftNet) Interrupt Enable Register 97
Table 5–10: DSP (SwiftNet) Interrupt Enable Register 97
5.5.4 Global Bus FIFO Interrupt Enable Register 98
Table 5–11: Global Bus FIFO Interrupt Enable Register 98
5.5.5 Built–In Self Test Registers #0 and #1 98
Table 5–12: Built–In Self Test Register #0 98
Table 5–13: Built–In Self Test Register #1 99
5.5.5.1 Word Test 99
5.5.5.2 Ones Test 99
5.5.5.3 BIST Status 99
5.5.5.4 BIST Start 99
5.5.5.5 Node Tests 99
5.5.6 Control Register 100
Table 5–14: Control Register 100
5.5.6.1 Jumper J1, J2 100
5.5.6.2 PMC Present 100
5.5.6.3 TBC Ready 100
5.5.6.4 LED 100
5.5.6.5 TBC Reset 100
5.5.7 General Purpose Semaphore Register 101
Table 5–15: General Purpose Semaphore Register 101
5.5.7.1 Grant2, Grant1, Grant0 101
5.5.7.2 Request2, Request1, Request0 101
5.5.8 PLX PCI 9054 Interrupt Status Register 102
Table 5–16: PLX PCI 9054 Interrupt Status Register 102
5.5.9 DSP (SwiftNet) Interrupt Flag Register 103
Table 5–17: DSP (SwiftNet) Interrupt Flag Register 103
5.5.10 Global Bus FIFO Interrupt Status Register 104
Table 5–18: Global Bus FIFO Interrupt Status Register 104

Table of Contents

Page

Chapter 6: PCI Bus Memory Maps and Resources

6.1 Overview 105
6.2 PCI Configuration Space 105
Table 6–1: PCI Configuration Space Addresses 105
6.3 PCI Bus Memory Map 106
Table 6–2: PCI Bus Memory Map 106
6.4 PCI Bus I/O Map 107
Table 6–3: PCI Bus I/O Map 107
6.5 PCI Bus Resources Operation 107
6.5.1 PLX PCI 9054 Registers 107
Table 6–4: PLX PCI 9054 Registers 107
6.5.2 DSP Expansion Bus Host Port 108
Table 6–5: DSP XBus Host Port Registers 108
6.5.3 i960 Messaging Unit 109
6.5.4 Global SDRAM 109
6.5.5 Ethernet Interface 109
6.5.6 Universe II Registers 109
6.5.7 VMEbus Master Access 110
Table 6–6: VMEbus Master Memory Map 110
6.5.8 PMC Module Access 110

Chapter 7: VME Interface

7.1 Overview 111
7.2 VMEbus Master Access 111
Table 7–1: VMEbus Master Memory Map 111
7.3 VMEbus Slave Access 112
7.3.1 Default VMEbus A16/A24 Slave Memory Map 113
Table 7–2: Default VMEbus A16/A24 Slave Memory Map 113
Table 7–3: Universe II Mailbox Registers 113
7.3.2 Extended VMEbus Slave Memory Map 114
Table 7–4: Extended VMEbus A24/A32 Slave Memory Map 114
7.3.2.1 Universe II Registers 115
7.3.2.2 PLX PCI 9054 Registers 115
Table 7–5: PLX PCI 9054 Registers 115
7.3.2.3 PMC Access 116
7.3.2.4 i960 Messaging Unit 116
7.3.2.5 DSP Expansion Bus Host Port 117
Table 7–6: DSP XBus Host Port Registers 117
7.3.2.6 Global SDRAM 118
Table 7–7: VMEbus Global SDRAM Size/Addresses 118

Table of Contents

Page

Chapter 8: Interrupt Operation

8.1 Overview....119
8.2 General Interrupt Operation....119
Figure 8–1: Model 4292/4293 Interrupt Connections....120
8.3 Master 'C6x Processor Interrupts....121
Table 8–1: Master DSP Outgoing Interrupts....121
Table 8–2: Master DSP Incoming Interrupts....121
8.4 i960 Processor Interrupts....122
Table 8–3: i960 Processor Outgoing Interrupts....122
Table 8–4: i960 Processor Incoming Interrupts....122
8.5 PCI Doorbell Interrupts....123
8.5.1 Local–to–PCI Doorbell Register....123
8.5.2 PCI–to–Local Doorbell Register....123
8.6 VMEbus Interrupts....124
8.6.1 Handling VMEbus Interrupts....124
8.6.2 Interrupting the VMEbus....124

Chapter 9: FIFO Operation

9.1 Overview....125
Figure 9–1: Model 4292/4293 FIFO Connections....125
9.2 Resetting the FIFOs....126
Table 9–1: DSP Miscellaneous Control Register (MCR)....126
9.3 Initializing the FIFOs....127
9.4 FIFO Data and Mailbox Transfers....128
9.4.1 Master 'C6x FIFOs Operation....128
Table 9–2: 'C6x FIFOs XBus I/O Port Address Map....128
Table 9–3: 'C6x Global Bus FIFO XBus HPI Address Map....129
9.4.2 PCI Global Bus FIFOs Operation....129
Table 9–4: PCI Global Bus FIFO Address Map....129
9.4.3 i960 Global Bus FIFOs Operation....130
Table 9–5: i960 Global Bus FIFO Address Map....130
9.4.4 VMEbus Global Bus FIFOs Operation....131
Table 9–6: VMEbus Global Bus FIFO Address Map....131
9.5 FIFO Interrupts....132
Figure 9–2: FIFO Interrupt Logic – Rev 0....132
Figure 9–3: FIFO Interrupt Logic – Rev 1....133
Figure 9–4: Global Bus FIFO Interrupt Logic – Rev 1....134

Table of Contents

Page

Chapter 10: FLASH Memory and Boot Firmware

10.1 Overview 135
10.2 DSP Local FLASH Memory Operation 136
Table 10–1: DSP FLASH Memory Map 136
10.3 i960 FLASH Memory Operation 136
Table 10–2: i960 FLASH Memory Map 136
10.4 Firmware Boot Operation 137
10.4.1 Use of Global SDRAM During Boot 137
Table 10–3: Global SDRAM Boot Map 138
10.4.1.1 'C6x Processors Section 140
10.4.1.2 i960 Processor Section 140
10.4.1.3 User Data 140
Table 10–4: VIM ID EEPROM Example Code (Model 6230 shown) 141
Table 10–5: Model 4292 ID EEPROM Example Code (Model 4292 shown) 142
10.4.2 Use of Universe II Mailbox Registers During Boot 143
Table 10–6: Universe II Mailbox Registers Used During Boot 143
Table 10–7: 'C6x User Boot Option Codes 143
10.4.3 Master 'C6x Processor Boot Sequence 144
Figure 10–1: Master 'C6x Firmware Boot Sequence 146
10.4.4 Slave 'C6x Processor Boot Sequence 147
Figure 10–2: Slave 'C6x Firmware Boot Sequence 149
10.4.5 i960 Processor Boot Sequence 150
Figure 10–3: i960 Firmware Boot Sequence 151
10.5 DSP Firmware Functions 152
Table 10–8: DSP Firmware Function Summary 152
10.5.1 Using Firmware Functions 153
Table 10–9: PCI 9054 Mailbox and Doorbell Registers 153
10.5.2 Using Built–In Self Test (BIST) 154
Figure 10–4: Master 'C6x BIST Interrupt Sequence 154
Table 10–10: BIST Failure Results 155
Figure 10–5: Master 'C6x Firmware Function 0x3 Sequence 156
10.5.3 Checking for VIM Modules 156
10.6 Firmware Test Failure LED Codes 157
Figure 10–6: Model 4292/4293 Front Panel 157
Table 10–11: 'C6x Red LED Blink Failure Codes 157
10.7 User Boot Programs 158
10.7.1 Converting COFF Files into Intel HEX Format 158
10.7.2 Erasing DSP FLASH Memories 159
10.7.3 Loading HEX Files into DSP FLASH 160
10.7.4 Bypassing DSP User Boot Code 160

Table of Contents

Page

Chapter 11: The Monitor Program

11.1 Overview 161
11.2 Connecting to the Serial Port 161
11.3 Starting the Monitor's User Interface 162
11.4 Command Processor Commands 163
11.4.1 dump 163
11.4.2 reset 164
11.4.3 version 165
11.4.4 fill 165
11.4.5 help 166
11.4.6 edit 167
11.4.7 pedit 167
11.4.8 pdump 168
11.4.9 modify 169
11.4.10 display 170
11.5 RedBoot Monitor Commands 171
11.5.1 dump 171
11.5.2 reset 172
11.5.3 version 173
11.5.4 fill 173
11.5.5 help 174
11.5.6 edit 175
11.5.7 pedit 175
11.5.8 pdump 176
11.5.9 cksum 177
11.5.10 fis init 177
11.5.11 fis list 178
11.5.12 fis free 178
11.5.13 fis create 179
11.5.14 fis delete 180
11.5.15 fis load 180
11.5.16 fis lock 181
11.5.17 fis unlock 181
11.5.18 fis erase 182
11.5.19 fis write 182
11.5.20 fconfig 183
11.5.21 load 185
11.5.22 go 186
11.5.23 ping 187
11.5.24 vmeslaveconfig 188
11.5.25 vmemasterconfig 191
11.5.26 appnetconfig 194

Table of Contents

Page

Chapter 12: The Command Processor

12.1 Overview 197
12.2 Theory of Operation 197
12.2.1 Getting a Message Frame 197
Table 12–1: Inbound Queue Port Addresses 198
12.2.2 Executing a Message 198
12.2.3 Freeing a Message Frame 198
12.3 Message Format 199
12.3.1 Message Header Format 199
Table 12–2: Message Header Format 199
12.3.1.1 Command Status 199
12.3.1.2 Command Id 200
12.3.1.3 Message Status 200
12.3.1.4 Message Id 200
12.3.1.5 Message Size 200
12.3.1.6 Message Count 200
12.3.1.7 Nr of Commands 201
12.3.1.8 Number of Times Message Executed 201
12.3.1.9 Processor Id 201
12.3.1.10 Return Message Frame 201
12.3.1.11 Notification Type and Parameters 201
12.3.2 Command Format 205
Table 12–3: Command Header Format 205
12.3.2.1 Command Status 205
12.3.2.2 Command Id 205
12.3.2.3 Notification Type and Parameters 205
12.4 Commands 206
Table 12–4: Command Processor Command List 206
12.4.1 READ_I960 208
12.4.2 WRITE_I960 208
12.4.3 READ_DSP 209
12.4.4 WRITE_DSP 210
12.4.5 EXECUTE_PROGRAM 211
12.4.6 RESET_DSP_NODE 211
12.4.7 RELEASE_DSP_NODE 212
12.4.8 EXECUTE_DSP_PROGRAM 212
12.4.9 READ_PCI_CONFIG 213
12.4.10 WRITE_PCI_CONFIG 214

Table of Contents

Page

Chapter 12: The Command Processor *(continued)*

12.4.11 XFER_SDRAM_TO_VME 215
12.4.12 XFER_VME_TO_SDRAM 217
12.4.13 XFER_SDRAM_TO_PMC 219
12.4.14 XFER_PMC_TO_SDRAM 220
12.4.15 XFER_SDRAM_TO_DSP 221
12.4.16 XFER_DSP_TO_SDRAM 222
12.4.17 I960_XFER_SDRAM_TO_DSP 223
12.4.18 TCP_SOCKET 224
12.4.19 TCP_BIND 226
12.4.20 TCP_LISTEN 227
12.4.21 TCP_ACCEPT 228
12.4.22 TCP_CONNECT 229
12.4.23 TCP_READ 230
12.4.24 TCP_WRITE 231
12.4.25 TCP_CLOSE 232
12.4.26 TCP_SELECT 233
12.4.27 TCP_SENDTO 234
12.4.28 TCP_RECVFROM 236
12.4.29 TCP_SETSOCKOPT 238
12.4.30 TCP_GETSOCKOPT 239
12.4.31 IOCTL 240
12.4.32 GENERATE_VME_INT 241
12.4.33 WAIT_ON_VME_INT 241
12.4.34 GENERATE_SYSRESET 242
12.4.35 WAIT_ON_INT 242
12.4.36 GET_DATA_BUFFER 243
12.4.37 FREE_DATA_BUFFER 243
12.4.38 GET_SEMAPHORE 244
12.4.39 FREE_SEMAPHORE 245
12.4.40 FREE_MSG_FRAME 246
12.4.41 FREE_PROC_RESOURCES 246

Table of Contents

Page

Appendix A: Model 4292/4293 Registers Summary

A.1 Introduction A–1
A.2 DSP Local Control Registers A–1
A.3 i960 Control Registers A–4

Appendix B: Model 4292/4293 ID EEPROM

B.1 Introduction B–1
B.2 ID EEPROM Format and Contents B–1
Table B–1: EEPROM Example Code (Model 4292 illustrated) B–2

Appendix C: Data Transfer Benchmarks

C.1 Introduction C–1
C.2 Master 'C6x Processor Data Transfer Timing C–1
Table C–1: Master 'C6x Processor Data Transfer Timing C–1
C.3 i960 Processor Data Transfer Timing C–2
Table C–2: i960 Processor Data Transfer Timing C–2

Appendix D: Option 005 RACEway Operation

D.1 Introduction D–1
D.2 General Operation D–1
D.3 RACEway Transfer Rates D–2

Appendix E: Tundra® Universe II™ Registers

Table E–1: Universe II Control and Status Registers E–1

Chapter 1: Introduction

1.1 General Description

Pentek's Models 4292 and 4293 DSP boards represent a high level of processor density and I/O capability for a single−slot VMEbus Digital Signal Processing (DSP) board. Four (Model 4292) or eight (Model 4293) Texas Instruments TMS320C6202 or TMS320C6203 ('C6x) processors provide up to 19,200 MIPS of DSP throughput. The Model 4292/4293 provides one Peripheral Component Interconnect (PCI) Mezzanine Card (PMC) site and four Velocity Interface Mezzanine (VIM®) sites to support data exchange with I/O devices by the 'C6x processors. Block diagrams of both models are presented in Figures 1−1 and 1−2 on the following two pages.

1.2 Digital Signal Processing Nodes

The Model 4292/4293 has four DSP Nodes (Nodes A, B, C, and D). Each node has one Master 'C6x processor (Processors A, B, C, and D, respectively). On the Model 4293, each node also has one Slave 'C6x processor (Processors E, F, G, and H). Each Master 'C6x processor has direct access to its node's Expansion Bus (XBus) and External Memory Interface (EMIF) resources, and to board resources outside its node. Each Slave 'C6x has access only to its EMIF resources. Each node also has a dedicated Velocity Interface Mezzanine (VIM) site that is controlled by the node's Master 'C6x processor.

1.2.1 Expansion Bus (XBus) Interface

Each DSP node's XBus provides a direct interface to four high−speed 32−bit FIFOs. Each Master 'C6x processor can send and receive data blocks using these FIFOs without arbitrating for a shared resource.

- Two 2k x 32 XBus interprocessor bi−directional FIFOs (Bi−FIFOs) (1k in each direction) allow data exchange with the XBuses of two other nodes (Node A to Nodes B & C, Node B to Nodes A & D, Node C to Nodes A & D, and Node D to Nodes B & C, see Figures 1−1 and 1−2).

- A 2k x 32 Global Bus (input) FIFO allows receiving data from other board resources, including the i960 I/O Processor, VMEbus, the PMC site, and the (optional) RACEway® interface.

- A 2k x 32 Mezzanine Bi−FIFO (1k in each direction) buffers 32−bit parallel data transfers with the associated VIM site.

With Option 320, the size of each Bi−FIFO is 32k x 32 (16k in each direction).

Each DSP node's XBus also provides a direct interface to a PCI Bus Master Interface (PLX Technology® PCI 9054) that allows each Master 'C6x to access the board's PCI buses and other board resources.

1.2 Digital Signal Processing Nodes (continued)

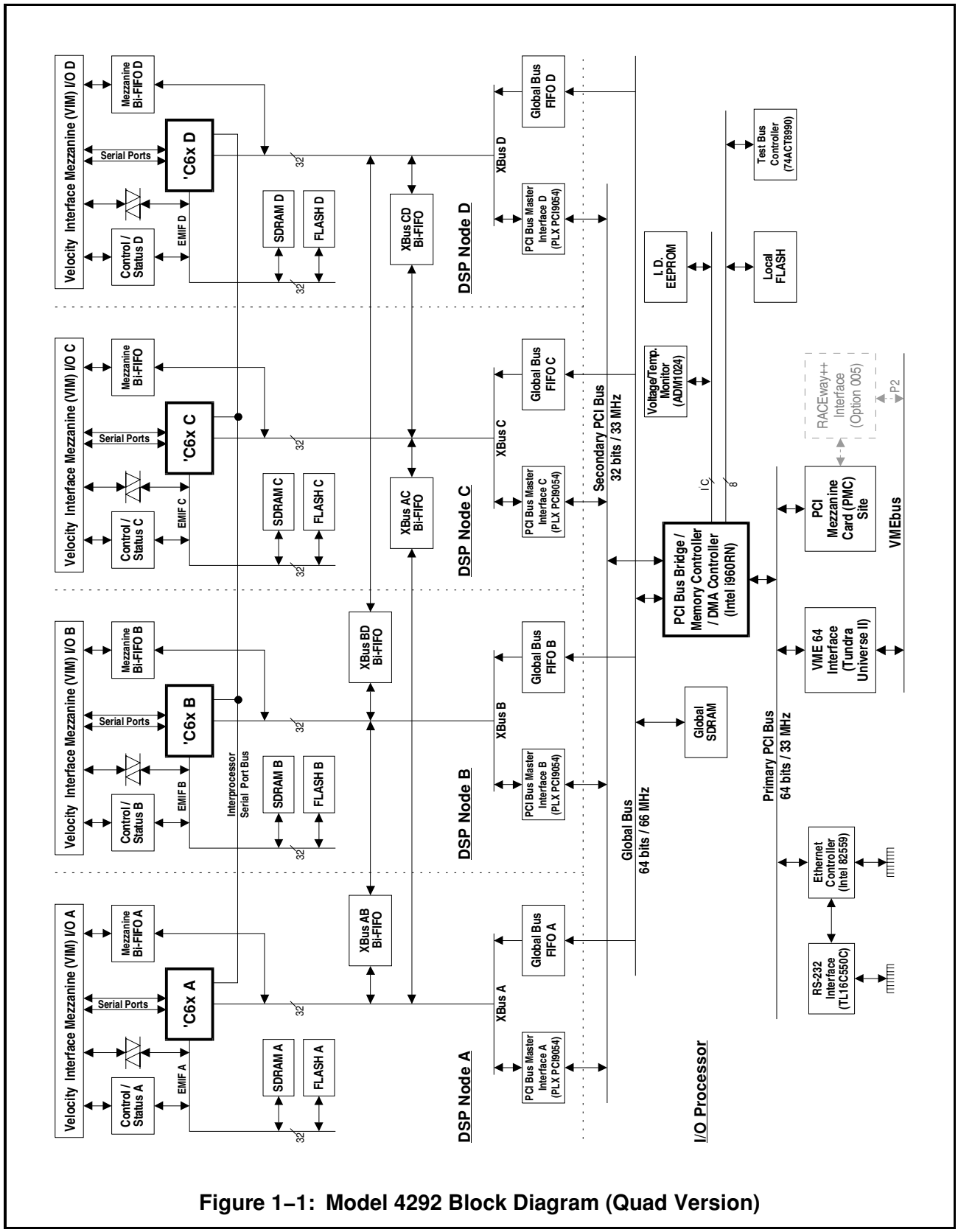


Figure 1–1: Model 4292 Block Diagram (Quad Version)

1.2 Digital Signal Processing Nodes (continued)

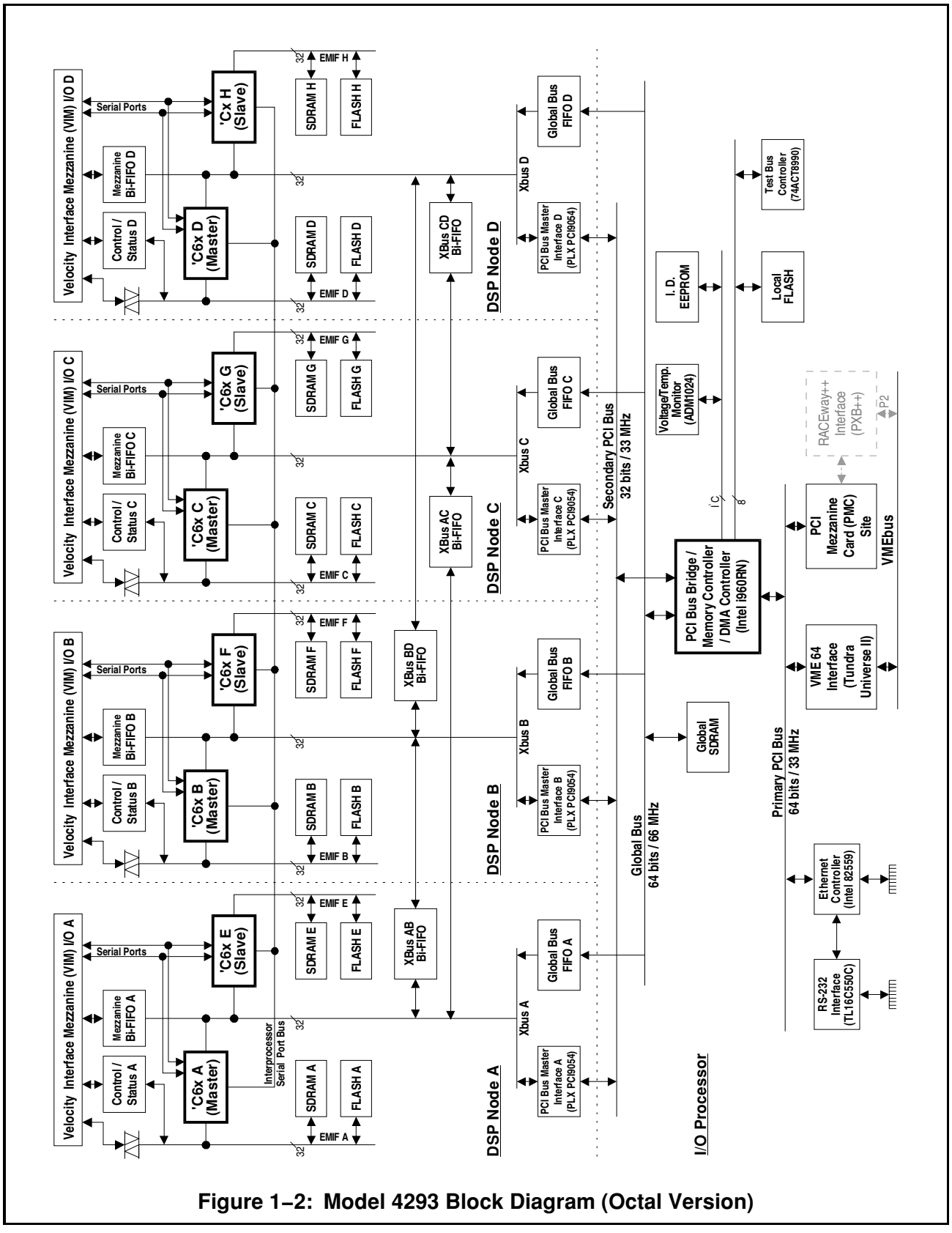


Figure 1–2: Model 4293 Block Diagram (Octal Version)

1.2 Digital Signal Processing Nodes (continued)

1.2.2 External Memory (EMIF) Interface

Using its External Memory Interface (EMIF) bus, each 'C6x processor has memory−mapped access to two types of local memories, an 8−MByte SDRAM (16−MByte SDRAM with Option 340) operating with single wait state access at transfer rates up to 600 MBytes/second, and a nonvolatile 1−MByte FLASH memory for initialization, self−test, and boot code.

Control and status registers for the DSP node and for the VIM site are also accessible from the Master 'C6x processor on its EMIF bus.

1.2.3 Serial Port Interfaces

Two of the three Serial Ports from each 'C6x processor, Serial Port 0 and Serial Port 1, are connected to the VIM interface associated with each DSP node. The remaining Serial Ports from each processor, Serial Port 2, are all connected to one another, providing an interprocessor serial communication bus.

1.2.4 Velocity Interface Mezzanine (VIM) Interfaces

The Model 4292/4293 PCB provides baseboard connectors for Pentek Velocity Interface Mezzanine (VIM) modules, either one or two VIM−2 modules, or one VIM−4 module. Each DSP Node has its own VIM connection (A, B, C, and D), providing the following three interfaces.

- A high−speed Mezzanine Bi−FIFO buffers 32−bit parallel data transfers between the VIM site and the node's Expansion Bus (Section 1.2.1), using the Bi−FIFO Streaming Data portion of the VIM interface.

- The Master 'C6x's External Memory Interface (EMIF) bus (Section 1.2.2) provides memory−mapped access to control and status registers for the VIM site, using the Random Access portion of the VIM interface.

- Two of each 'C6x's serial ports (Section 1.2.3) are brought to the VIM site, using the Serial Port portion of the VIM interface.

For detailed descriptions of the VIM interface signals, please refer to Pentek's VIM Specification, Pentek part number 800.00002.

The VIM architecture allows a variety of application−specific front panel con−nections including both analog and digital signal interfaces. VIM board func−tions include multichannel digital receivers and transmitters, data acquisition subsystems, multichannel codecs, high−speed A/D and D/A converters, high−speed parallel digital I/O, and custom boards for special applications. Refer to the Operating Manual supplied with each VIM module for further information.

1.3 I/O Processor

The Model 4292/4293 supports two PCI bus structures, the Primary PCI Bus and the Secondary PCI Bus. An Intel® i960® RN I/O Processor acts as the bridge between the two PCI buses, and acts as data executive for the four DSP nodes.

The Secondary PCI Bus provides the i960 access to each DSP node's XBus through four PCI Bus Master interfaces (PLX Technology PCI 9054). The Primary PCI Bus provides the i960 access to the RS−232 and Ethernet external interfaces (Intel 82559 and Texas Instruments TL16C550C), the PMC site, and the board's master/slave VMEbus interface (Tundra® Universe II™). Using the i960 PCI bus bridge, each 'C6x processor can master the Primary PCI Bus for read/write access to the VMEbus and the PMC site.

In addition to the two PCI buses, the i960 I/O Processor supports a Global Bus and two local buses. Resources available on the 64−bit synchronous Global Bus include a 4M x 64 bank of Global SDRAM, and four 2k x 32 Global Bus FIFOs, one connected to each DSP node's XBus. The i960's 8−bit asynchronous bus supports a 4M x 8 bank of FLASH memory and a JTAG Test Bus Controller (Texas Instruments SN74ACT8990). A voltage and temperature monitor (Analog Devices ADM1024) and a board ID EEPROM are accessible to the i960 on its local I^2C bus.

1.4 Global Resources

The Model 4292/4293 Global Bus provides DSP access to four Global Bus FIFOs, a Global SDRAM, the VMEbus Master/Slave interface, the PCI Mezzanine Card (PMC) site, the JTAG interface, and the (optional) RACEway interface. The Global Bus may be mastered by the i960, any Master 'C6x processor, or an external VMEbus Master.

1.4.1 Global Bus FIFOs

A 2k x 32 Global Bus FIFO on each node's XBus allows each Master 'C6x processor to receive high−speed data from global resources, including VMEbus, the PMC site, and the optional RACEway interface. These Global Bus FIFOs decouple the 'C6x processors from these resources by eliminating wait cycles and allowing 300 MBytes/second block transfers to the DSPs.

1.4.2 Global SDRAM

Serving as a convenient shared resource for staging data, a 32−MByte Global SDRAM on the Global Bus is accessible from all Master 'C6x processors, the i960, an external VMEbus Master, the PMC site, and the optional RACEway interface.

1.4 Global Resources (continued)

1.4.3 VMEbus Interface

The Model 4292/4293's Master/Slave VME interface, the Tundra Universe II, allows any Master 'C6x processor or the i960 processor to act as a VMEbus Master for data transfers to or from any VMEbus device. It also provides VMEbus Slave access to global board resources from an external VMEbus Master.

1.4.4 PMC Site

The 64−bit, 33−MHz PCI Mezzanine Card (PMC) site accepts industry−standard PMCs for a wide variety of interface and communication functions. The PMC site is accessible from all Master 'C6x processors and from the VMEbus.

Note that most PMC modules (particularly ones with front panels) will occupy one of the spaces that could be used for a VIM module.

1.4.5 RS−232/Ethernet Interface

The RS−232/Ethernet controller, an Intel 82559, provides an input/output path for RS−232 and Ethernet LAN signals using a front panel connector. This controller uses a Texas Instruments TL16C550C Single UART to provide RS−232 conversion for this interface. The RS−232/Ethernet controller enables high−speed data transfers across the Primary PCI Bus, and is accessible only to the i960 processor. This interface can operate in either full duplex or half duplex mode.

1.4.6 Optional RACEway Interface (Option 005)

The Model 4292/4293 equipped with Pentek's Option 005 RACE++® interface delivers up to 267 MBytes/second data transfers with other RACEway−equipped VME boards. This RACEway® module occupies the PMC site, and is accessible from all Master 'C6x processors and from an external VMEbus Master.

Note that the RACEway PMC does not occupy a VIM module space and you can still install two VIM−2 modules or one VIM−4 module with Option 005 installed.

1.4 Global Resources (continued)

1.4.7 JTAG Test Bus Controller

The JTAG Test Bus Controller (TBC), a Texas Instruments SN74ACT8990, provides access to each 'C6x processor's JTAG/XDS emulator ports. The TBC is accessible in VME memory space, and can access any 'C6x processor, allowing emulation across the VMEbus. Pentek's SwiftNet communications interface (see Section 1.7) provides a path for seamless integration of third party emulation tools, such as Texas Instruments' Code Composer.

The TBC connections are illustrated in the figure below.

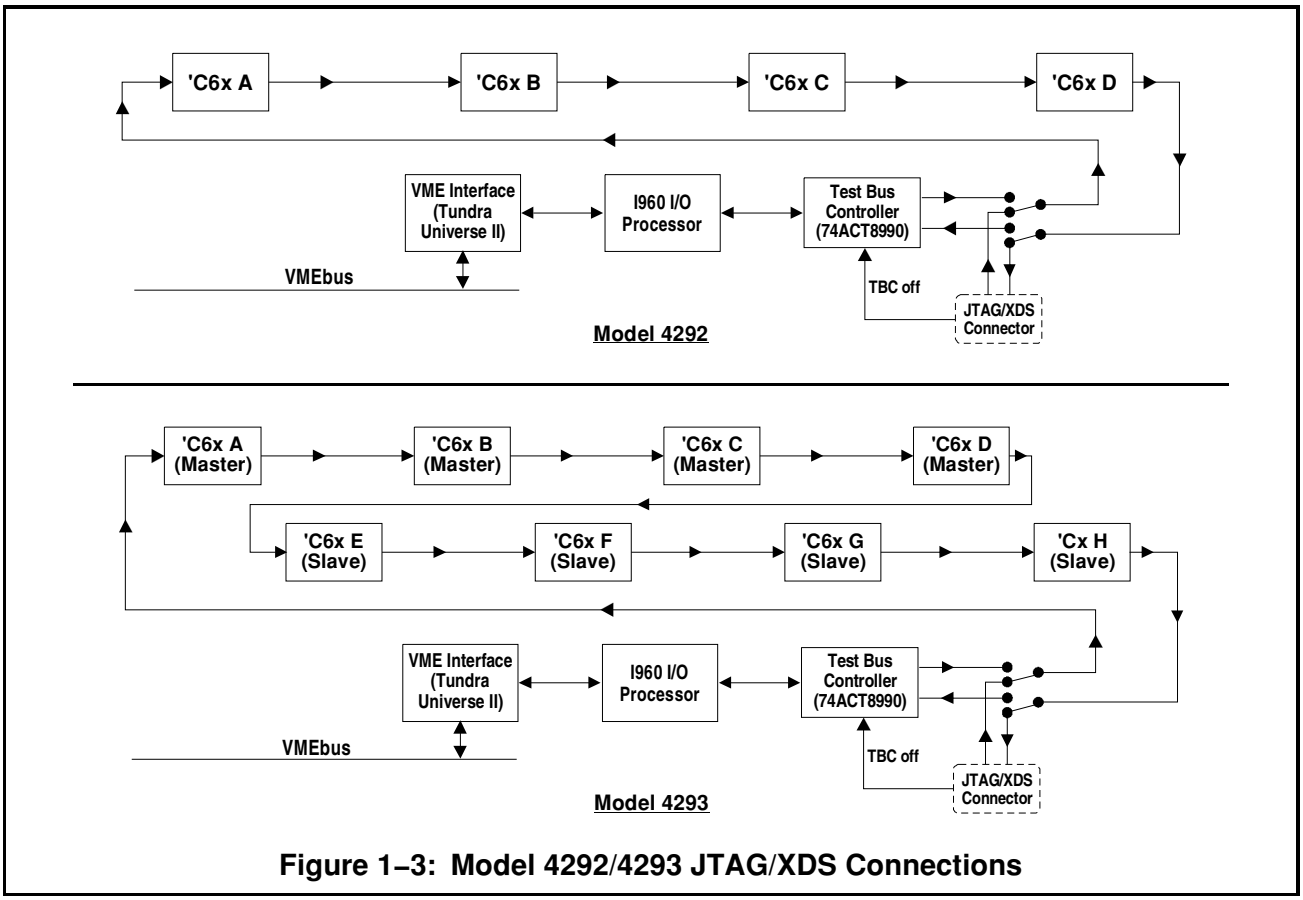


Figure 1–3: Model 4292/4293 JTAG/XDS Connections

Note: When a JTAG/XDS connector (Section 2.3.3) is inserted, the TBC is switched off.

1.5 Theory of Operation

Designed for applications requiring extremely high−speed signal processing and data transfers, the Model 4292/4293 features four (4292) or eight (4293) Texas Instruments 'C6x DSPs. Operating at up to 300 MHz, they can deliver a combined peak processing capability of 19,200 MIPS. The DMA controller transfers data to peripherals over dual 32−bit external data buses at rates as high as 600 MBytes/second.

Four identical DSP Nodes contain interface and memory resources matching the I/O capabilities of the 'C6x to high−performance system components. Each node includes local memory, a mezzanine (VIM) site interface, interprocessor links, a dedicated Global Bus I/O channel, and connections to shared board resources. Two types of local memory resources are provided for each 'C6x processor: an SDRAM, and a nonvolatile FLASH memory for initialization, self−test, and boot code.

On the Model 4293, each node contains two 'C6x's, a Master and a Slave, both equipped with local SDRAM and FLASH memory. The Master controls the VIM interface, interprocessor links, Global Bus I/O, and connections to shared resources. The Master moves data to/from the Slave's memory resources, and controls its execution through interrupts and flags. The Slave can be used to off−load tasks from the Master.

Each DSP Node has its own VIM (Velocity Interface Mezzanine) connector, providing three interfaces. A high−speed synchronous Bi−FIFO buffers 32−bit parallel data transfers between the VIM module and the Master 'C6x Expansion Bus at rates up to 300 MBytes/second. Two of each Master 'C6x's synchronous serial ports are brought to the VIM interface. The External Memory Interface (EMIF) bus of the Master 'C6x provides memory−mapped control and status information to the VIM interface.

Four interprocessor Bi−FIFOs join the Expansion Buses of each adjacent DSP Node in a ring structure. Each Master 'C6x can send and receive interprocessor data blocks at 300 MBytes/second without having to arbitrate for a shared resource.

Using a dedicated Global Bus FIFO, each Master 'C6x processor can receive blocks of data from global resources including the Global SDRAM, the VMEbus, the PMC site, and (with Option 005) RACEway. These FIFOs decouple the 'C6xs from these global resources by eliminating wait cycles and allowing 300 MBytes/second block transfers.

Serving as a shared resource for staging data, the Global SDRAM is accessible from each Master 'C6x processor, the VMEbus, the PMC site, and (optional) RACEway.

Each Master 'C6x processor can master the Primary PCI Bus for read/write access to the VMEbus, the Global SDRAM, the PMC site, and (optional) RACEway.

Using the Universe II interface, each Master 'C6x processor can master the VMEbus for data transfers to or from any VMEbus device. As a VMEbus Slave, Model 4292/4293 presents the Global SDRAM, the four Global Bus FIFOs, and the PMC site as memory−mapped resources to an external VMEbus Master.

The PCI Mezzanine Card (PMC) site accepts industry−standard modules for a wide variety of functions. The PMC site is accessible from each Master 'C6x processor and from the VMEbus. Delivering up to 267 MBytes/second between VME boards, the optional RACE++ interface occupies the PMC site, and allows each 'C6x to send and receive RACEway packets using a Global Bus FIFO as data buffer.

1.6 Board Support Software

Pentek's ReadyFlow® Board Support Libraries for the Model 4292/4293 contain a set of C−language routines for the Model 4292/4293 baseboard. These routines provide a complete library of hardware initialization, control, and application functions for the Model 4292/4293 baseboard. Refer to the Model 4999, Option 292, ReadyFlow Board Support Libraries Documentation (Pentek part #801.42920) for complete description of these capabilities.

ReadyFlow speeds application development by providing a high−level API for access to all of the board's memory and communication resources, and control of the board's I/O interfaces and I/O modules. Some general board functions include:

- Access to Master DSP memory−mapped registers,
- Reading and writing to FIFOs,
- Loading FLASH memory,
- Programming XBus DMA reads and writes,
- Interprocessor synchronization,
- Master/Slave DSP operations (Model 4293 only), and
- Control of the optional RACE++ interface.

The ReadyFlow routines also provide high and low level functions for the Command Processor application (see Chapter 12), and provides an interface to Pentek's SwiftNet® Software (see Section 1.7, below).

Model 4999 ReadyFlow® Board Support Libraries are also available for each Pentek VIM and PMC module that can be used on the Model 4292/4293. These routines pro−vide C−language functions to control and operate I/O and module resources.

The ReadyFlow libraries provide complete access to all hardware, provide develop−ment tools for quick start−up through application completion, allow programming at high, intermediate, and low levels to meet various needs, and include complete source code and documentation for all functions.

1.7 SwiftNet Software

The Model 4292/4293 can be used as a Target DSP with Pentek's SwiftNet® Software. SwiftNet software serves as a communications interface between an application devel−opment system and Pentek's VMEbus DSP boards. SwiftNet provides a simple, inter−rupt driven, inter−device communication protocol for distributed DSP applications. Where applicable, this manual identifies the Model 4292/4293 features that are used with SwiftNet software.

1.8 Supporting Documentation

In addition to the operating instructions provided in this manual, copies of relevant user documentation for the programmable devices on the Model 4292/4293 are available by separate request the respective companies. Refer to each company's website for the latest versions.

- ❑ Texas Instruments Incorporated — http://www.ti.com
  - Texas Instruments TMS320C6000 Peripherals Reference Guide, Number: SPRU190D
  - Texas Instruments TMS320C6000 CPU and Instruction Set Reference Guide, Number: SPRU189F
- ❑ Intel Corporation — http://www.intel.com
  - Intel i960® RM/RN I/O Processor Developer's Manual, Number: 273158–001
  - Intel 82559 Fast Ethernet Multifunction PCI/CardBus Controller Datasheet, Number: 743892–002, Revision 2.0
- ❑ PLX Technology, Inc. — http://www.plxtech.com
  - PLX PCI 9054 Data Book, Version 2.1, Number: 9054–SIL–DB–P1–2.1
- ❑ Tundra Semiconductor Corporation — http://www.tundra.com
  - Tundra® Universe II™ User Manual, Document: 80A3010_MA001_01 (A reference list of the Universe II registers is provided in Appendix E.)
- ❑ Analog Devices, Inc. — http://www.analog.com
  - Analog Devices ADM1024 Datasheet, C00059–0–2/01, Rev. A

NOTE: All references in this Operating Manual to the user documents listed above refer to the document versions identified above. If you use different versions of these documents, these references may not be the same.

1.9 Specifications

Node Processor

Model 4292:	Four, one for each DSP node
Type and Speed:	Option 002: Texas Instruments TMS320C6202
	Option 304: 200 MHz
	Option 306: 250 MHz
	Option 003: Texas Instruments TMS320C6203
	Option 308: 300 MHz
Model 4293:	Eight, two for each DSP node
Type and Speed:	Texas Instruments TMS320C6203, 300 MHz

DSP Local SDRAM:	Four (4292) or eight (4293), one for each Processor
Size:	Standard: 8 MBytes (2M x 32)
	Option 340: 16 MBytes (4M x 32)
Access:	Associated 'C6x processor on its EMIF Bus
Speed:	See Appendix C

DSP FLASH Memory:	Four (4292) or eight (4293), one for each Processor
Size:	1 MByte (1M x 8)
Access:	Associated 'C6x processor on its EMIF Bus

XBus Interprocessor Bi–FIFO:	Four: XBus AB, XBus AC, XBus BD, XBus CD
Size:	Standard: 8 kBytes (two 1k x 32 buffers)
	Option 320: 128 kBytes (two 16k x 32 buffers)
Access:	Each 'C6x has access to two XBus I/P Bi–FIFOs
Speed:	See Appendix C

Mezzanine (VIM) Bi–FIFO:	Four, one for each DSP node
Size:	Standard: 8 kBytes (two 1k x 32 buffers)
	Option 320: 128 kBytes (two 16k x 32 buffers)
Access:	Associated 'C6x processor on its XBus
Speed:	See Appendix C

Global Bus FIFO:	Four, one for each DSP node
Size:	Standard: 8 kBytes
	Option 320: 128 kBytes
Access (read only):	Associated 'C6x processor on its XBus
Speed:	See Appendix C

VIM EMIF Interface:	Four, one VIM connector for each DSP node
Access:	Associated 'C6x processor on its EMIF Bus
Compliance:	Pentek's VIM Specification, Pentek part # 800.00002

PCI Bus Master Interface:	Four, one for each DSP node
Controller:	PLX PCI 9054 I/O processor
Access:	Local Bus: Associated 'C6x processor on its XBus
	Secondary PCI Bus: i960, VMEbus, PMC site
Compliance:	32–bit PCI bus

1.9 Specifications (continued)

I/O Processor

Type:	Intel i960 RN I/O Processor
Speed:	33.3 MHz

FLASH Memory

Size:	4 MBytes (4M x 8)
Access:	i960 processor on its 8−bit asynchronous bus

ID EEPROM

Size:	1 MByte (1M x 8)
Access:	i960 processor on its I^2C bus

Voltage/Temperature Monitor

Controller:	Analog Devices ADM1024
Access:	i960 processor on its I^2C bus

Test Bus Controller

Controller:	Texas Instruments SN74ACT8990 Scan Path IC
Access:	i960 processor on its 8−bit asynchronous bus
JTAG Compliance:	IEEE Standard 1149.1 (JTAG)
XDS510 Interface:	14−pin PCB connector

Communication Ports

Controller:	Intel 82559 Ethernet PCI/CardBus Controller, with TI TL16C550C Single UART
Access:	i960 processor on the Primary PCI Bus
Ethernet Data Rate:	10/100 Mbps
RS−232 Data Rate:	9600, 19200, 38400, 57600, or 115200 Baud
External Connection:	Front panel 15−pin Micro−D connector

Global Bus

Access:	Master 'C6x processors, i960 processor, VMEbus
Data Width:	64 bits (32−bit mode selectable by PCB jumper)
Data Rate:	66 MHz

Global SDRAM

Size:	32 MBytes (4M x 64)
Access:	Global Bus: i960 processor PCI Bus: All 'C6x processors, VMEbus, PMC site
Arbitration:	Semaphore Registers for 'C6x and i960 processors
Speed:	See Appendix C

Global Bus FIFOs: Four, one per DSP node (see DSP Node Resources)

Size/Speed:	(Same as under DSP Node Resources)
Access (write only):	Global Bus: i960 processor PCI Bus (via i960 bus bridge): All 'C6xs, VMEbus, PMC
Arbitration:	Yes, Global Bus, Rotating Priority

1.9 Specifications (continued)

Primary PCI Bus

Access:	Master 'C6x processors, i960 processor, VMEbus, PMC Site
Data Width:	64 bits
Data Rate:	33 MHz

Secondary PCI Bus

Access:	Master 'C6x processors, i960 processor, VMEbus
Data Width:	32 bits
Data Rate:	33 MHz

VMEbus Interface

Controller:	Tundra Universe II
Master Access:	Master 'C6x processors, i960 processor: A32/D64, A32/D32, A24/D32, A24/D16, A16/D32, A16/D16
Slave Access:	External VMEbus Master: A32/D64, A32/D32, A24/D32, A24/D16, A16/D32, A16/D16
Compliance:	VME64, A32/D64

PMC Interface

Access:	Master 'C6x processors, i960 processor, VMEbus
Data Width:	64 bits
Data Rate:	33 MHz
Compliance:	PMC Standard IEEE P1386.1

RACEway Interface (Option 005 only) Occupies PMC Site

Access:	Master 'C6x processors, i960 processor, VMEbus
Data Width:	32 bits
Data Rate:	66 MHz
Compliance:	RACEway Interlink Standard ANSI/VITA 5.1–1999

Physical

Dimensions:	Standard 6U VMEbus board
Depth:	160.0 mm (6.3 in)
Height:	233.5 mm (9.2 in)
Panel Width:	20.3 mm (0.8 in)
Weight:	470 grams (1.04 lbs)

Power All quantities are typical, executing code after boot

Model 4292:	
TMS320C6202, 200 MHz	4.5 A @ +5.0 VDC
TMS320C6202, 250 MHz	5.0 A @ +5.0 VDC
TMS320C6203, 300 MHz	3.9 A @ +5.0 VDC
Model 4293:	4.7 A @ +5.0 VDC

Environmental

Operating Temperature:	0° to 50° C, measured at VME card cage exhaust port
Storage Temperature:	–20° to 90° C
Relative Humidity:	95% non–condensing
VME Rack Exhaust Temp:	0° to 50° C

This page is intentionally blank

Chapter 2: Installation and Connections

2.1 Inspection

After unpacking the Model 4292/4293, inspect it carefully for damage to connectors or components. If any damage is discovered, please contact Pentek immediately at (201) 818–5900. Please save the original shipping container and packing material, in case return shipment is required.

2.2 Jumper Settings

The following subsections cover operating parameters that you can set by shorting jumpers on the Model 4292/4293 PCB. Do not change jumpers that are not described in the following sections—they are reserved for factory test and setup purposes only.

The term jumper (or jumper block) refers to a group of two or more pins on a circuit board that may be connected in pairs by a shorting jumper to set or change an operating characteristic of the board. A pair of pins connected in this manner are referred to as installed, or shorted. A pair of pins that are not connected are called removed, or open. The shorting jumpers used on the Model 4292/4293 PCB are for 0.020" (0.51 mm) square pins spaced on 0.079" (2.00 mm) centers. These jumpers are DuPont™ part number 86730–001, or equivalent. Pentek's part number for these jumpers is 356.00010.

As shipped from the factory, several jumpers are installed in default positions on your board. The default operating parameters they select (for example, the VMEbus address of the board) may or may not meet your requirements. Before installing your Model 4292/4293 into the VMEbus card cage, please review the following subsections to determine whether you need to change any of these settings.

Figure 2–1, on the following page, shows the locations of jumpers that may be changed for your application requirements.



1) Do not change any jumpers that are not identified in this figure—they are reserved for factory test and setup purposes only.

2) PCB jumper **J13** pins are permanently shorted (soldered) at the factory. If you have a Model 4292/4293 PCB with a removable jumper at **J13**, this jumper **must** be installed for proper firmware operation.

2.2 Jumper Settings (continued)

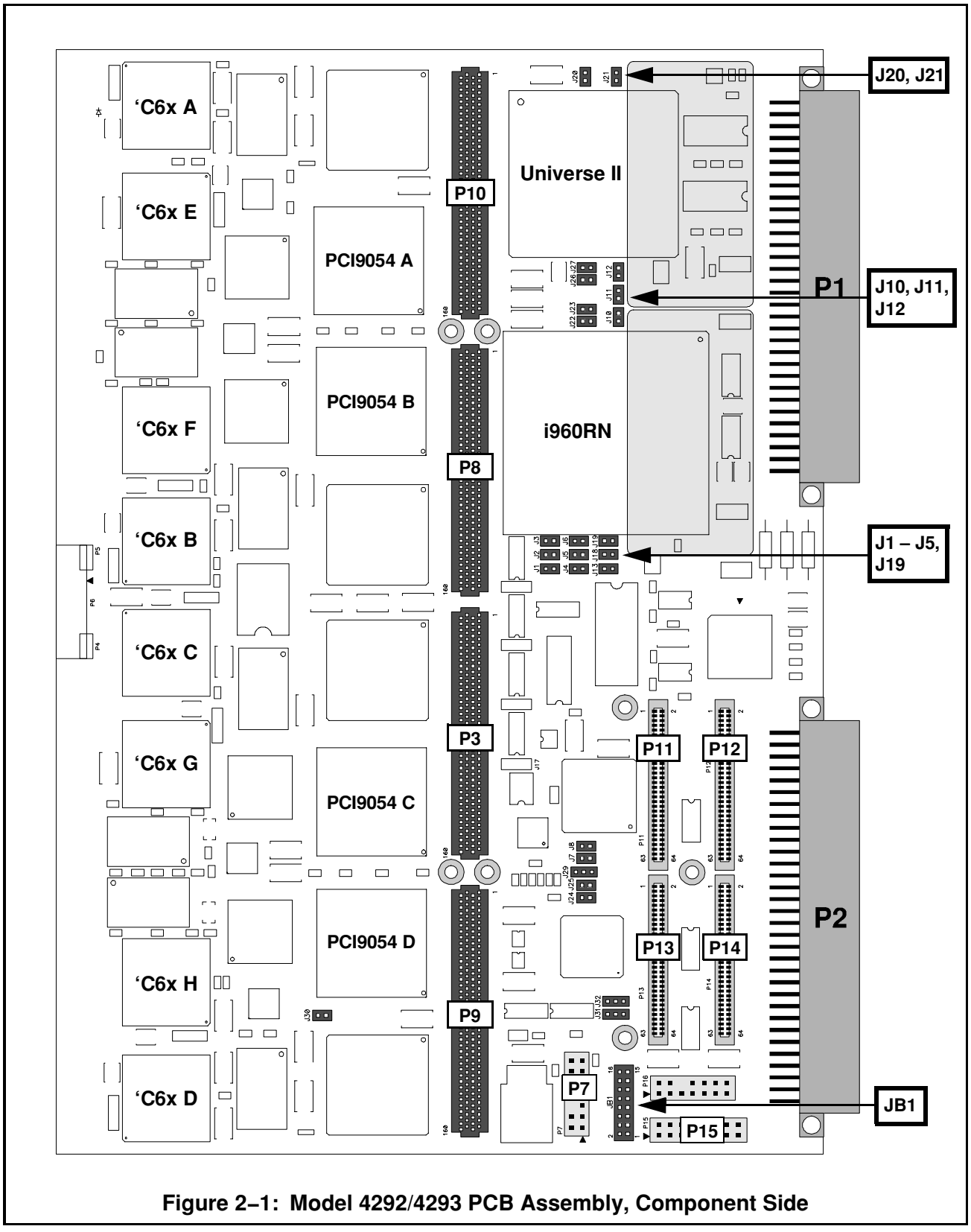


Figure 2–1: Model 4292/4293 PCB Assembly, Component Side

2.2 Jumper Settings (continued)

2.2.1 Board Power–Up Mode Jumpers—J1 & J2

At power–up of the Model 4292/4293 board, two jumpers, **J1** and **J2**, determine the diagnostic self–test operation and the state of the i960 and DSP FLASH memories. The following table shows the jumper settings for the four power–up modes that can be selected.

Table 2–1: Jumpers J1 & J2 – Board Power–Up Mode

J1 State	J2 State	Power–Up Mode
Removed*	Removed*	No FLASH erased, No diagnostics run
	Installed	Erase i960 FLASH User Blocks
Installed	Removed	Erase DSPs FLASH User Blocks
	Installed	Run factory diagnostics, No FLASH erased
* Factory Default Settings		



If you select either 'Erase i960/DSPs FLASH User Blocks' mode, you must also enable Write for the appropriate FLASH, i960 (Section 2.2.5) or DSP (Section 2.2.3). Also, if you select the 'Erase DSPs FLASH...' mode, the board erases the user blocks of all four DSP FLASH memories.

The user blocks of each FLASH memory include all blocks except the boot code blocks. See the following two sections (Sections 2.2.5 and 2.2.3) for identification of the boot blocks in each type of FLASH memory.

2.2.2 Endian Mode Jumper—J3

The 'C6x processors can be configured to operate in either Big or Little Endian mode using jumper **J3,** allowing the Model 4292/4293 to easily interface with different host processors. This jumper setting applies to all 'C6x processors on the board. The following table shows the jumper settings and explains the two Endian (byte ordering) mode settings.

Table 2–2: Jumper J3 – Endian Mode

J3 State	Endian Mode
Removed	Big Endian Mode – the most significant byte of a given word is located in the lowest byte address occupied by the word
Installed*	Little Endian Mode – the most significant byte of a given word is located in the highest byte address occupied by the word
* Factory Default Setting	

2.2 Jumper Settings (continued)

2.2.3 DSP FLASH Memory Write Enable Jumpers—J4 & J5

Each DSP FLASH memory is organized as separate boot, parameter, and user blocks. The boot blocks are the first two logical blocks of FLASH, 8 kBytes each. Each DSP FLASH is shipped with a default 'C6x boot program installed by Pentek (see Section 10.2). Two PCB jumpers are provided to enable writing to all four DSP FLASH memories. Jumper **J5** provides V_{PP} (program/erase voltage)—this jumper must be installed to enable writing to any block on any FLASH. Jumper **J4** write protects the two boot blocks on all memories (see **NOTE** below). The following table shows these jumper settings.

Table 2–3: Jumpers J4 & J5 – DSP FLASH Memory Write Enable

J5 State	J4 State	FLASH Write Enable
Installed	Removed *	User and Parameter blocks write enabled, Boot blocks write protected
	Installed	All blocks write enabled
Removed *	Do Not Care	All blocks write protected
* Factory Default Settings		



Pentek default boot code is stored in the first 64 kBytes of each DSP FLASH, in both boot and parameter blocks. When you enable writing to the user blocks, the parameter blocks are also write–enabled.
Users who tamper with the Pentek boot code do so at their own risk.

2.2 Jumper Settings (continued)

2.2.4 i960 Initialization Mode Jumpers—J10, J11, & J12

At power–up or reset of the board, three jumpers select the i960 I/O processor initialization modes: **J10** selects the memory/bus data width (and the Global SDRAM size), **J11** selects the retry mode of Primary PCI bus PCI configuration cycles, and **J12** selects the i960 processor Reset mode. Refer to Intel's i960 RM/RN I/O Processor Developer's Manual for additional information on these i960 initialization modes. The following table shows these jumper settings.

Table 2–4: Jumpers J10, J11, & J12 – i960 Initialization Mode

Jumper	Jumper State	i960 Initialization Mode
J10	Removed*	64–bit Memory/Data Bus (Global Bus), Global SDRAM is 32 MBytes
	Installed	32–bit Memory/Data Bus (Global Bus), Global SDRAM is 16 MBytes
J11	Removed*	Retries All Configuration Transactions
	Installed	Accepts PCI Transactions
J12	Removed*	Normal Initialization
	Installed	Processor Held in Reset
* Factory Default Settings		

NOTE: When jumper **J10** is set for 64–bit mode, the Global Bus FIFOs are not available in the default memory maps.

2.2.5 i960 FLASH Memory Write Enable Jumper—J19

The i960 FLASH memory is organized as separate boot and user blocks. The boot block is the last 64 kBytes of FLASH. The i960 FLASH is shipped with a default boot program installed by Pentek (see Section 10.3). This Pentek boot code is protected using write–protect lock bits. Jumper **J19** enables writing to or erasing any block except the boot code blocks. The following table shows the i960 FLASH Memory Write Enable jumper settings.

Table 2–5: Jumper J19 – i960 FLASH Memory Write Enable

J19 State	FLASH Write Enable
Removed	All blocks write protected
Installed*	All User blocks (except Boot) write enabled
* Factory Default Setting	

2.2 Jumper Settings (continued)

2.2.6 VMEbus A16/A24 Slave Select Jumper—J20

PCB jumper **J20** configures the Model 4292/4293 to respond as a VMEbus Slave in either A16 or A24 address space for accessing the Universe II VME–bus Configuration Registers. When jumper **J20** is removed (the factory default setting), the board will respond as a Slave in A16 space only; when jumper **J20** is installed, the board will respond as a Slave in A24 space. In either of these address modes, the board occupies 4 kBytes of VME space for the default configuration. The following table shows the jumper settings. See Section 2.2.8 on the next page to set the A16/A24 slave base address.

Table 2–6: Jumper J20 – VMEbus A16/A24 Slave Select

J20 State	VMEbus Slave Function
Removed *	A16 Address Space Enabled
Installed	A24 Address Space Enabled
* Factory Default Setting	

The Model 4292/4293 is shipped from the factory with VMEbus A32 space slave access disabled. The user can enable and configure the board's A32 slave access either from the i960 serial port or from VME A16/A24 space. Refer to Section 7.3 for more information on VMEbus Slave access.

2.2.7 VMEbus System Reset Monitor Jumper—J21

Jumper **J21** enables the Model 4292/4293 to respond to or ignore a VME System Reset. With this jumper removed (the factory default setting), the 4292/4293 responds to a VME System Reset issued by a VMEbus system controller. This is typically the mode needed when the board is NOT a Slot 1 Controller. With this jumper installed, the 4292/4293 ignores a VME System Reset. This is typically the mode needed as Slot 1 Controller. The following table shows the jumper settings.

Table 2–7: Jumper J21 – System Reset Monitor

J21 State	System Reset Function
Removed *	4292/4293 responds to System Reset
Installed	4292/4293 ignores System Reset
* Factory Default Setting	

NOTE: Install this jumper when the Model 4292/4293 is installed in slot 1 as the system controller and remove this jumper all other times.

2.2 Jumper Settings (continued)

2.2.8 VMEbus Slave Base Address Jumpers—JB1

Jumper block **JB1** sets the Model 4292/4293 VMEbus Slave Base Address (the lowest address that this board will respond to as a Slave), in VMEbus A16 and A24 address spaces. Use Jumper **J20** (Section 2.2.6, prior page) to select which slave address mode the board will respond to, A16 or A24.

Jumper pins 15 and 16 of jumper block **JB1** correspond to the most significant address bit (A23) in A24 space, and to the most significant address bit (A15) in A16 space. The next two lower–numbered jumper pins correspond to the next least significant address bit, such that pins 9 and 10 correspond to address bit A12 in A16 space, and pins 1 and 2 correspond to address bit A16 in A24 space. Shorting a pair of jumper pins (jumper installed) clears the corresponding address bit to '0', and an open pair of jumper pins sets the corresponding address bit to '1'. Table 2–8 below shows the corresponding VMEbus address bit for each pair of jumper block pins, and gives the hexadecimal address weight of each jumper pair, in both address spaces.

Table 2–8: Jumper Block JB1 – VMEbus A16/A24 Slave Base Address

Jumper Pins	VMEbus Address Bit		A16 Hex Weight	A24 Hex Weight
	A16	A24		
1 – 2	–	A16	–	**0x01 0000**
3 – 4	–	A17	–	**0x02 0000**
5 – 6	–	A18	–	**0x04 0000**
7 – 8	–	A19	–	**0x08 0000**
9 – 10	A12	A20	**0x1000**	**0x10 0000**
11 – 12	A13	A21	**0x2000**	**0x20 0000**
13 – 14	A14	A22	**0x4000**	**0x40 0000**
15 – 16	A15	A23	**0x8000**	**0x80 0000**
Jumper installed, bit = 0 Jumper removed, bit = 1 Default, all jumpers installed: A16_base = **0x0000**, A24_base = **0x00 0000**				

NOTE: If you want an external VMEbus Master to access the board's A24 memory space in A24 address mode, set the A24 slave base address to either **0x00 0000** or **0x80 0000**.

The Model 4292/4293 is shipped with A16/A24 slave access enabled only to the Tundra Universe II registers, at the slave base address set by **JB1**. You can configure these registers from the board's serial port or from the VMEbus, as described in Section 7.3, to enable extended A24 or A32 slave access and set the slave base address for other board resources.

2.2 Jumper Settings (continued)

2.2.9 Factory Default Jumpers

The following jumpers are reserved for factory test and setup purposes only.

Table 2–9: Factory Default Jumpers

2–pin Jumpers			
Jumper	**Jumper State**	**Function**	
J6	Removed *	EEPROM Write Protect	
J7	Removed *	’C6x Boot Configuration	
J8	Installed *		
J13	Removed *	Board ID EEPROM Write Protect	
J22	Removed *	’C6x Boot Configuration	
J23	Installed *		
J24	Removed *		
J25	Installed *		
J26	Removed *		
J27	Installed *		
J30 **	Removed *	JTAG Boundary Scan	
3–pin Jumpers			
Jumper	**Jumper Pins**	**Jumper State**	**Function**
J29 **	**1 – 2**	Installed *	JTAG Boundary Scan
J31 **	**2 – 3**	Installed *	
J32 **	**2 – 3**	Installed *	

* Factory Default Settings

** These jumpers are available only on Model 4292/4293 boards labeled as REV C or greater on the PCB, part # 320.42930.



DO NOT change these jumper settings!

2.3 Signal Connectors

The following subsections describe the signal connectors on the Model 4292/4293 PCB. Refer to Figure 2–1 for location of these connectors on the component side of the board.

2.3.1 VIM Connectors

Four 160–pin connectors, designated **P10**, **P8**, **P3**, and **P9**, provide connections for Pentek Velocity Interface Mezzanine (VIM) modules. Each connector provides an identical VIM signal interface to each node's Master 'C6x processor. **P10** and **P8** comprise the upper module site (DSP Nodes A and B), and **P3** and **P9** comprise the lower module site (DSP Nodes C and D). The Model 4292/4293 can support one or two VIM–2 modules (in either site), or one VIM–4 module (which occupies both sites).

For description of each signal on this connector, please refer to Pentek's VIM Specification, Pentek part # 800.00002.

See Section 2.5.1 for VIM module installation instructions.

2.3.2 PMC Connectors

The PCI Mezzanine Card (PMC) site accepts industry–standard PMC modules for a wide variety of interface and communication functions (such as the RACEway interface module, Option 005). The PMC is connected to the Model 4292/4293 PCB using four 64–pin connectors, designated **P11** through **P14**. These connectors are defined by the PMC standard, as follows:

- **P11** and **P12** contain the 32–bit PCI bus interface, plus associated power, ground, reserved pins, and other necessary signals.
- **P13** supports expansion to the 64–bit PCI bus interface.
- **P14** supports user–defined inputs/outputs. The Model 4292/4293 routes I/O signals from the **P14** connector to the VMEbus backplane **P2** connector.

Refer to the IEEE P1386.1 PMC Standard for information concerning the PCI Mezzanine Card standards and pin definitions.

See Section 2.5.1 for PMC installation instructions.

NOTE: Generally, if you install a PMC on the Model 4292/4293, you can install only one VIM–2 module, on the upper VIM site. However, if you install the Option 005 (RACEway) PMC, you can still install two VIM–2 modules or a VIM–4 module on the board.

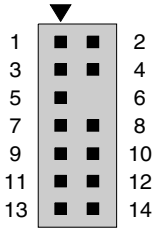
2.3 Signal Connectors (continued)

2.3.3 JTAG/XDS Connector

A JTAG/XDS connector, designated **P7** on the PCB (see Figure 2–1), provides a user connection to download programs to the 'C6x processors and to control the execution of those programs during software development using Texas Instruments' XDS510 and JTAG emulation protocols. When an XDS510 emulator is connected to **P7**, the Test Bus Controller (Section 1.4.7) is switched off. The pinout for this 14–pin connector is given in the following table.

Table 2–10: JTAG/XDS Connector (P7)

Signal	Pin Number		Pin Number	Signal
TMS	1		2	$\overline{\text{TRST}}$
TDI	3		4	GND
V_{CC}	5		6	(KEY – no pin)
TDO	7		8	GND
TCK_RET	9		10	GND
TCK	11		12	GND
EMU0	13		14	EMU1





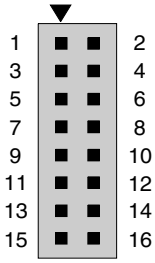
Some emulators, such as the Spectrum Digital 510 series, do not have pin–keyed connectors—thus, be sure to orient its cable connector so that its pin 1 matches pin 1 of PCB connector **P7**.

2.3.4 i960 Emulator Connector

An i960 Emulator connector, designated **P15** on the PCB (see Figure 2–1), provides a connection to download programs to the i960 processor and control their execution. **This connector is reserved for Pentek factory use only.** The pinout for this 16–pin connector is given in the following table.

Table 2–11: i960 Emulator Connector (P15)

Signal	Pin Number		Pin Number	Signal
$\overline{\text{TRST}}$	1		2	GND
TDI	3		4	GND
TDO	5		6	GND
TMS	7		8	GND
TCK	9		10	GND
$\overline{\text{LCDINIT}}$	11		12	GND
$\overline{\text{I_RST}}$	13		14	GND
PWRVLD	15		16	GND



2.4 DSP Timer Test Points

The solder side of the Model 4292/4293 PCB contains several test points that can be accessed by the user. Table 2–12, below, lists the 'C6x processor timer test points available on the PCB. Figure 2–2, on the next page, shows the locations of these test points on the solder side of the PCB (near the front edge of the board).

Table 2–12: 'C6x Timer Test Points

DSP	Test Point	'C6x Signal Pin
'C6x A	TP22	TOUT0 (Timer 0 Out)
	TP23	TOUT1 (Timer 1 Out)
	TP24	TINP0 (Timer 0 In)
	TP25	TINP1 (Timer 1 In)
'C6x B	TP8	TOUT0 (Timer 0 Out)
	TP9	TOUT1 (Timer 1 Out)
	TP10	TINP0 (Timer 0 In)
	TP11	TINP1 (Timer 1 In)
'C6x C	TP1	TOUT0 (Timer 0 Out)
	TP2	TOUT1 (Timer 1 Out)
	TP3	TINP0 (Timer 0 In)
	TP4	TINP1 (Timer 1 In)
'C6x D	TP15	TOUT0 (Timer 0 Out)
	TP16	TOUT1 (Timer 1 Out)
	TP17	TINP0 (Timer 0 In)
	TP18	TINP1 (Timer 1 In)

Note that the TOUT0 and TOUT1 signals from each 'C6x processor also drive two of that processor's LEDs on the front panel (see Section 2.6.5).

In addition to the above 'C6x processor test points, four Ground test points are provided on the PCB, near these DSP test points: **TP56**, **TP53**, **TP31**, and **TP48**. These are also identified on Figure 2–2, on the following page.



The user should not use test points that are not identified in this table or the PCB drawing on the following page—they are reserved for factory purposes only.

2.4 DSP Timer Test Points (continued)

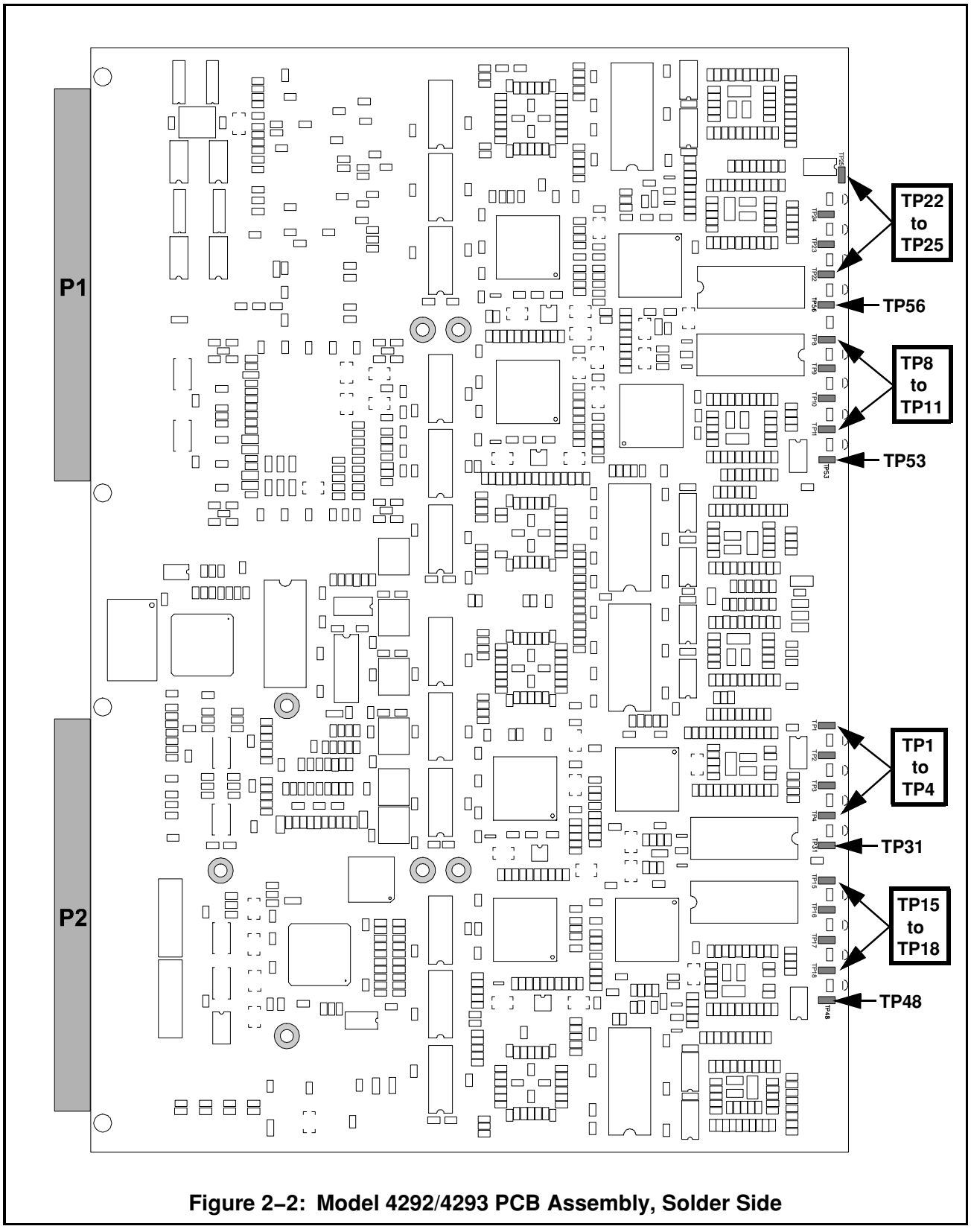


Figure 2–2: Model 4292/4293 PCB Assembly, Solder Side

2.5 Installation

CAUTION

TURN OFF ALL POWER TO THE VME CARD CAGE BEFORE INSERTING OR REMOVING ANY BOARD.

2.5.1 Installing VIM and PMC Modules

Before you install the Model 4292/4293 board into a VMEbus card cage, you must install any expansion VIM and/or PMC module(s) for your application.

NOTE: You can install one VIM−4, two VIM−2's, one VIM−2 plus one PMC, one VIM−2, or one PMC on the Model 4292/4293 baseboard. However, if you install the Option 005 (RACEway) PMC, you can still install two VIM−2 modules or one VIM−4 module on the board.

Each Pentek VIM and PMC module is supplied with an Operating Manual that provides setup and installation instructions for that module, such as jumper settings, installation onto the VIM baseboard, and front panel connections. Refer to the manual supplied with your module(s) for the proper installation procedures.

For installation of a PMC on Model 4292 (Quad) boards, please call Pentek for mounting information.

2.5.2 Installing the Model 4292/4293 in the VME Card Cage

After the necessary jumpers on the Model 4292/4293 have been set and the assembly of the board and expansion VIM and/or PMC modules (if any) has been completed, install the PCB assembly into the VMEbus card cage.

When inserting the Model 4292/4293, use only the front panel ejector handles of the board. After inserting the assembly into the cage card guides, push in on the top and bottom ejector handles to fully seat the connectors. Once the assembly is seated, screw the captive panel screws at the top and bottom of the front panel into the top and bottom rails of the card cage.

2.5.3 Removing the Model 4292/4293 from the VME Card Cage

When removing the Model 4292/4293 assembly, first loosen the captive screws on the top and the bottom of the front panel. Then, push both ejector handles of the board away from the center of the panel to eject the assembly from the cage. Once the assembly is disengaged, pull outward on the ejector handles to remove the assembly from the card cage guides.

2.6 Front Panel

The Model 4292/4293 front panel has three fixed plates at the top, center, and bottom, and two removable blank panels. The top fixed plate includes two LEDs labeled **Pwr** and **Temp**. The center fixed plate contains a connector for RS–232 and Ethernet communications, plus three LEDs labeled **Fault**, **Active**, and **Link**. The bottom fixed plate provides a push button labeled **Reset**. Four LEDs associated with the 'C6x processors, labeled **0**, **1**, **2**, and **3** on the blank panels, are visible to the left of each blank panel.

Before installing a Velocity Interface Mezzanine (VIM) or PCI Mezzanine Card (PMC) module, you must remove one or both blank panels, whose position(s) will be occupied by the module's front panel(s). Refer to the manual supplied with your module(s) for installation procedures.

The figure at the right depicts the Model 4292/4293 front panel as it appears with no VIM or PMC modules installed. The following subsections describe the user controls and indicators on the front panel.

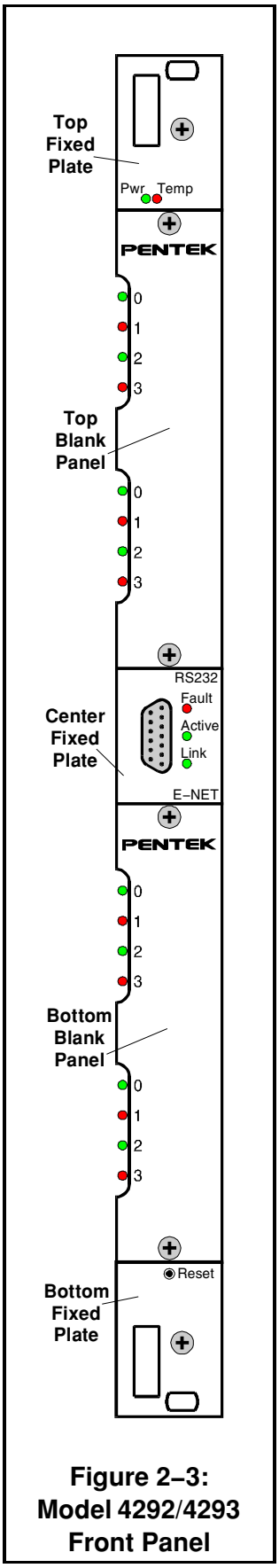


Figure 2–3: Model 4292/4293 Front Panel

2.6.1 Reset Button

Press the **Reset** push–button on the bottom fixed plate to start a board reset cycle that returns the Model 4292/4293 to its power–up state. Note that this button is recessed in the panel—you must use a small probe (paperclip, etc.) to depress it.

2.6.2 Power LED

At the lower left of the top fixed plate is a green LED labeled **Pwr** that illuminates whenever +5 V_{DC} power is applied to the Model 4292/4293.

2.6.3 Over–Temperature LED

At the lower center of the top fixed plate is a red LED labeled **Temp** that illuminates when the Model 4292/4293 Voltage/Temperature Monitor (ADM1024) detects an over–temperature condition on the PCB. The i960 I/O processor sets the sensor thresholds (see Voltage/Temperature Monitor, Section 5.4.9). In addition to this LED, an over–temperature interrupt is asserted to the i960 (see Table 8–4, page 122).

2.6 Front Panel (continued)

2.6.4 I/O LEDs

On the fixed center plate, to the right of the RS–232/Ethernet connector, are three LEDs associated with the i960 processor and RS–232/Ethernet communications. The red **Fault** LED is illuminated by the i960 when it detects a fault or when the LED bit is set in the i960 Control Register (Section 5.5.6). The green **Active** LED is illuminated while an Ethernet data transfer is in progress. The green **Link** LED is illuminated when a good Ethernet connection is established.

2.6.5 'C6x Processor LEDs

To the left of the blank panels, on the edge of the PCB, are four groups of four LEDs associated with the 'C6x processors. The top group is for DSP Node A, the next is Node B, the next is Node C, and the bottom group is for Node D. In each group, the green LED labeled **0** is driven by the timer 0 output, TOUT0, and the red LED labeled **1** is driven by the timer 1 output, TOUT1, of the node's Master 'C6x. On the Model 4293, LEDs **2** and **3** of each group are driven by TOUT0 and TOUT1, respectively, of the node's Slave 'C6x.

These LEDs inform the user of the results of the board's power–up self–tests. If all tests for a given 'C6x processor pass, the associated processor's green LED (**0** or **2**) will blink at a rate of about five times per second. If one of the tests fails, that processor's red LED (**1** or **3**) will blink repetitively until power is removed or the **Reset** button is pressed (the **Reset** initialization cycle does not include the self–test). See Section 10.6 for the test failure LED blink codes.

2.6.6 RS–232/Ethernet Connector

The front panel connector on the fixed center plate provides input/output of both RS–232 and Ethernet serial signals. The following table gives the pinout of this 15–pin Micro–D plug. A mating cable is provided, Pentek part # 002.28150, which is described on the next page.

Table 2–13: RS–232/Ethernet Connector

Signal	Pin Number		Pin Number	Signal
E–net, TXN	9		1	Gnd
Gnd (75Ω)	10		2	RS–232, TX
Gnd (75Ω)	11		3	NC
E–net, RXP	12		4	RS–232, RX
E–net, RXN	13		5	NC
Gnd (75Ω)	14		6	Gnd
Gnd (75Ω)	15		7	Gnd
			8	E–net, TXP

2.6 Front Panel (continued)

2.6.6 RS–232/Ethernet Connector (continued)

The mating cable for the front panel RS–232/Ethernet connector, illustrated below, provides three breakout connectors: one Ethernet RJ45 connector and two RS–232 DB9 connectors (the second RS–232 connector is not used on the Model 4292/4293). The cable length from the RS–232/Ethernet connector to the RS–232 (1) DB9 connector is 18 inches; the cable length from the RS–232 (1) connector to each of the other two breakout connectors is 6 inches.

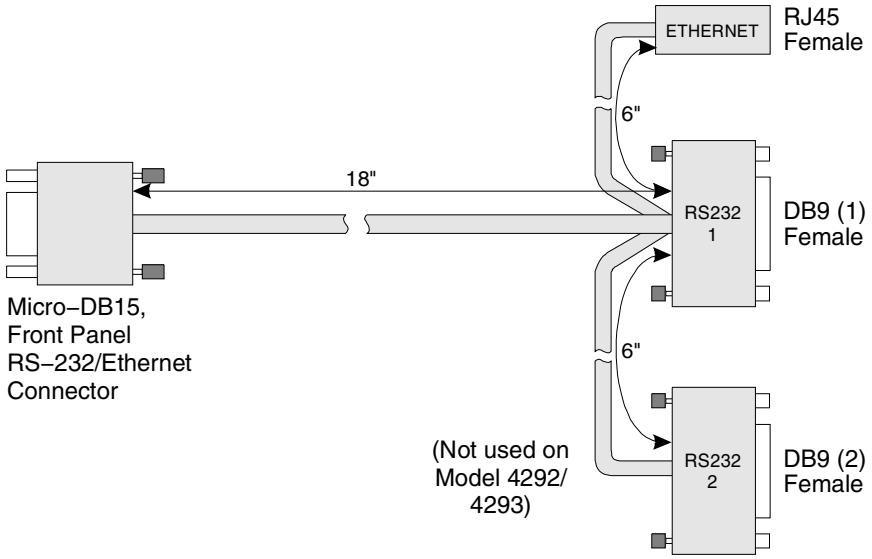


Figure 2–4: RS–232/Ethernet Cable
(Pentek Part #002.28150)

The following table shows the pin–to–pin signal wiring from the front panel RS–232/Ethernet Micro–DB15 connector to each breakout connector.

Table 2–14: RS–232/Ethernet Cable Pin Connections

Micro–DB15 Pin	Signal	RJ45 Pin	DB9 (1) Pin	DB9 (2) Pin
2	RS–232, TX	NC	2	Not used on Model 4292/ 4293
4	RS–232, RX		3	
6	Gnd		5	
8	E–net, TXP	3	NC	
9	E–net, TXN	6		
12	E–net, RXP	1		
13	E–net, RXN	2		

2.7 Board Reset

The following figure shows the reset logic for the Model 4292/4293 following an operator power on or reset of the board, or a VMEbus System Reset. The resets issued by 'C6x (Section 3.6.8) and i960 (Section 5.5.6) control registers are also shown.

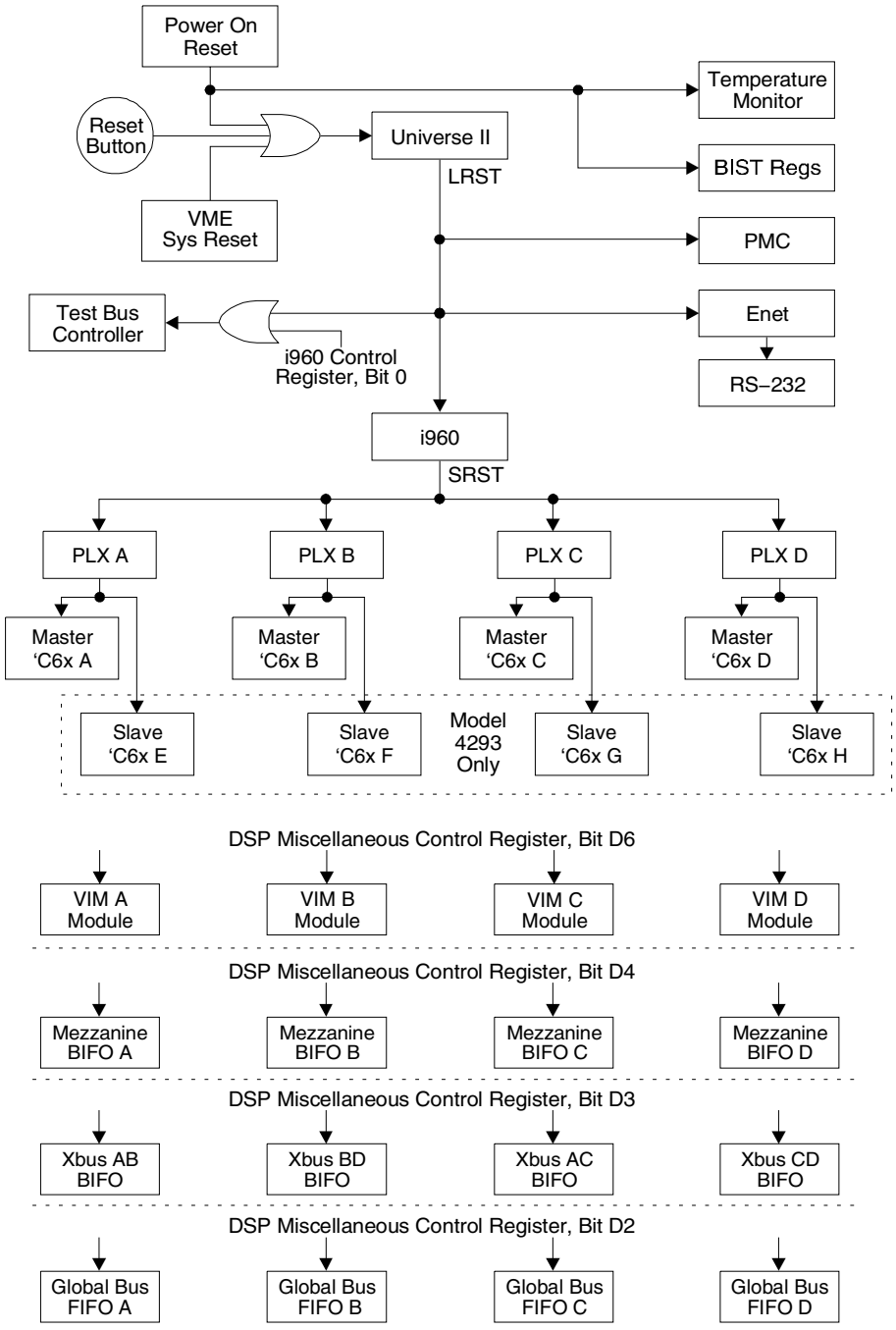


Figure 2–5: Model 4292/4293 Reset Logic

This page is intentionally blank

Chapter 3: Master 'C6x Processor Memory Maps and Resources

3.1 Overview

This chapter presents memory maps and operation of Model 4292/4293 resources available to each Master 'C6x processor. In addition to internal memory and peripherals, each Master 'C6x processor has direct access to its node's Expansion Bus (XBus) and External Memory Interface (EMIF) resources, plus access to other board resources outside the node using its XBus Host Port Interface. In addition to the resources identified in this chapter, refer to Chapter 8 for 'C6x interrupt operation, Chapter 9 for 'C6x FIFO operation, and Chapter 10 for 'C6x FLASH and boot firmware operation.

3.2 'C6x Processor Internal Resources

The 'C6x processor internal resources include program and data memories and on-chip peripherals. Descriptions of these internal resources are provided in Texas Instruments TMS320C6000 Peripherals Reference Guide. A memory map of these resources is provided in that manual for each of the processors used on the Model 4292/4293 (note that at Model 4292/4293 boot, MAP 0 is set as the first section of the memory map). See Table 3–1, on the next page, for the board's memory map of all internal and external/board resources available to each Master 'C6x.

The size of the internal program and data memories depends on the 'C6x processor type on your board, as follows:

Processor	Internal Program RAM	Internal Data RAM
TMS320C6202	256 kBytes	128 kBytes
TMS320C6203	384 kBytes	512 kBytes

Memory-mapped registers on the processor's internal peripheral bus configure the External Memory Interface (EMIF), DMA, Expansion Bus (XBus), Multichannel Buffered Serial Port (McBSP), Timers, Interrupt Selector, and Peripheral Power-Down.

- The EMIF, DMA, XBus, Interrupt Selector, and Peripheral Power-Down registers are initialized by the 'C6x boot sequence (Section 10.4.3). XBus Host Port registers allow an external XBus Host to read/write any address on the Master DSP memory map.

- For operation of the 'C6x Serial Ports (McBSP) and Timers, refer to TI TMS320C6000 Peripherals Reference Guide.

The 'C6x XBus Global Control Register (XBGC) configures XBus parameters. For a Master 'C6x, bit 11 (XARB) of this register is set to logic '1.' Refer to TI TMS320C6000 Peripherals Reference Guide for description of this register.

3.3 Master DSP Memory Map

Each Master 'C6x processor has memory–mapped access to its internal resources, DSP local SDRAM and FLASH memory, DSP registers, the VIM interface and Bi–FIFO, interprocessor Bi–FIFOs, and the Global Bus FIFO. The following table shows the resources accessible from each Master DSP, the path to each resource (internal, EMIF, or XBus), the memory addresses of that resource, and a reference to information on 'C6x operation of each resource.

Table 3–1: Master DSP Memory Map

Resource	Path	Memory Address	Information
DSP Local SDRAM	EMIF	0x0000 0000 – 0x00FF FFFF	Section 3.4
DSP Local FLASH Memory		0x0100 0000 – 0x013F FFFF	Section 10.2
Internal Program RAM C6202 Internal Program RAM C6203	Internal	0x0140 0000 – 0x0143 FFFF 0x0140 0000 – 0x0145 FFFF	Section 3.2 and TI TMS320C6000 Peripherals Reference Guide [1]
Reserved		0x0146 0000 – 0x017F FFFF	
Internal Peripheral Bus EMIF Registers		0x0180 0000 – 0x0183 FFFF	
Internal Peripheral Bus DMA Registers		0x0184 0000 – 0x0187 FFFF	
Internal Peripheral Bus XBus Registers		0x0188 0000 – 0x018B FFFF	
Internal Peripheral Bus McBSP 0 Registers		0x018C 0000 – 0x018F FFFF	
Internal Peripheral Bus McBSP 1 Registers		0x0190 0000 – 0x0193 FFFF	
Internal Peripheral Bus Timer 0 Registers		0x0194 0000 – 0x0197 FFFF	
Internal Peripheral Bus Timer 1 Registers		0x0198 0000 – 0x019B FFFF	
Internal Peripheral Bus Interrupt Registers		0x019C 0000 – 0x019C 01FF	
Internal Peripheral Bus Power–Down Register		0x019C 0200 – 0x019C FFFF	
Reserved		0x01A0 0000 – 0x01A3 FFFF	
Internal Peripheral Bus McBSP 2 Registers		0x01A4 0000 – 0x01A7 FFFF	
Reserved	–	0x01A8 0000 – 0x01FF FFFF	–
VIM Control/Status Registers	EMIF	0x0200 0000 – 0x021F FFFF	VIM Operating Manual [2]
DSP Control Registers		0x0220 0000 – 0x02FF FFFF	Section 3.6
Reserved	–	0x0300 0000 – 0x3FFF FFFF	–
Global Bus FIFO Data	XBus I/O Port	0x4000 0000 – 0x4FFF FFFF	Section 9.4.1
XBus Inter–Processor XX Bi–FIFO Data		0x5000 0000	
XBus Inter–Processor XX Bi–FIFO Mailbox		0x5000 0020	
XBus Inter–Processor YY Bi–FIFO Data		0x6000 0000	
XBus Inter–Processor YY Bi–FIFO Mailbox		0x6000 0020	
Mezzanine (VIM) Bi–FIFO Data		0x7000 0000	
Mezzanine (VIM) Bi–FIFO Mailbox		0x7000 0020	
Internal Data RAM C6202 Internal Data RAM C6203	Internal	0x8000 0000 – 0x8001 FFFF 0x8000 0000 – 0x8007 FFFF	Section 3.2 and TI Per. Ref. Guide [1]
Reserved	–	0x8008 0000 – 0xFFFF FFFF	–

1 – Memory maps of the internal 'C6x resources are provided in Texas Instruments TMS320C6000 Peripherals Reference Guide.

2 – The VIM Control/Status Registers are described in the Operating Manual supplied with each VIM module.

3.4 Master DSP Local SDRAM Operation

Each Master 'C6x processor is connected, on its EMIF bus, to an 8–MByte local SDRAM (16–MByte SDRAM with Option 340), which is used for 'C6x boot operation at reset or power–up of the board (see Section 10.4). Each 'C6x processor can access its local SDRAM starting at memory address **0x0000 0000**.



The first 64 kBytes of each local SDRAM is reserved for use by the default 'C6x boot firmware (Section 10.4). **Loading user code into this area or accessing data in this area can cause unpredictable behavior**.

3.5 Master DSP XBus Host Port Interface Operation

Each Master 'C6x processor's Expansion Bus (XBus) Host Port is connected to the Local Bus of a PLX PCI 9054. Using its XBus Host Port Interface (HPI), in synchronous host port mode, each Master DSP has access to the PCI 9054 registers, to the associated Slave DSP's Host Port registers, and to other global resources through the PCI 9054 and the i960 PCI Bus Bridge. Refer to the TI TMS320C6000 Peripherals Reference Guide for detailed description of 'C6x Expansion Bus Host Port operation.

The following table shows the resources available to each Master DSP from its XBus HPI, the path to each resource, the XBus HPI addresses of that resource, and a reference to information on 'C6x operation of each resource.

Table 3–2: Master DSP XBus Host Port Memory Map

Resource	Path	XBus HPI Address	Information
i960 Messaging Unit	Secondary PCI Bus	0x1000 0000 – 0x1000 0FFF	Section 3.5.5
Global SDRAM Reserved [1]	Global Bus	0x1000 1000 – 0x1040 FFFF	Section 10.4.1
Global SDRAM User Data	Global Bus, 32–bit mode	0x1041 0000 – 0x10FF FFFF	Section 3.5.1
	Global Bus, 64–bit mode	0x1041 0000 – 0x11FF FFFF	
Node A Global Bus FIFO	Global Bus, 32–bit mode [2]	0x1100 0000 – 0x1107 FFFF	Section 9.4.1
Node C Global Bus FIFO		0x1108 0000 – 0x110F FFFF	
Node D Global Bus FIFO		0x1110 0000 – 0x1117 FFFF	
Node B Global Bus FIFO		0x1118 0000 – 0x111F FFFF	
Universe II Registers	Primary PCI Bus	0x1850 0000 – 0x1850 0FFF	Section 3.5.2
PXB Registers (Option 005 only)		0x1860 0000 – 0x1860 FFFF	Section 3.5.7
PLX PCI 9054 Registers	XBus	0x2000 0000 – 0x27FF FFFF	Section 3.5.3
Slave DSP HPI (Model 4293 only)		0x2800 0000 – 0x28FF FFFF	Section 3.5.6
VMEbus Master Access	Primary PCI Bus	0x8001 0000 – 0xEFFF FFFF	Section 3.5.4
PMC Module Access RACEway Access (Option 005 only)		0xF000 0000 – 0xFFFF FFFF	Section 3.5.7

1 – This area of Global SDRAM is reserved for Pentek boot firmware and should not be used.

2 – If PCB jumper **J10** is set for 64–bit mode (Section 2.2.4), the four Global Bus FIFOs are not available.

3.5 Master DSP XBus Host Port Interface Operation (continued)

3.5.1 Global SDRAM

Each Master 'C6x has access to the Global SDRAM User Data area, starting at its XBus HPI address **0x1041 0000**. The available size of Global SDRAM is 16 or 32 MBytes, depending on the setting of PCB jumper **J10** (Section 2.2.4). Note that the first 4 MBytes of Global SDRAM (**0x1000 1000 – 0x1040 FFFF**) are reserved for default boot firmware (see Section 10.4.1) and should not be used.

3.5.2 Universe II Registers

The Tundra Universe II provides programmable registers for VME system configuration and for control of Universe II operational characteristics: PCI Configuration Space, VMEbus Configuration and Status Registers, and Universe II Device Specific Status Registers. A quick reference listing of these registers is provided in Appendix E. Refer to the Tundra Universe II User Manual for description of each register. Each Master 'C6x processor can access these registers, starting at its XBus HPI address **0x1850 0000**.

3.5.3 PLX PCI 9054 Registers

Each PCI 9054 provides several internal registers on its Local Bus: Configuration, DMA, Mailbox, PCI–to–Local and Local–to–PCI Doorbell, Messaging Queue, Power Management, Hot Swap, and VPD registers. These registers are described in the PLX PCI 9054 Data Book. Each Master 'C6x can access these registers for its PCI 9054, starting at its XBus HPI address **0x2000 0000**. Also, the default 'C6x firmware provides user functions that can be called using the PCI 9054 Mailbox and Doorbell registers (see Section 10.5).

3.5.4 VMEbus Master Access

When the Model 4292/4293 is a VMEbus Master, each Master DSP can access data and perform D16, D32, and D64 transfers in all relevant VMEbus address spaces, at the XBus Host Port addresses shown in the following table. See Section 7.2 for more information on VMEbus Master operation.

Table 3–3: Master DSP VMEbus Master Memory Map

VMEbus Address Space (Address Modifiers)	XBus HPI Address	Generated VME Address
A16/D16 (Non–Privileged, 0x29)	**0x8001 0000 – 0x8001 FFFF**	**0x0000 – 0xFFFF**
A16/D32 (Non–Privileged, 0x29)	**0x8002 0000 – 0x8002 FFFF**	**0x0000 – 0xFFFF**
A24/D16 (Standard, Non–Privileged, Data, 0x39)	**0x8100 0000 – 0x81FF FFFF**	**0x00 0000 – 0xFF FFFF**
A24/D32 (Standard, Non–Privileged, Data, 0x39)	**0x8200 0000 – 0x82FF FFFF**	**0x00 0000 – 0xFF FFFF**
A32/D32 (Extended, Non–Privileged, Data, 0x09)	**0x9000 0000 – 0xBFFF FFFF**	**0x0000 0000 – 0x2FFF FFFF**
A32/D64 (Extended, Non–Privileged, Data, 0x09)	**0x8400 0000 – 0x8FFF FFFF**	**0x0400 0000 – 0x0FFF FFFF**

3.5 Master DSP XBus Host Port Interface Operation (continued)

3.5.5 i960 Messaging Unit

The i960 Messaging Unit (MU) allows a Master 'C6x and the i960 I/O processor to communicate through command passing and interrupt generation. Description of the i960 MU is provided in the Intel i960® RM/RN I/O Processor Developer's Manual. Each Master 'C6x processor can access the i960 MU registers starting at its XBus HPI address **0x1000 0000**. The i960 Command Processor (see Chapter 12) provides user functions that can be called by each Master 'C6x processor using the i960 MU.

3.5.6 Slave DSP XBus Host Port (Model 4293 only)

Each Master DSP has access to two registers on the associated Slave DSP's XBus Host Port: the XBus Internal Slave Address Register (XBISA) and the XBus Data Register (XBD). Definitions of these registers are provided in the Texas Instruments TMS320C6000 Peripherals User Guide. The XBISA and XBD registers allow the Master DSP read/write access to any address on the Slave DSP Memory Map (Section 4.3).

- The XBISA controls the memory location, in the Slave DSP, being accessed by the Master DSP. Each Master 'C6x can access the associated Slave's XBISA register at XBus HPI addresses **0x2800 4000** to **0x2800 7FFF**.

- The XBD contains the data that was read from the memory location if the access is a read, or the data that is written to the memory location if the access is a write. Each Master 'C6x can access the associated Slave's XBD register at XBus HPI addresses **0x2800 0000** to **0x2800 3FFF**.

To read or write to a Slave resource, first write the desired Slave DSP memory address into the XBISA, then read or write the data at that address using the XBD. The 'C6x can perform single 32–bit, 16–bit, or 8–bit accesses to the XBD. Bursts longer than one word to the XBD should always be 32 bits wide.

3.5.7 PMC Module Access

The PCI Mezzanine Card (PMC) site accepts industry–standard PMC modules. If a PMC module is installed, each Master 'C6x processor can access PMC local address space resources (I/O, memory, registers, DMA, etc.) starting at XBus HPI address **0xF000 0000**. Each PMC module is supplied with an Operating Manual that describes its operation and programmable resources.

When the Option 005 RACEway PMC is installed, each Master 'C6x can access RACEway address space, for outgoing (Master) and incoming (Slave) RACEway operations, starting at XBus HPI address **0xF000 0000**, and can access the module's PXB++ chip configuration registers starting at memory address **0x1860 0000**. Refer to Appendix D for description of this RACEway interface.

3.6 Master DSP Control Registers

Each Master 'C6x processor has access to 32−bit local control registers. The following table presents a memory map of these registers. The referenced sections, following this table, provide information about each register.

Table 3−4: Master DSP Register Memory Map

Memory Address	DSP Register	Access	Information
0x0220 0008	Global Bus FIFO Interrupt Enable (BIER0)	R/W	Section 3.6.1
0x0220 000C	XBus Inter−Processor XX Bi−FIFO Interrupt Enable (BIER1)	R/W	
0x0220 0010	XBus Inter−Processor YY Bi−FIFO Interrupt Enable (BIER2)	R/W	
0x0220 0014	Mezzanine Bi−FIFO Interrupt Enable (BIER3)	R/W	
0x0220 0020	Global Bus FIFO Flags (BIFR0)	R/Clr	Section 3.6.2
0x0220 0024	XBus Inter−Processor XX Bi−FIFO Flags (BIFR1)	R/Clr	
0x0220 0028	XBus Inter−Processor YY Bi−FIFO Flags (BIFR2)	R/Clr	
0x0220 002C	Mezzanine Bi−FIFO Flags (BIFR3)	R/Clr	
0x0220 0038	Global Bus FIFO Status (BISR0)	R.O.	Section 3.6.3
0x0220 003C	XBus Inter−Processor XX Bi−FIFO Status (BISR1)	R.O.	
0x0220 0040	XBus Inter−Processor YY Bi−FIFO Status (BISR2)	R.O.	
0x0220 0044	Mezzanine Bi−FIFO Status (BISR3)	R.O.	
0x0220 0048	Miscellaneous Interrupt Flags (MIFR)	R/Clr	Section 3.6.4
0x0220 004C	Miscellaneous Interrupt Status (MISR)	R.O.	Section 3.6.5
0x0220 0050	External Interrupt #4 Enable (EIER4) [1]	R/W	Section 3.6.6
0x0220 0054	External Interrupt #5 Enable (EIER5) [1]	R/W	
0x0220 0058	External Interrupt #6 Enable (EIER6) [1]	R/W	
0x0220 005C	External Interrupt #7 Enable (EIER7) [1]	R/W	
0x0220 0060	Board Status (BSR)	R.O.	Section 3.6.7
0x0220 0064	Miscellaneous Control (MCR)	R/W	Section 3.6.8
0x0220 0068	General Purpose Semaphores (GPSR)	R/W	Section 3.6.9
0x0220 006C	External Interrupt Mode (EIMR) [2]	R/W	Section 3.6.10
0x0220 0070	External Interrupt Status (EISR) [2]	R/W	Section 3.6.11
0x0220 0074	Miscellaneous Interrupt Flags #2 (MIFR2) [2]	R/Clr	Section 3.6.12
0x0220 0078	Miscellaneous Interrupt Status #2 (MISR2) [2]	R.O.	Section 3.6.13

1 − There are four External Interrupt Enable Registers, one for each 'C6x processor external interrupt. Each register has an identical set of interrupt enable bits that allow you to select which of the four 'C6x external interrupts to use for each interrupt source.

2 − This register is available only on boards with FPGA firmware revision 1 or greater (see Section 3.6.7.1).

3.6 Master DSP Control Registers (continued)

3.6.1 FIFO Interrupt Enable Registers

Four DSP registers enable interrupts for the FIFO boundary flag conditions (Almost Full, Almost Empty, Mailbox, etc.) associated with each FIFO accessible to a Master DSP. These registers are:

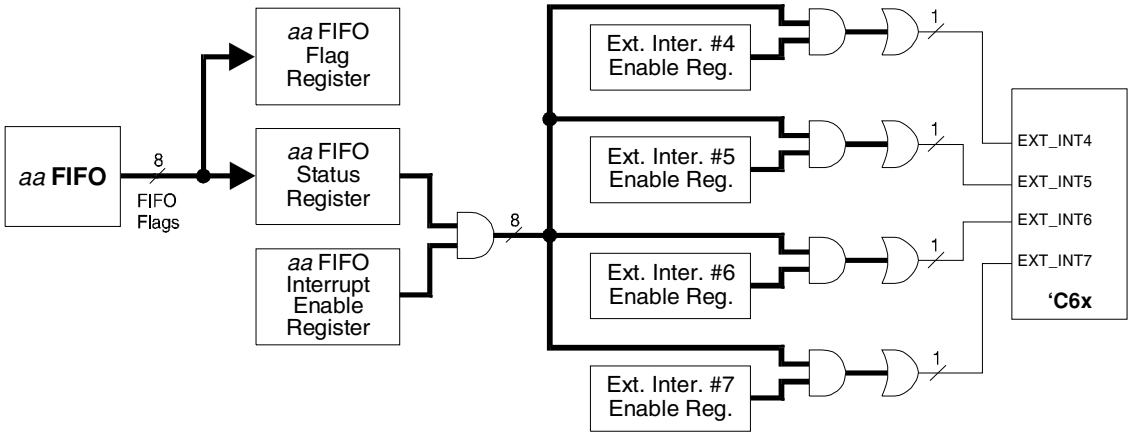
- Global Bus FIFO Interrupt Enable Register (BIER0)
- XBus Inter–Processor XX Bi–FIFO Interrupt Enable Register (BIER1)
- XBus Inter–Processor YY Bi–FIFO Interrupt Enable Register (BIER2)
- Mezzanine Bi–FIFO Interrupt Enable Register (BIER3)

Each Master 'C6x processor has a unique copy of each FIFO Interrupt Enable Register. Each register contains interrupt enable bits that allow the associated FIFO boundary flags to assert an interrupt. Set a bit to logic '1' to enable the interrupt for the specified flag condition. If an enable bit is set to logic '1' and that FIFO flag condition is active, an interrupt can occur to the 'C6x processor (see note, below). Clear the bit to logic '0' to disable the interrupt.



These are nested interrupt enable bits. In order for the 'C6x to receive any interrupts that you enable with these FIFO Interrupt Enable Registers, you must also enable one of the 'C6x external interrupts for these registers, using the External Interrupt Enable Registers, Section 3.6.6.

The following figure shows a simplified logic flow of the FIFO flags, the interrupt enable registers, and the associated 'C6x interrupts. See Section 9.5 for detailed logic diagrams of the FIFO interrupts.



Note: On boards with FPGA firmware revision 1

Figure 3–1: FIFO Flags and Interrupts

3.6 Master DSP Control Registers (continued)

3.6.1 FIFO Interrupt Enable Registers (continued)

The following four tables (below and on the next page) show the bit layout of each FIFO Interrupt Enable Register. The subsections following these tables define the enable bits for each register—these bit definitions are the same for all four registers. Note that the Global Bus FIFOs have only Incoming data capability and have no Mailboxes. Refer to Chapter 9 for more information on the operation and programming of the FIFO boundary flag conditions.

Table 3–5: Global Bus FIFO Interrupt Enable Register (BIER0)
R/W @ 0x0220 0008

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Reserved	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Reserved			
Function	Write zeros, Mask read	0 = Interrupt disabled 1 = Interrupt enabled			Write with zeros, Mask when reading			
All bits default to the logic '0' state at power up and reset.								

Table 3–6: XBus Inter–Processor XX Bi–FIFO Interrupt Enable Register (BIER1)
R/W @ 0x0220 000C

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	0 = Interrupt disabled 1 = Interrupt enabled							
All bits default to the logic '0' state at power up and reset.								

3.6 Master DSP Control Registers (continued)

3.6.1 FIFO Interrupt Enable Registers (continued)

Table 3–7: XBus Inter–Processor YY Bi–FIFO Interrupt Enable Register (BIER2)
R/W @ 0x0220 0010

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	0 = Interrupt disabled 1 = Interrupt enabled							
All bits default to the logic '0' state at power up and reset.								

Table 3–8: Mezzanine Bi–FIFO Interrupt Enable Register (BIER3)
R/W @ 0x0220 0014

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	0 = Interrupt disabled 1 = Interrupt enabled							
All bits default to the logic '0' state at power up and reset.								

The following subsections describe the FIFO Interrupt Enable Register bits. Except as noted, these bit definitions are the same for all four registers.

3.6.1.1 Incoming Mailbox D7

Set this bit to logic '1' to enable an interrupt when a write occurs to the Incoming Mailbox word of the FIFO, by the device on the other side of the FIFO. Clear the bit to logic '0' to disable the interrupt. This bit does not exist in the Global Bus FIFO register, BIER0.

3.6.1.2 Incoming Almost Full D6

Set this bit to logic '1' to enable an interrupt when data being written to the Incoming FIFO, by the device on the other side of the FIFO, reaches the Almost Full level. Clear the bit to logic '0' to disable the interrupt.

3.6 Master DSP Control Registers (continued)

3.6.1 FIFO Interrupt Enable Registers (continued)

3.6.1.3 Incoming Not Almost Empty D5

Set this bit to logic '1' to enable an interrupt when data being written to the Incoming FIFO, by the device on the other side of the FIFO, reaches the Almost Empty level. Clear the bit to logic '0' to disable the interrupt.

3.6.1.4 Incoming Not Empty D4

Set this bit to logic '1' to enable an interrupt when data is written to an empty Incoming FIFO, by the device on the other side of the FIFO. Clear the bit to logic '0' to disable the interrupt.

3.6.1.5 Outgoing Mailbox D3

Set this bit to logic '1' to enable an interrupt when a write occurs to the Outgoing Mailbox word of the FIFO, by this processor. Clear the bit to logic '0' to disable the interrupt. This bit does not exist in the Global Bus FIFO register, BIER0.

3.6.1.6 Outgoing Not Almost Full D2

Set this bit to logic '1' to enable an interrupt when data being read from the Outgoing FIFO, by the device on the other side of the FIFO, falls to the Almost Full level. Clear the bit to logic '0' to disable the interrupt. This bit does not exist in the Global Bus FIFO register, BIER0.

3.6.1.7 Outgoing Almost Empty D1

Set this bit to logic '1' to enable an interrupt when data being read from the Outgoing FIFO, by the device on the other side of the FIFO, falls to the Almost Empty level. Clear the bit to logic '0' to disable the interrupt. This bit does not exist in the Global Bus FIFO register, BIER0.

3.6.1.8 Outgoing Empty D0

Set this bit to logic '1' to enable an interrupt when all data has been read from the Outgoing FIFO, by the device at the other side of the FIFO. Clear the bit to logic '0' to disable the interrupt. This bit does not exist in the Global Bus FIFO register, BIER0.

3.6 Master DSP Control Registers (continued)

3.6.2 FIFO Flag Registers

Four DSP registers contain latched flag bits for the FIFO boundary flag conditions associated with each FIFO. These DSP registers are:

- Global Bus FIFO Flag Register (BIFR0)
- XBus Inter–Processor XX Bi–FIFO Flag Register (BIFR1)
- XBus Inter–Processor YY Bi–FIFO Flag Register (BIFR2)
- Mezzanine Bi–FIFO Flag Register (BIFR3)

Each register is associated with the programmable boundary flags (Almost Full, Almost Empty, Mailbox, etc.) from the FIFO. Each Master 'C6x processor has a unique copy of each FIFO Flag Register.

Each flag bit is a latched version of the associated FIFO boundary flag, and is set when that flag transitions from inactive (logic '0') to active (logic '1'). Since FIFO flag conditions may be transient, they are latched in these registers so that the processor can determine the cause. Each flag bit is a read/clear bit. When read as logic '1', a FIFO flag condition has been latched. When read as logic '0', the FIFO flag is not active. A flag bit remains latched until you clear it by writing logic '1' to that bit position.



When you attempt to clear a latched flag bit on boards with FPGA Revision 0 (see Section 3.6.7.1), the flag bit will stay at logic '1' if the FIFO flag condition is not cleared and the associated FIFO Status Register bit (Section 3.6.3) is still at logic '1'.
On later boards, the bit clears regardless of the state of the FIFO flag.

The following four tables (below and on the next page) show the layout of each register. The subsections following these tables define the bits for each register—these bit definitions are the same for all four registers. Note that the Global Bus FIFOs have only Incoming data capability and have no Mailboxes.

Table 3–9: Global Bus FIFO Flag Register (BIFR0)
R/Clr @ 0x0220 0020

	D31 – D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	Reserved	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Reserved			
Function	Write zeros, Mask read	**READ**: 0 = FIFO Flag not active 1 = FIFO Flag latched **WRITE**: 0 = No effect 1 = Clear latch			Write with zeros, Mask when reading			

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

3.6 Master DSP Control Registers (continued)

3.6.2 FIFO Flag Registers (continued)

Table 3–10: XBus Inter–Processor XX Bi–FIFO Flag Register (BIFR1)
R/Clr @ 0x0220 0024

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	**READ**: 0 = FIFO Flag not active; 1 = FIFO Flag latched **WRITE**: 0 = No effect; 1 = Clear latch							

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

Table 3–11: XBus Inter–Processor YY Bi–FIFO Flag Register (BIFR2)
R/Clr @ 0x0220 0028

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	**READ**: 0 = FIFO Flag not active; 1 = FIFO Flag latched **WRITE**: 0 = No effect; 1 = Clear latch							

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

Table 3–12: Mezzanine Bi–FIFO Flag Register (BIFR3)
R/Clr @ 0x0220 002C

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	**READ**: 0 = FIFO Flag not active; 1 = FIFO Flag latched **WRITE**: 0 = No effect; 1 = Clear latch							

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

3.6 Master DSP Control Registers (continued)

3.6.2 FIFO Flag Registers (continued)

The following subsections describe the FIFO Flag Register bits. These bit definitions also apply to the FIFO Status Register (Section 3.6.3). For all bits, logic '1' indicates that the FIFO flag condition is active (true). The Global Bus FIFOs have only Incoming FIFO data bits (D6, D5, & D4) and no Mailbox bits.

3.6.2.1 Incoming Mailbox D7

This flag indicates that a write has occurred to the Incoming Mailbox word of the FIFO, by the device on the other side of the FIFO.

3.6.2.2 Incoming Almost Full D6

This flag indicates that data being written to the Incoming FIFO, by the device on the other side of the FIFO, is at or above the Almost Full level.

3.6.2.3 Incoming Not Almost Empty D5

This flag indicates that data being written to the Incoming FIFO, by the device on the other side of the FIFO, is at or above the Almost Empty level.

3.6.2.4 Incoming Not Empty D4

This flag indicates that data was written to an empty Incoming FIFO, by the device on the other side of the FIFO.

3.6.2.5 Outgoing Mailbox D3

This flag indicates that a write has occurred to the Outgoing Mailbox word of the FIFO, by this processor.

3.6.2.6 Outgoing Not Almost Full D2

This flag indicates that the data being read from the Outgoing FIFO, by the device on the other side of the FIFO, is at or below the Almost Full level.

3.6.2.7 Outgoing Almost Empty D1

This flag indicates that the data being read from the Outgoing FIFO, by the device on the other side of the FIFO, is at or below the Almost Empty level.

3.6.2.8 Outgoing Empty D0

This flag indicates that all data has been read from the Outgoing FIFO, by the device at the other side of the FIFO.

3.6 Master DSP Control Registers (continued)

3.6.3 FIFO Status Registers

Four DSP registers contain status bits for the FIFO boundary flag conditions associated with each FIFO accessible to a Master DSP. These registers are:

- Global Bus FIFO Status Register (BISR0)
- XBus Inter–Processor XX Bi–FIFO Status Register (BISR1)
- XBus Inter–Processor YY Bi–FIFO Status Register (BISR2)
- Mezzanine Bi–FIFO Status Register (BISR3)

Each status register is associated with the programmable boundary flags (Almost Full, Almost Empty, Mailbox, etc.) from the FIFO. Each Master 'C6x processor has a unique copy of each FIFO Status Register.

Each status bit is a buffered version of the associated FIFO boundary flag condition, and may be monitored by software to verify that the condition has been properly cleared. Each status register is read only. When read as logic '1', a FIFO flag condition is currently active (true). When read as logic '0', the flag condition is inactive.

When a FIFO boundary flag condition transitions from inactive to active (from logic '0' to logic '1'), the associated flag bit is set in the associated FIFO Flag Registers (Section 3.6.2), and a 'C6x external interrupt can be generated if it is enabled in the Interrupt Enable Registers (Sections 3.6.1 and 3.6.6).

The following four tables (below and on the next page) show the bit layout of each FIFO Status Register. See Sections 3.6.2.1 through 3.6.2.8 for definitions of the register bits—these bit definitions are the same for all four registers. Note that the Global Bus FIFOs have only Incoming data capability and have no Mailboxes.

Table 3–13: Global Bus FIFO Status Register (BISR0)
R.O. @ 0x0220 0038

	D31 – D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	Reserved	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Reserved			
Function	Mask read	0 = FIFO Flag not active 1 = FIFO Flag active			Mask when reading			
When reset, including power–up, the state of this register is unknown.								

3.6 Master DSP Control Registers (continued)

3.6.3 FIFO Status Registers (continued)

Table 3–14: XBus Inter–Processor XX Bi–FIFO Status Register (BISR1)
R.O. @ 0x0220 003C

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	0 = FIFO Flag not active 1 = FIFO Flag active							
When reset, including power–up, the state of this register is unknown.								

Table 3–15: XBus Inter–Processor YY Bi–FIFO Status Register (BISR2)
R.O. @ 0x0220 0040

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	0 = FIFO Flag not active 1 = FIFO Flag active							
When reset, including power–up, the state of this register is unknown.								

Table 3–16: Mezzanine Bi–FIFO Status Register (BISR3)
R.O. @ 0x0220 0044

	D31 – D8							
Bit Name	Reserved							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	Incoming Mailbox	Incoming Almost Full	Incoming Not Almost Empty	Incoming Not Empty	Outgoing Mailbox	Outgoing Not Almost Full	Outgoing Almost Empty	Outgoing Empty
Function	0 = FIFO Flag not active 1 = FIFO Flag active							
When reset, including power–up, the state of this register is unknown.								

3.6 Master DSP Control Registers (continued)

3.6.4 Miscellaneous Interrupt Flag Register

This register contains flag bits for miscellaneous interrupt sources. Each Master 'C6x processor has a unique copy of this register. Each flag bit is a latched version of the associated bit in the Miscellaneous Interrupt Status Register (Section 3.6.5), and is set when an interrupt condition transitions from inactive (logic '0') to active (logic '1').

The miscellaneous interrupt status and flag bits represent the same interrupt signals—the only difference is when the interrupt source is cleared (goes not active), the status bit clears automatically in the status register (Section 3.6.5), while the flag bit must be cleared by writing to this flag register.

Since the interrupting conditions may be transient, they are latched in this register so that the processor can determine the cause. Each flag bit is a read/clear bit. When read as logic '1', an interrupt condition has been latched. When read as logic '0', the interrupt is inactive. A flag bit remains latched until you clear it by writing logic '1' to that bit.



When you attempt to clear a latched flag bit on boards with FPGA Revision 0 (see Section 3.6.7.1), the flag bit will stay at logic '1' if the interrupt condition is not cleared and the associated Miscellaneous Interrupt Status Register bit (Section 3.6.5) is still at logic '1'. On later boards, the bit clears regardless of the state of the interrupt.

The following table shows this register's bit layout. The subsections following the table describe each status bit (these bit assignments and interrupt definitions also apply to the Miscellaneous Interrupt Status Register, Section 3.6.5).

Table 3–17: Miscellaneous Interrupt Flag Register (MIFR)
R/Clr @ 0x0220 0048

	D31 – D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME	PLX	VIM
Function	Write zeros, Mask read	**READ**: 0 = Interrupt not active; 1 = Interrupt latched **WRITE**: 0 = Normal operation; 1 = Clear latch						

* Model 4293 only

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

NOTE: On boards with FPGA Revision 1 or greater (see Section 3.6.7.1), these flag bits also appear in the Miscellaneous Interrupt Flag Register #2, Section 3.6.12. When you clear these flag bits, the associated bits in that register are also cleared.

3.6 Master DSP Control Registers (continued)

3.6.4 Miscellaneous Interrupt Flag Register (continued)

3.6.4.1 Slave DSP (Model 4293 only) D6

This bit is the interrupt signal from the Slave DSP (Section 4.5.3.1). When read as logic '1', an interrupt has been received from the Slave DSP. When read as logic '0', the interrupt is not active.

3.6.4.2 Semaphore D5

This bit is the OR'ed result of the three Semaphore Register Grant bits (Section 3.6.9). When read as logic '1', a semaphore grant bit is active (at least one semaphore is currently 'owned' by a processor). When read as logic '0', the semaphore is available (not owned by any processor).

3.6.4.3 i960 PINT D4

This bit is the PCI interrupt, P_INTx, from the i960 (Section 8.4). When read as logic '1', this interrupt signal is active. When read as logic '0', the interrupt is not active. The i960 P_INTx interrupt lines are routed to each DSP Node as follows:

P_INTA to DSP Node A	**P_INTB** to DSP Node C
P_INTC to DSP Node D	**P_INTD** to DSP Node B

3.6.4.4 i960 SwiftNet D3

This bit is the i960 SwiftNet interrupt from the i960 DSP Interrupt Request Register (Section 5.5.1). When read as logic '1', an interrupt is active. When read as logic '0', the interrupt is not active.

3.6.4.5 VME D2

This bit is the interrupt LINT0 from the VME interface controller (Tundra Universe II). When read as logic '1', this interrupt signal is active. When logic '0', the interrupt is not active.

3.6.4.6 PLX D1

This bit is the interrupt LINT from the PCI bus master (PLX PCI 9054) for this DSP node. When read as logic '1', this interrupt signal is active. When read as logic '0', the interrupt is not active.

3.6.4.7 VIM D0

This bit is the VIM interrupt DBINT from the installed VIM module for this node. When read as logic '1', the VIM interrupt signal is active. When read as logic '0', the interrupt is not active.

3.6 Master DSP Control Registers (continued)

3.6.5 Miscellaneous Interrupt Status Register

This register contains status bits for several miscellaneous interrupt sources. Each Master 'C6x processor has a unique version of this register. Each status bit is a buffered version of the interrupting source and may be monitored by software to verify that an interrupting condition has been properly cleared.

This register is read only. When a bit is read as logic '1', the interrupt condition is active (true). When read as logic '0', the interrupt is not active.

When an interrupting source transitions from inactive to active (status bit transitions from logic '0' to logic '1'), the associated flag bit is set in the Miscellaneous Interrupt Flag Register (Section 3.6.4), and a 'C6x external interrupt can be generated for the associated bits that are enabled in the 'C6x External Interrupt Enable Registers (Section 3.6.6) (see **NOTE** below).

The miscellaneous interrupt status and flag bits represent the same interrupt signals—the only difference is when the interrupt source is cleared (goes not active), the status bit clears automatically in this register, while the flag bit must be cleared by writing to the flag register, Section 3.6.4.

The following table shows this register's bit layout. See Sections 3.6.4.2 through 3.6.4.7 for description of interrupt signals for these bits.

Table 3–18: Miscellaneous Interrupt Status Register (MISR)
R.O. @ 0x0220 004C

	D31 – D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME	PLX	VIM
Function	Mask read	0 = Interrupt not active 1 = Interrupt active						

* Model 4293 only
When reset, including power–up, the state of this register is unknown.

NOTE: On boards with FPGA Revision 1 or greater (see Section 3.6.7.1), these status bits also appear in the Miscellaneous Interrupt Status Register #2, Section 3.6.13, and this register is not used for the 'C6x external interrupts. On these boards, refer to the External Interrupt Mode Register, Section 3.6.10, for the 'C6x interrupts.

3.6 Master DSP Control Registers (continued)

3.6.6 'C6x External Interrupt Enable Registers

There are four External Interrupt Enable Registers (4, 5, 6, and 7), one for each 'C6x processor external interrupt (**EXT_INT4**, **EXT_INT5**, **EXT_INT6**, and **EXT_INT7**). Each register has an identical set of interrupt enable bits. These allow you to select which of the four 'C6x external interrupts to use for each interrupt source. Each Master 'C6x processor has a unique copy of each register, and can enable different sets of sources for each external interrupt.

Each register contains enable bits for the interrupt sources to be associated with one 'C6x external interrupt (4, 5, 6, or 7).

- On boards with FPGA Revision 0 (see Section 3.6.7.1), the sources of these interrupts are the Miscellaneous Interrupt Status Register (Section 3.6.5) and the FIFO Status Registers (Section 3.6.3) enabled by the appropriate FIFO Interrupt Enable Registers (Section 3.6.1).

- On boards with FPGA Revision 1 or greater, the source of each interrupt bit can be selected from either the Miscellaneous Interrupt Flag Register #2 (Section 3.6.12) or Status Register #2 (Section 3.6.13). To select the interrupt mode (source) of each bit, refer to Section 3.6.10.

Set an enable bit to logic '1' to enable the external interrupt. If an enable bit is set to logic '1' and the source is active, an interrupt is asserted to the 'C6x processor. Clear the enable bit to logic '0' to disable that interrupt.

The following four tables (below and on the next page) show the layout of each register—all registers have identical formats. The subsections following these tables identify the interrupt source for each enable bit—these descriptions are the same for all four registers.

Table 3–19: External Interrupt #4 Enable Register (EIER4)
R/W @ 0x0220 0050

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Write with zeros, Mask when reading	0 = Interrupt disabled 1 = Interrupt enabled				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	0 = Interrupt disabled 1 = Interrupt enabled						Write with zeros, Mask when reading	

* Model 4293 only
All bits default to the logic '0' state at power up and reset.

3.6 Master DSP Control Registers (continued)

3.6.6 'C6x External Interrupt Enable Registers (continued)

Table 3–20: External Interrupt #5 Enable Register (EIER5)
R/W @ 0x0220 0054

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Write with zeros, Mask when reading	0 = Interrupt disabled 1 = Interrupt enabled				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	0 = Interrupt disabled 1 = Interrupt enabled						Write with zeros, Mask when reading	

All bits default to the logic '0' state at power up and reset. * Model 4293 only

Table 3–21: External Interrupt #6 Enable Register (EIER6)
R/W @ 0x0220 0058

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Write with zeros, Mask when reading	0 = Interrupt disabled 1 = Interrupt enabled				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	0 = Interrupt disabled 1 = Interrupt enabled						Write with zeros, Mask when reading	

All bits default to the logic '0' state at power up and reset. * Model 4293 only

Table 3–22: External Interrupt #7 Enable Register (EIER7)
R/W @ 0x0220 005C

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Write with zeros, Mask when reading	0 = Interrupt disabled 1 = Interrupt enabled				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	0 = Interrupt disabled 1 = Interrupt enabled						Write with zeros, Mask when reading	

All bits default to the logic '0' state at power up and reset. * Model 4293 only

3.6 Master DSP Control Registers (continued)

3.6.6 'C6x External Interrupt Enable Registers (continued)

The following subsections describe the interrupt enable bits of all four external interrupt enable registers.

3.6.6.1 Slave DSP (Model 4293 only) D12

Set this bit to logic '1' to enable an interrupt from the Slave DSP (Section 4.5.3). Clear the bit to logic '0' to disable the interrupt.

3.6.6.2 Semaphore D11

Set this bit to logic '1' to enable an interrupt from the OR'ed result of the three Semaphore Register Grant bits (Section 3.6.9). Clear the bit to logic '0' to disable the interrupt.

3.6.6.3 i960 PINT D10

Set this bit to logic '1' to enable an interrupt from the i960 processor PCI interrupt signal, P_INTx (Section 8.4). Clear the bit to logic '0' to disable the interrupt. The i960 P_INTx interrupt lines are routed to each DSP Node as follows:

P_INTA to DSP Node A
P_INTB to DSP Node C
P_INTC to DSP Node D
P_INTD to DSP Node B

3.6.6.4 i960 SwiftNet D9

Set this bit to logic '1' to enable the i960 SwiftNet interrupt from the i960 processor's DSP Interrupt Request Register (Section 5.5.1). Clear the bit to logic '0' to disable the interrupt.

3.6.6.5 VME D8

Set this bit to logic '1' to enable an interrupt from the VME interface controller Tundra Universe II interrupt signal LINT0. Clear the bit to logic '0' to disable the interrupt.

3.6.6.6 PLX D7

Set this bit to logic '1' to enable an interrupt from the PCI Bus Master Interface PLX PCI 9054 interrupt signal LINT. Clear the bit to logic '0' to disable the interrupt.

3.6 Master DSP Control Registers (continued)

3.6.6 'C6x External Interrupt Enable Registers (continued)

3.6.6.7 VIM D6

Set this bit to logic '1' to enable an interrupt from the VIM interrupt signal DBINT. Clear the bit to logic '0' to disable the interrupt.

3.6.6.8 Mezzanine BI–FIFO D5

Set this bit to logic '1' to enable an interrupt from any Mezzanine Bi–FIFO boundary flag condition that is enabled by the Mezzanine Bi–FIFO Interrupt Enable Register (Section 3.6.1). Clear the bit to logic '0' to disable the interrupt.

3.6.6.9 XBus I/P YY BI–FIFO D4

Set this bit to logic '1' to enable an interrupt from any XBus Inter–Processor YY Bi–FIFO boundary flag condition that is enabled by the XBus Inter–Processor YY Bi–FIFO Interrupt Enable Register (Section 3.6.1). Clear the bit to logic '0' to disable the interrupt.

3.6.6.10 XBus I/P XX BI–FIFO D3

Set this bit to logic '1' to enable an interrupt from any XBus Inter–Processor XX Bi–FIFO boundary flag condition that is enabled by in the XBus Inter–Processor XX Bi–FIFO Interrupt Enable Register (Section 3.6.1). Clear the bit to logic '0' to disable the interrupt.

3.6.6.11 Global Bus FIFO D2

Set this bit to logic '1' to enable an interrupt from any Global Bus FIFO boundary flag condition that is enabled by the Global Bus FIFO Interrupt Enable Register (Section 3.6.1). Clear the bit to logic '0' to disable the interrupt.

NOTE: On boards with FPGA Revision 1 or greater, the source of these interrupts can be from either the Miscellaneous Interrupt Status Register #2 (Section 3.6.13) or Flag Register #2 (Section 3.6.12).

On boards with FPGA Revision 0, the sources of these interrupts include the Miscellaneous Interrupt Status Register (Section 3.6.5) and the respective FIFO Status Registers (Section 3.6.3).

3.6 Master DSP Control Registers (continued)

3.6.7 Board Status Register

This register contains read–only status bits for the board. Each Master 'C6x processor has a unique copy of this register.

The following table shows this register's bit layout. The subsections following this table describe the bits.

Table 3–23: Board Status Register (BSR)
R.O. @ 0x0220 0060

	D31 – D13			D12 – D8		
Bit Name	Reserved			FPGA Revision		
Function	Mask when reading			FPGA firmware revision level		
	D7	**D6**	**D5**	**D4 – D3**	**D2**	**D1 – D0**
Bit Name	Reserved	Mezzanine Bi-FIFO Reset	Reserved	Board Speed	VIM Module Present	Processor ID
Function	Mask read	0 = see Section 3.6.7.2 1 = Reset Complete	Mask read	('C6x, FIFOs) 00 = 200 MHz, 100 MHz 01 = 250 MHz, 62.5 MHz 10 = 300 MHz, 75 MHz 11 = 200 MHz, 50 MHz	0 = No VIM 1 = VIM	00 = Processor A 01 = Processor B 10 = Processor C 11 = Processor D
All bits default to the board configuration state at power up and reset.						

3.6.7.1 FPGA Revision D12 to D8

These five bits indicate the revision level for the board's FPGA firmware.

NOTE: On boards with FPGA firmware revision 1 or greater, four additional DSP interrupt control registers are provided, Sections 3.6.10 through 3.6.13.

3.6.7.2 Mezzanine Bi–FIFO Reset D6

This bit indicates the reset state of this node's Mezzanine Bi–FIFO. When you set the Mezzanine Bi–FIFO Reset (bit D4 in Miscella–neous Control Register, Section 3.6.8.5) to logic '1', which places the FIFO in reset mode, this status bit will read as logic '0' for approxi–mately four FIFO clock cycles. When the FIFO completes its reset cycle, this status bit changes to logic '1' indicating that reset is complete and you can clear the reset (clear bit D4, Section 3.6.8.5). When you clear the reset bit, this status bit clears to logic '0'.

3.6 Master DSP Control Registers (continued)

3.6.7 Board Status Register (continued)

3.6.7.3 Board Speed D4 & D3

These two bits indicate the 'C6x processor clock speed (the rated speed of the 'C6x processors and option installed on your board), and the FIFOs' speed. The settings for these two bits are:

00 – 200 MHz (Option 304, 'C6202 & 'C6203), 100 MHz FIFOs
01 – 250 MHz (Option 306, 'C6202 only), 62.5 MHz FIFOs
10 – 300 MHz (Option 308, 'C6203 only), 75 MHz FIFOs
11 – 200 MHz (Option 304, 'C6202 & 'C6203), 50 MHz FIFOs

3.6.7.4 VIM Module Present D2

This bit indicates whether or not a VIM module is installed on this DSP node's VIM interface. When read as logic '1', a VIM module is installed. When read as logic '0', no VIM module is present.

3.6.7.5 Processor ID D1 & D0

These two bits indicate the DSP Node identity of this Master processor (A, B, C, or D). The settings for these two bits are:

00 – Processor A
01 – Processor B
10 – Processor C
11 – Processor D

3.6 Master DSP Control Registers (continued)

3.6.8 Miscellaneous Control Register

This register contains control bits for several board resources. Each Master 'C6x processor has a unique copy of this register.

The following table shows the layout of this register. The subsections following this table describe each register bit.

Table 3–24: Miscellaneous Control Register (MCR)
R/W @ 0x0220 0064

	D31 – D10	D9	D8
Bit Name	Reserved	Slave Interrupt Request*	VIM Module Clock
Function	Write with zeros, Mask when reading	0 = Inactive 1 = Interrupt	0 = Div. by 4 1 = Div. by 8

	D7	D6	D5	D4	D3	D2	D1 – D0
Bit Name	i960 Interrupt Request	VIM Module Reset	Reserved	Mezzanine Bi-FIFO Reset	XBus XX Bi-FIFO Reset	Global Bus FIFO Reset	Reserved
Function	0 = Inactive 1 = Interrupt	0 = Normal 1 = Reset	Write zeros, Mask read	0 = Normal 1 = Reset	0 = Normal 1 = Reset	0 = Normal 1 = Reset	Write zeros, Mask read

* Model 4293 only
All bits default to the logic '0' state at power up and reset.

3.6.8.1 Slave Interrupt Request (Model 4293 only) D9

This bit allows the Master DSP to assert an interrupt (**EXT_INT4**) to its associated Slave DSP (Section 4.5.1). Set the bit to logic '1' to assert this interrupt to that processor—the interrupt is asserted when the bit transitions from logic '0' to logic '1'. This interrupt bit must be cleared by the Slave DSP (see Section 4.5.1).

3.6.8.2 VIM Module Clock D8

This bit selects the clock output to this node's VIM module, relative to the processor's clock (see Section 3.6.7.3 for the processor's clock speed). Clear this bit to logic '0' to divide the processor clock speed by 4. Set this bit to logic '1' to divide the processor clock by 8. The latter divider (8) may be needed for Model 4292/4293 boards with 250 or 300 MHz processor clocks, since many existing VIM modules have a maximum rated speed of 50 MHz.

3.6 Master DSP Control Registers (continued)

3.6.8 Miscellaneous Control Register (continued)

3.6.8.3 i960 Interrupt Request D7

The rising edge of this bit sets the interrupt flag bit in the i960 processor's DSP Interrupt Flag Register (Section 5.5.9) for the applicable Master DSP. An i960 interrupt will occur if it is enabled in the i960 DSP Interrupt Enable Register (Section 5.5.3). This interrupt is reserved for use by SwiftNet—if SwiftNet is not used, it is available for general use.

You must toggle this bit from logic '0' to logic '1' to set the interrupt flag bit in the i960 register to logic '1'. Note that when you clear this bit to logic '0', the i960 interrupt flag bit remains latched until you clear it by writing logic '1' to that bit, see Section 5.5.9.

3.6.8.4 VIM Module Reset D6

This bit is the reset for the VIM module installed on this node's VIM interface. Set this bit to logic '1' to hold the VIM module (not the 4292/4293 board) in its reset state. Clear this bit to logic '0' to return the VIM to normal operation.

3.6.8.5 Mezzanine Bi–FIFO Reset D4

This bit is the reset for this node's Mezzanine Bi–FIFO. Set this bit to logic '1' to hold the FIFO in its reset state. Clear this bit to logic '0' to return the FIFO to normal operation. After a reset, all data in the FIFO is erased and the FIFO flags are reset (not active).

3.6.8.6 XBus XX Bi–FIFO Reset D3

This bit is the reset for this DSP's XBus Inter–Processor XX Bi–FIFO. (The DSP connected to the other side of the XBus I/P YY Bi–FIFO controls its Reset—see Figure 9–1, Section 9.1, for the Inter–Processor Bi–FIFO XX/YY connections.) Set this bit to logic '1' to hold the XBus I/P XX Bi–FIFO in its reset state. Clear this bit to logic '0' to return the FIFO to normal operation. After a reset, all data in the FIFO is erased and the FIFO flags are reset (not active).

3.6.8.7 Global Bus FIFO Reset D2

This bit is the reset for this node's Global Bus FIFO. Set this bit to logic '1' to hold the FIFO in its reset state. Clear this bit to logic '0' to return the FIFO to normal operation. After a reset, all data in the FIFO is erased and the FIFO flags are reset (not active).

Refer to Section 9.2 for more information on resetting the FIFOs.

3.6 Master DSP Control Registers (continued)

3.6.9 General Purpose Semaphore Register

The General Purpose Semaphore Register provides three semaphores that allow any Master 'C6x processor or the i960 processor to request and monitor ownership of common resources on the Model 4292/4293. Each Master 'C6x processor has a unique copy of this register (see Section 5.5.7 for the i960 processor copy of this register).

Each semaphore has two bits: one bit to request ownership of the semaphore (Request*n*) by the processor, and one read–only bit that grants ownership of the semaphore (Grant*n*) by the board's arbitration logic.

The following table shows the bit layout of the semaphore register. The subsections following this table describe each set of bits.

Table 3–25: General Purpose Semaphore Register (GPSR)
R/W @ 0x0220 0068

	D31 – D6	D5	D4	D3	D2	D1	D0
Bit Name	Reserved	Grant2	Grant1	Grant0	Request2	Request1	Request0
Function	Write with zeros, Mask when reading	Read only: 1 = Semaphore is owned 0 = Semaphore is available			Read/Write: 1 = Request ownership of Semaphore 0 = Release the Semaphore		
All bits default to the logic '0' state at power up and reset.							

3.6.9.1 Grant2, Grant1, Grant0 D5, D4, & D3

These read–only bits indicate the status of each semaphore. When read as logic '1', ownership of this semaphore has been granted to this processor by the board's arbitration logic. When read as logic '0', this semaphore is available.

3.6.9.2 Request2, Request1, Request0 D2, D1, & D0

These bits request ownership of each semaphore. Set a bit to logic '1' to request ownership of that semaphore. Clear the bit to logic '0' to release the semaphore for ownership by another processor.

3.6 Master DSP Control Registers (continued)

3.6.9 General Purpose Semaphore Register (continued)

Two of the semaphores are pre−assigned to global resources, as follows:

- Semaphore 0 is assigned to the Universe II VMEbus Master.
- Semaphore 1 is assigned to the Global SDRAM.
- Semaphore 2 is user−defined—you may use this semaphore for requesting and monitoring use of any resource.

To use a semaphore, a processor first requests ownership of the semaphore by setting the 'Request' bit (D2 to D0) to logic '1'.

- If the semaphore is not in use, the associated 'Grant' bit (D5 to D3) is set to logic '1' by the board's arbitration logic, and the requesting processor now 'owns' the semaphore.
- If the semaphore is already 'owned' by another processor, the Grant bit will remain logic '0' until the processor that owns it releases it.

After using the resource, the owning processor must clear the applicable 'Request' bit to logic '0' to release the semaphore for use by another processor.

3.6 Master DSP Control Registers (continued)

3.6.10 'C6x External Interrupt Mode Register

NOTE: This register is available only on Model 4292/4293 boards with FPGA firmware revision 1 or greater (see Section 3.6.7.1).

The External Interrupt Mode Register allows you to select the type of interrupt to use for each interrupt source of the four 'C6x processor external interrupts (**EXT_INT4**, **EXT_INT5**, **EXT_INT6**, and **EXT_INT7**). Each Master 'C6x processor has a separate copy of this register.

This register contains mode bits that select the source for each interrupt in the 'C6x processor External Interrupt Enable Registers (Section 3.6.6). Each mode bit sets the source for all four external interrupt registers. The source can be either the associated miscellaneous interrupt status bit (Section 3.6.13) or the associated miscellaneous interrupt flag bit (Section 3.6.12).

The following table shows this register's bit layout. The paragraphs following the table describe the register bits.

Table 3–26: External Interrupt Mode Register (EIMR)
R/W @ 0x0220 006C

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Write with zeros, Mask when reading	0 = Use Status 1 = Use Flag				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	0 = Use Status 1 = Use Flag						Write with zeros, Mask when reading	

* Model 4293 only
All bits default to the logic '0' state at power up and reset.

Set a bit to logic '0' to route the associated status register bit (Section 3.6.13) to the 'C6x processor external interrupt. Set a bit to logic '1' to route the associated flag register bit (Section 3.6.12) to the 'C6x processor external interrupt.

The selected interrupt sources must be enabled using the applicable External Interrupt Enable Register (Section 3.6.6) to assert an interrupt to the 'C6x processor. See Figure 3–2 on the next page for illustration of the 'C6x external interrupt flow, showing the DSP registers that control this flow. As indicated in this flow, the OR'ed results of the enabled 'C6x interrupts are routed to the External Interrupt Status Register (Section 3.6.11) and to the 'C6x processor external interrupt inputs.

3.6 Master DSP Control Registers (continued)

3.6.10 'C6x External Interrupt Mode Register (continued)

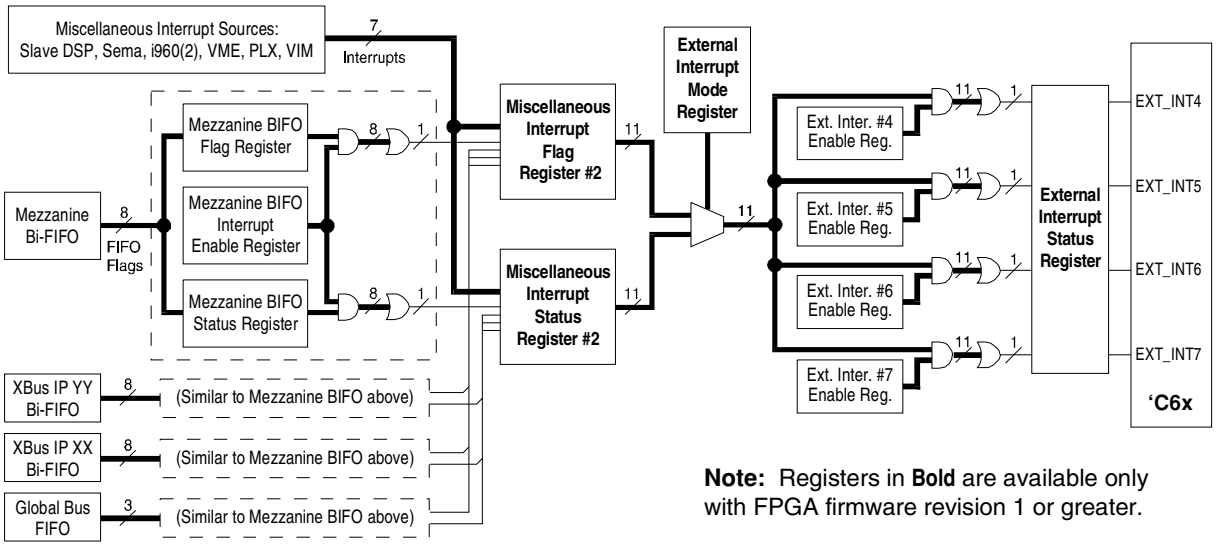


Figure 3–2: 'C6x External Interrupt Flow

3.6.11 'C6x External Interrupt Status Register

NOTE: This register is available only on Model 4292/4293 boards with FPGA firmware revision 1 or greater (see Section 3.6.7.1).

The External Interrupt Status Register contains read–only status bits for the four 'C6x processor external interrupts (**EXT_INT4**, **EXT_INT5**, **EXT_INT6**, and **EXT_INT7**). Each Master 'C6x processor has a separate copy of this register.

Each status bit is an OR'ed result of the enabled sources for this 'C6x external interrupt (see Section 3.6.10 and Figure 3–2 above). When read as logic '1', an interrupt is currently active. When read as logic '0', all interrupts are inactive or not enabled. The following table shows this register's bit layout.

Table 3–27: External Interrupt Status Register (EISR)
R.O. @ 0x0220 0070

	D31 – D4	D3	D2	D1	D0
Bit Name	Reserved	Ext Int 7	Ext Int 6	Ext Int 5	Ext Int 4
Function	Mask when reading	0 = Interrupt not active 1 = Interrupt active			
All bits default to the logic '0' state at power up and reset.					

3.6 Master DSP Control Registers (continued)

3.6.12 Miscellaneous Interrupt Flag Register #2

NOTE: This register is available only on Model 4292/4293 boards with FPGA firmware revision 1 or greater (see Section 3.6.7.1).

This register contains flag bits for several miscellaneous interrupt sources. Each Master 'C6x processor has a separate copy of this register. The following table shows this register's bit layout. The paragraphs and subsections following the table describe the register bits.

Table 3–28: Miscellaneous Interrupt Flag Register 2 (MIFR2)
R/Clr @ 0x0220 0074

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Write with zeros, Mask when reading	**READ**: 0 = No Interrupt; 1 = Interrupt latched **WRITE**: 0 = Normal operation; 1 = Clear latch				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	**READ**: 0 = No Interrupt 1 = Interrupt latched **WRITE**: 0 = Normal 1 = Clear latch		**READ**: 0 = No Interrupt 1 = Interrupt latched **WRITE**: 0/1 = no action				Write with zeros, Mask when reading	

* Model 4293 only

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

NOTE: This register may be selected as the source for any interrupt bit of the four 'C6x processor external interrupts, see Section 3.6.10.

Flag bits D12 through D6 are latched versions of the associated bits of the Miscellaneous Interrupt Status Register #2 (Section 3.6.13). Each flag bit is set when that interrupt condition transitions from inactive (logic '0') to active (logic '1'). Since these interrupting conditions may be transient, they are latched in this register so that the processor can determine the cause.

Flag bits D12 through D6 are read/clear bits. When read as logic '1', an interrupt condition has been latched; when read as logic '0', the interrupt condition is inactive. Each flag bit remains latched until you clear it by writing logic '1' to that bit. When you write logic '1' to a latched flag bit, that bit clears regardless of the state of the interrupt source and will not be set again until the next interrupt condition transition from inactive to active (status bit transitions from logic '0' to logic '1'). When you clear any bits, the associated bits of the Miscellaneous Interrupt Flag Register (Section 3.6.4) are also cleared.

3.6 Master DSP Control Registers (continued)

3.6.12 Miscellaneous Interrupt Flag Register #2 (continued)

Flag bits D5 through D2 are OR'ed results of all bits of each respective FIFO Flag Register (Section 3.6.2). Each bit of the FIFO Flag Register is included in the OR'ed result only when the associated bit is enabled in the respective FIFO Interrupt Enable Register (Section 3.6.1), as illustrated in the figure below.

Flag bits D5 through D2 are read bits. When read as logic '1', a FIFO boundary flag condition has been latched in the respective FIFO Flag Register. When read as logic '0', all boundary flag conditions are cleared or not enabled. Writing to these bits has no action. Each flag bit remains latched until you clear the respective FIFO Flag Register (Section 3.6.2).

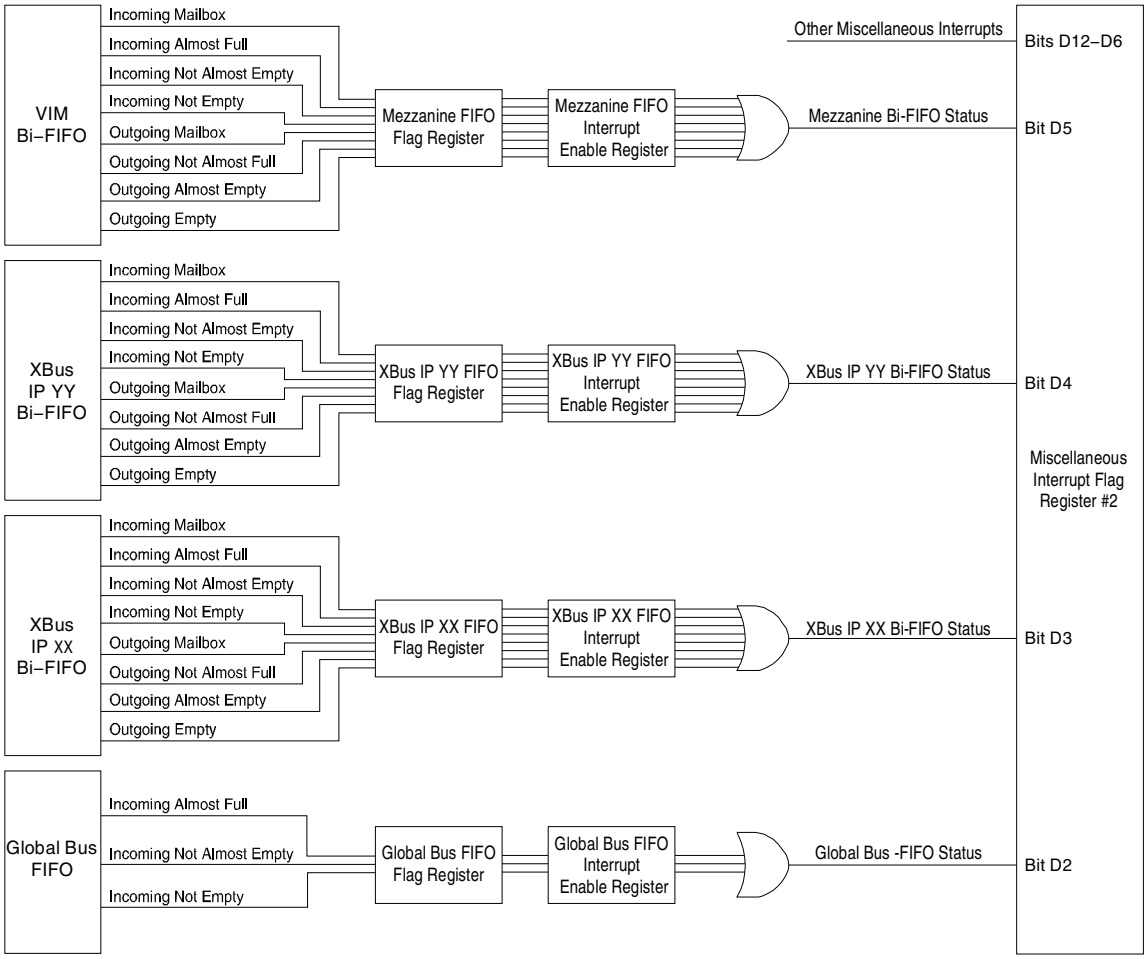


Figure 3–3: FIFO Interrupt Flags Flow

3.6 Master DSP Control Registers (continued)

3.6.12 Miscellaneous Interrupt Flag Register #2 (continued)

3.6.12.1 Slave DSP (Model 4293 only) D12

This bit is the interrupt signal from the Slave DSP (Section 4.5.3.1). When read as logic '1', an interrupt has been received. When read as logic '0', the interrupt is not active.

When you clear the latched interrupt flag by writing logic '1' to this bit position, an interrupt is asserted back to the associated Slave DSP on its External Interrupt #5 line (Section 4.5.3.1).

3.6.12.2 Semaphore D11

This bit is the OR'ed result of the three Semaphore Register Grant bits (Section 3.6.9). When read as logic '1', a semaphore grant bit is active (at least one semaphore is currently 'owned' by a processor). When read as logic '0', the semaphore is available (not owned by any processor).

3.6.12.3 i960 PINT D10

This bit is the PCI interrupt line, P_INTx, from the i960 processor (Section 8.4). When read as logic '1', this interrupt signal is active. When read as logic '0', the interrupt is not active. The i960 P_INTx interrupt lines are routed to each DSP Node as follows:

P_INTA to DSP Node A
P_INTB to DSP Node C
P_INTC to DSP Node D
P_INTD to DSP Node B

3.6.12.4 i960 SwiftNet D9

This bit is the i960 SwiftNet interrupt from the i960 processor's DSP Interrupt Request Register (Section 5.5.1). When read as logic '1', an interrupt is active. When read as logic '0', the interrupt is not active.

3.6.12.5 VME D8

This bit is the interrupt LINT0 from the VME interface controller (Tundra Universe II). When read as logic '1', this interrupt signal is active. When logic '0', the interrupt is not active.

3.6 Master DSP Control Registers (continued)

3.6.12 Miscellaneous Interrupt Flag Register #2 (continued)

3.6.12.6 PLX D7

This bit is the interrupt LINT from the PCI bus master (PLX PCI 9054) for this DSP node. When read as logic '1', this interrupt signal is active. When read as logic '0', the interrupt is not active.

3.6.12.7 VIM D6

This bit is the VIM interrupt DBINT from the installed VIM module for this node. When read as logic '1', the VIM interrupt signal is active. When read as logic '0', the interrupt is not active.

3.6.12.8 Mezzanine BI–FIFO D5

This bit is an OR'ed result from the Mezzanine Bi–FIFO Interrupt Flag Register bits (Section 3.6.2), see Figure 3–3. When read as logic '1', at least one of the enabled Mezzanine Bi–FIFO's flag conditions is active. When read as logic '0', the Mezzanine Bi–FIFO flag conditions are cleared or not enabled.

3.6.12.9 XBus I/P YY BI–FIFO D4

This bit is an OR'ed result from the XBus Inter–Processor YY Bi–FIFO Interrupt Flag Register bits (Section 3.6.2), see Figure 3–3. When read as logic '1', at least one of the enabled XBus Inter–Processor YY Bi–FIFO's flag conditions is active. When read as logic '0', the XBus Inter–Processor YY Bi–FIFO flag conditions are cleared or not enabled.

3.6.12.10 XBus I/P XX BI–FIFO D3

This bit is an OR'ed result from the XBus Inter–Processor XX Bi–FIFO Interrupt Flag Register bits (Section 3.6.2), see Figure 3–3. When read as logic '1', at least one of the enabled XBus Inter–Processor XX Bi–FIFO's flag conditions is active. When read as logic '0', the XBus Inter–Processor XX Bi–FIFO flag conditions are cleared or not enabled.

3.6.12.11 Global Bus FIFO D2

This bit is an OR'ed result from the Global Bus FIFO Interrupt Flag Register bits (Section 3.6.2), see Figure 3–3. When read as logic '1', at least one of the enabled Global FIFO's flag conditions is active. When read as logic '0', the Global FIFO flag conditions are cleared or not enabled.

3.6 Master DSP Control Registers (continued)

3.6.13 Miscellaneous Interrupt Status Register #2

NOTE: This register is available only on Model 4292/4293 boards with FPGA firmware revision 1 or greater (see Section 3.6.7.1).

This read–only register contains status bits for several miscellaneous interrupt sources. Each Master 'C6x processor has a separate copy of this register.

The following table shows this register's bit layout. The paragraphs following the table describe the register bits.

Table 3–29: Miscellaneous Interrupt Status Register 2 (MISR2)
R.O. @ 0x0220 0078

	D31 – D13	D12	D11	D10	D9	D8
Bit Name	Reserved	Slave DSP*	Semaphore	i960 PINT	i960 SwiftNet	VME
Function	Mask when reading	0 = Interrupt not active 1 = Interrupt active				

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	PLX	VIM	Mezzanine BI-FIFO	XBus I/P YY BI-FIFO	XBus I/P XX BI-FIFO	Global Bus FIFO	Reserved	
Function	0 = Interrupt not active 1 = Interrupt active						Mask when reading	

* Model 4293 only
When reset, including power–up, the state of this register is unknown.

NOTE: This register may be selected as the source for any interrupt bit of the four 'C6x processor external interrupts, see Section 3.6.10.

Status bits D12 through D6 are buffered versions of the interrupting sources and may be monitored by software to verify that an interrupting condition has been properly cleared. When one of these bits reads as logic '1', the inter–rupt condition is currently active. When read as logic '0', the interrupt is not active. Refer to Sections 3.6.12.1 through 3.6.12.7 for description of the inter–rupt sources for each of these bits.

When the interrupting source for bits D12 through D6 transitions from inac–tive to active, the associated flag bit is set in the Miscellaneous Interrupt Flag Register #2 (Section 3.6.12). The miscellaneous interrupt status and flag bits represent the same interrupt sources—the only difference is when the inter–rupt source clears (goes not active), the status bit clears automatically in this register, while the flag bit must be cleared by writing to the appropriate bit in the flag register (Section 3.6.12).

3.6 Master DSP Control Registers (continued)

3.6.13 Miscellaneous Interrupt Status Register #2 (continued)

Status bits D5 through D2 are OR'ed results of all bits of each respective FIFO Status Register (Section 3.6.3). Each bit of a FIFO Status Register is included in the OR'ed result only when the associated bit is enabled in the respective FIFO Interrupt Enable Register (Section 3.6.1), as illustrated in the figure below.

When any bit D5 through D2 reads as logic '1', a FIFO boundary flag condition is currently active in the respective FIFO Status Register. When read as logic '0', all boundary flag conditions are inactive or not enabled. (Note that the associated bits in the Miscellaneous Interrupt Flag Register #2, Section 3.6.12, are OR'ed results from the respective FIFO Flag Registers, and are not associated with these Miscellaneous Interrupt Status Register #2 bits.)

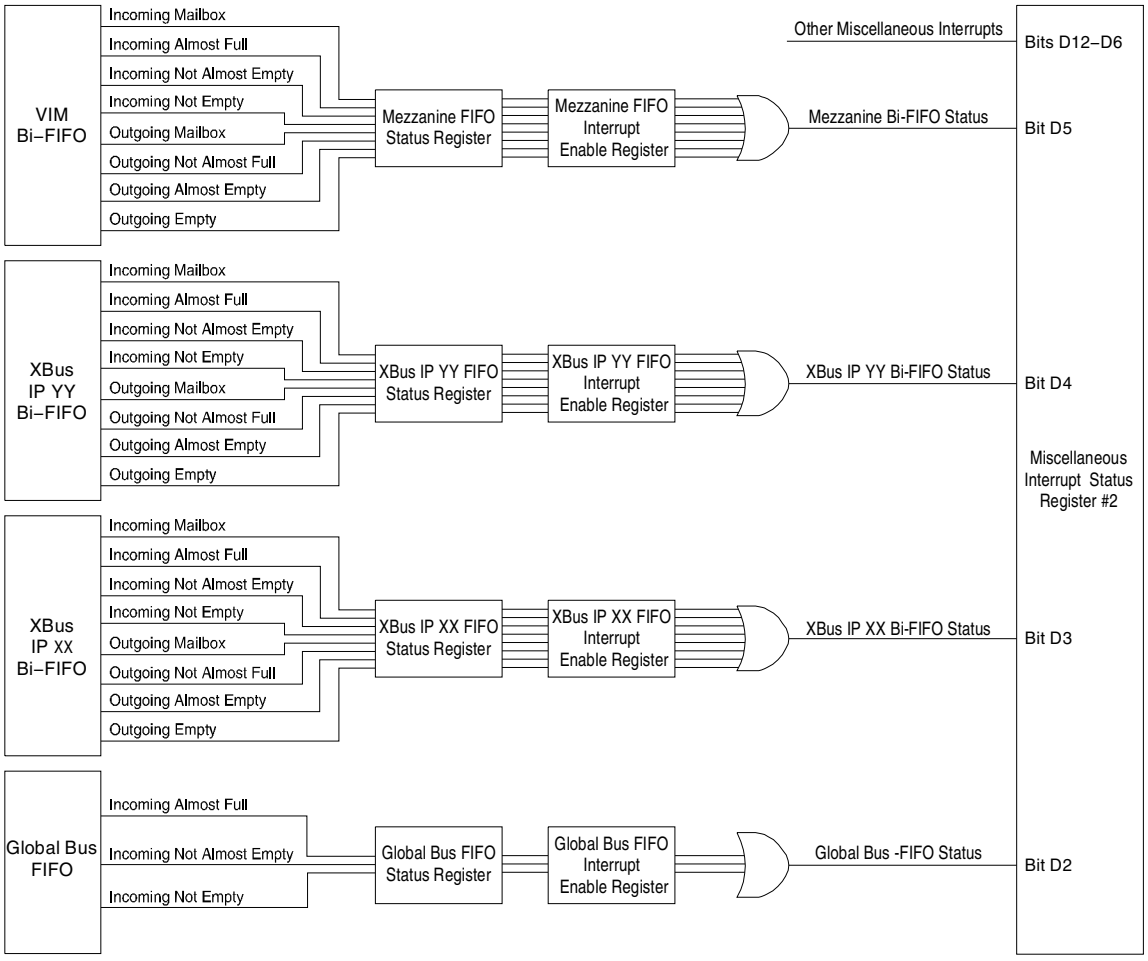


Figure 3–4: FIFO Interrupt Status Flow

Chapter 4: Slave 'C6x Processor Memory Maps and Resources



This Chapter is applicable only for the Model 4293.

4.1 Overview

This chapter presents the memory map and operation of board resources available to each Slave 'C6x processor. In addition to internal resources, each Slave 'C6x processor has direct access to its External Memory Interface (EMIF) resources. In addition to this chapter, refer to Chapter 10 for Slave 'C6x FLASH and boot firmware operation.

4.2 'C6x Processor Internal Resources

The Slave 'C6x processor internal resources include program and data memories and on−chip peripherals. A memory map of these resources is provided in Texas Instruments TMS320C6000 Peripherals Reference Guide (at Model 4293 boot, MAP 0 is set as the first section of the memory map).

The size of the internal program and data memories depends on the 'C6x processor type on your board, as follows:

Processor	Internal Program RAM	Internal Data RAM
TMS320C6202	256 kBytes	128 kBytes
TMS320C6203	384 kBytes	512 kBytes

Memory−mapped registers on the processor's internal peripheral bus configure the External Memory Interface (EMIF), DMA, Expansion Bus (XBus), Multichannel Buffered Serial Port (McBSP), Timers, Interrupt Selector, and Peripheral Power−Down.

- The EMIF, DMA, XBus, Interrupt Selector, and Peripheral Power−Down registers are initialized at boot time. XBus Host Port registers allow an external XBus Host to read/write any address on the Slave DSP memory map.

- For operation of the 'C6x Serial Ports (McBSP) and Timers, refer to TI TMS320C6000 Peripherals Reference Guide.

The 'C6x XBus Global Control Register (XBGC) configures XBus parameters. For a Slave 'C6x processor, bit 11 (XARB) of this register is set to logic '0.' Refer to TI TMS320C6000 Peripherals Reference Guide for description of this register.

4.3 Slave DSP Memory Map

Each Slave 'C6x processor has memory–mapped access to its internal resources, its local SDRAM, and its FLASH memory. The following table shows the resources accessible from each Slave DSP, the path to each resource (internal, EMIF), and the memory addresses of that resource. The referenced sections provide information on 'C6x operation of each resource.

Table 4–1: Slave DSP Memory Map

Resource	Path	Memory Address	Information
DSP Local SDRAM	EMIF	0x0000 0000 – 0x00FF FFFF	Section 4.4
DSP Local FLASH Memory		0x0100 0000 – 0x013F FFFF	Section 10.2
Internal Program RAM C6202 Internal Program RAM C6203	Internal	0x0140 0000 – 0x0143 FFFF 0x0140 0000 – 0x0145 FFFF	Section 4.2 and TI TMS320C6000 Peripherals Reference Guide [1]
Reserved		0x0146 0000 – 0x017F FFFF	
Internal Peripheral Bus EMIF Registers		0x0180 0000 – 0x0183 FFFF	
Internal Peripheral Bus DMA Registers		0x0184 0000 – 0x0187 FFFF	
Internal Peripheral Bus XBus Registers		0x0188 0000 – 0x018B FFFF	
Internal Peripheral Bus McBSP 0 Registers		0x018C 0000 – 0x018F FFFF	
Internal Peripheral Bus McBSP 1 Registers		0x0190 0000 – 0x0193 FFFF	
Internal Peripheral Bus Timer 0 Registers		0x0194 0000 – 0x0197 FFFF	
Internal Peripheral Bus Timer 1 Registers		0x0198 0000 – 0x019B FFFF	
Internal Peripheral Bus Interrupt Registers		0x019C 0000 – 0x019C 01FF	
Internal Peripheral Bus Power–Down Register		0x019C 0200 – 0x019C FFFF	
Reserved		0x01A0 0000 – 0x01A3 FFFF	
Internal Peripheral Bus McBSP 2 Registers		0x01A4 0000 – 0x01A7 FFFF	
Reserved	–	0x01A8 0000 – 0x01FF FFFF	–
Clear Master–to–Slave Interrupt	EMIF CE2	0x0200 0000 – 0x02FF FFFF	Section 4.5
Interrupt to Master/i960	EMIF CE3	0x0300 0000 – 0x03FF FFFF	
Reserved	–	0x0400 0000 – 0x7FFF FFFF	–
Internal Data RAM C6202 Internal Data RAM C6203	Internal	0x8000 0000 – 0x8001 FFFF 0x8000 0000 – 0x8007 FFFF	Section 4.2 and TI Per. Ref. Guide [1]
Reserved	–	0x8008 0000 – 0xFFFF FFFF	–

1 – Memory maps of the internal 'C6x resources are provided in Texas Instruments TMS320C6000 Peripherals Reference Guide.

4.4 Slave DSP Local SDRAM Operation

Each Slave 'C6x processor is connected, on its EMIF bus, to an 8–MByte local SDRAM (16–MByte SDRAM with Option 340), which is used for 'C6x boot operation at reset or power–up of the Model 4293 (see Section 10.4). Each 'C6x processor can access its local SDRAM starting at memory address **0x0000 0000**.

The first 64 kBytes of each local SDRAM is reserved for use by the default 'C6x boot firmware (Section 10.4). ***Loading user code into this area or accessing data in this area can cause unpredictable behavior.***



Data written from a Master processor to a Slave processor's SDRAM should originate in the Master's internal ID MEM.

Data read from a Slave processor's SDRAM by a Master processor should be transferred to the Master's internal ID MEM.

4.5 Slave DSP Interrupt Operation

Each Slave 'C6x processor can assert interrupts from certain memory writes, and can respond to interrupts received on its external interrupt lines: **EXT_INT4**, **EXT_INT5**, or **EXT_INT6**. The following figure illustrates these interrupt paths. The subsections on the next page table describe these interrupts.

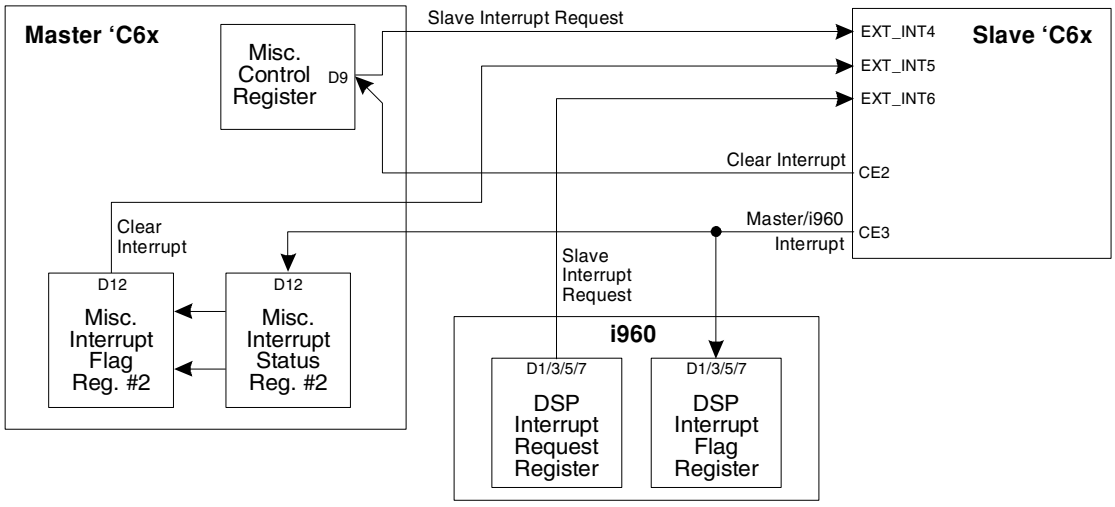


Figure 4–1: Model 4293 Slave DSP Interrupt Paths

Refer to the Texas Instruments TMS320C6000 Peripherals Reference Guide for description of 'C6x external interrupt programming.

4.5 Slave DSP Interrupt Operation (continued)

4.5.1 Responding to Master DSP Interrupt

Each Master DSP can assert an interrupt to its associated Slave DSP using bit D9 in its Miscellaneous Control Register (see Section 3.6.8). The Slave DSP receives this interrupt on its External Interrupt #4 line (**EXT_INT4**).

Following response to this Master DSP interrupt, the Slave DSP must clear the interrupt in the Master DSP by writing to any memory address in the range **0x0200 0000** to **0x02FF FFFF** (EMIF CE2 address space).

4.5.2 Responding to i960 Interrupt

The i960 I/O processor can assert an interrupt to any Slave DSP using bits D7 (for DSP H), D5 (for DSP G), D3 (for DSP F), and D1 (for DSP E) in its DSP Interrupt Request Register (see Section 5.5.1). Each Slave DSP receives this interrupt on its External Interrupt #6 line (**EXT_INT6**).

4.5.3 Asserting Slave DSP Interrupt

A Slave 'C6x processor can assert an interrupt to the associated Master DSP and to the i960 I/O processor, by writing to any memory address in the range **0x0300 0000** to **0x03FF FFFF** (EMIF CE3 address space).

4.5.3.1 Slave Interrupt to Master DSP

When the Slave writes to CE3 space, the 'Slave DSP' status bits in the associated Master DSP, bit D6 of the Miscellaneous Interrupt Status Register and bit D12 of Miscellaneous Interrupt Status Register #2 (see Sections 3.6.5 and 3.6.13), toggle briefly to logic '1'. The Master DSP can enable these status bits as an external interrupt with its External Interrupt Enable Registers (see Section 3.6.6).

The corresponding latched bits in the Master DSP Miscellaneous Interrupt Flag Registers (see Sections 3.6.4 and 3.6.12) are also set to logic '1' from the Slave CE3 write. When the Master DSP clears this latched bit (see Section 3.6.12.1), an interrupt is asserted to the Slave on its External Interrupt #5 line (**EXT_INT5**), which can be used to verify if the Master has responded to its interrupt.

4.5.3.2 Slave Interrupt to i960

In the i960, when the Slave writes to CE3 space, the appropriate bit (D7 for DSP H, D5 for DSP G, D3 for DSP F, D1 for DSP E) of the DSP Interrupt Flag Register (see Section 5.5.9) is set to logic '1'. The i960 can enable these bits as an interrupt using the associated bit in the DSP Interrupt Enable Register (see Section 5.5.3).

Chapter 5: I/O Processor Memory Maps and Resources

5.1 Overview

The Model 4292/4293 supports two PCI bus structures, the Primary and Secondary PCI Buses. An Intel i960RN I/O Processor acts as the bridge between the two PCI buses, and functions as a memory and DMA controller, and as data executive for the four DSP nodes. The Primary PCI Bus provides the i960 access to the RS−232 and Ethernet interfaces (Intel 82559), the PMC site, and the VMEbus interface (Tundra Universe II). The Secondary PCI Bus provides access to each DSP node's Expansion Bus, through the four PCI Bus Masters (PLX PCI 9054s).

In addition to the two PCI buses, the i960 supports the board's Global Bus, which provides access to the Global SDRAM and to four Global Bus FIFOs, one connected to each DSP node's Expansion Bus. The i960 also has an 8−bit local control bus, supporting a 4−MByte FLASH memory and a test bus controller (TI SN74ACT8990). A voltage and temperature monitor (ADM1024) and the board's ID EEPROM are accessible to the i960 on its I^2C bus.

This chapter presents the memory map and operation of board resources available to the i960 I/O processor. In addition to the resource operations described in this chapter, refer to Chapter 8 for i960 interrupt operation, Chapter 9 for i960 FIFO operation, and Chapter 10 for i960 FLASH and boot firmware operation.

5.2 i960 Processor Internal Resources

The i960 I/O processor has two sets of internal memory−mapped registers. One set is used to control the i960 core processor functions; the other set is used to control the i960 integrated peripherals. The memory−mapped peripheral registers support the DMA controller, memory controller, PCI and peripheral interrupt controller, messaging unit, internal arbitration unit, PCI to PCI bridge unit, PCI address translation unit, I^2C bus interface unit, performance monitoring unit, and the application accelerator unit. Descriptions of these i960 internal resources and registers are provided in the Intel i960® RM/RN I/O Processor Developer's Manual. A memory map of these registers is provided in Appendix C of that manual.

The i960 Messaging Unit (MU) has four messaging mechanisms that allow the i960 to communicate with PCI resources through command passing and interrupt generation. The four mechanisms are: Message Registers, Doorbell Registers, Circular Queues, and Index Registers. Description of the MU is provided in the Intel i960® RM/RN I/O Processor Developer's Manual. The i960 MU is accessible to the Master 'C6x processors (see Section 3.5.5), to a PCI bus device (see Section 6.5.3), and to an external VMEbus Master (see Section 7.3.2.4). In addition, the Command Processor application (see Chapter 12) provides several user functions that can be called using the i960 MU.

5.3 i960 Processor Memory Maps

The following table shows the memory–mapped resources accessible by the i960, the path to each resource, the i960 memory address, and a reference to more information.

Table 5–1: i960 Processor Memory Map

Resource	Path	i960 Memory Address	Information
i960 Internal Data RAM	Internal	0x0000 0000 – 0x0000 03FF	Section 5.2
Reserved		0x0000 0400 – 0x0000 0FFF	
i960 Peripheral Registers		0x0000 1000 – 0x0000 1FFF	
Reserved	–	0x0000 2000 – 0x0FFF FFFF	–
VMEbus Master Access	Primary PCI Bus	0x1000 0000 – 0x6FFF FFFF	Section 5.4.1
PMC Module Access RACEway Access (Option 005 only)		0x7000 0000 – 0x7FFF FFFF	Section 5.4.4
82559 Ethernet Controller		0x8020 0000 – 0x8020 0FFF	Section 5.4.2
TL16C550C UART		0x8040 0000 – 0x8040 0008	
Universe II Registers		0x8050 0000 – 0x8050 0FFF	Section 5.4.3
PXB Registers (Option 005 only)		0x8060 1000 – 0x8060 FFFF	Section 5.4.4
Reserved	–	0x8070 1000 – 0x87FF FFFF	–
Node A PLX PCI 9054 Registers	Secondary PCI Bus	0x8800 0000 – 0x8800 00FF	Section 5.4.8
Node A Host Port Registers (NP)		0x8801 0000 – 0x8801 FFFF	Section 5.4.7
Node B PLX PCI 9054 Registers		0x8803 0000 – 0x8803 00FF	Section 5.4.8
Node B Host Port Registers (NP)		0x8804 0000 – 0x8804 FFFF	Section 5.4.7
Node C PLX PCI 9054 Registers		0x8806 0000 – 0x8806 00FF	Section 5.4.8
Node C Host Port Registers (NP)		0x8807 0000 – 0x8807 FFFF	Section 5.4.7
Node D PLX PCI 9054 Registers		0x8809 0000 – 0x8809 00FF	Section 5.4.8
Node D Host Port Registers (NP)		0x880A 0000 – 0x880A FFFF	Section 5.4.7
Node A Host Port Registers (P)		0x8810 0000 – 0x8810 FFFF	Section 5.4.7
Node B Host Port Registers (P)		0x8811 0000 – 0x8811 FFFF	
Node C Host Port Registers (P)		0x8812 0000 – 0x8812 FFFF	
Node D Host Port Registers (P)		0x8813 0000 – 0x8813 FFFF	
Reserved	–	0x8814 0000 – 0x8FFF FFFF	–
PCI Bus I/O Space	Primary PCI Bus	0x9000 0000 – 0x9000 FFFF	Section 5.3.1
	Secondary PCI Bus	0x9001 0000 – 0x9001 FFFF	
Reserved	–	0x9002 0000 – 0x9FFF FFFF	–
i960 Messaging Unit Registers	Internal	0xA000 0000 – 0xA000 0FFF	Section 5.2
Global SDRAM	Global Bus, 32–bit mode	0xA000 1000 – 0xA0FF FFFF	Section 5.4.5
	Global Bus, 64–bit mode	0xA000 1000 – 0xA1FF FFFF	
Node A Global Bus FIFO	Global Bus, 32–bit mode *	0xA100 0000 – 0xA107 FFFF	Section 9.4.3
Node C Global Bus FIFO		0xA108 0000 – 0xA10F FFFF	
Node D Global Bus FIFO		0xA110 0000 – 0xA117 FFFF	
Node B Global Bus FIFO		0xA118 0000 – 0xA11F FFFF	
Reserved	–	0xA200 0000 – 0xFDFF FFFF	–
i960 Control Registers	8–bit Local Bus	0xFE00 0000 – 0xFE00 000A	Section 5.5
Test Bus Controller		0xFE00 0040 – 0xFE00 007F	Section 5.4.6
i960 FLASH Memory		0xFEC0 0000 – 0xFEFF FFFF	Section 10.3
i960 Core Processor Registers	Internal	0xFF00 0000 – 0xFFFF FFFF	Section 5.2

(NP) = Non–prefetched data access; (P) = Prefetched data access

* If PCB jumper **J10** is set for 64–bit data width (Section 2.2.4), the four Global Bus FIFOs are not available.

5.3 i960 Processor Memory Maps (continued)

5.3.1 i960 PCI Bus I/O Address Map

The i960 can access several PCI bus resources using PCI I/O–mapped addresses. The following table shows the i960 memory addresses for these resources on each PCI bus, and a reference to more information.

Table 5–2: i960 PCI Bus I/O Map

Resource	Path	i960 Memory Address	Information
82559 Ethernet Controller	Primary PCI Bus I/O	**0x9000 1000 – 0x9000 1FFF**	Section 5.4.2
Universe II Registers		**0x9000 2000 – 0x9000 2FFF**	Section 5.4.3
PMC Module Access *		**0x9000 3000 – 0x9000 3FFF**	Section 5.4.4
Node A PLX PCI 9054 Registers	Secondary PCI Bus I/O	**0x9001 0000 – 0x9001 00FF**	Section 5.4.8
Node B PLX PCI 9054 Registers		**0x9001 0300 – 0x9001 03FF**	
Node C PLX PCI 9054 Registers		**0x9001 0600 – 0x9001 06FF**	
Node D PLX PCI 9054 Registers		**0x9001 0900 – 0x9001 09FF**	

* Module dependent—I/O–mapped resources may not be present on all PCI devices.

These resources are also available using PCI memory–mapped addresses as indicated in the i960 Processor Memory Map, Table 5–1, prior page.

5.3.2 i960 I^2C Bus Memory Map

The board's voltage and temperature monitor (ADM1024) and ID EEPROM are accessible to the i960 on its I^2C bus. The following table shows the i960 I^2C bus address for these resources, and a reference to more information. Note that each device on the I^2C bus has a 7–bit binary address.

Table 5–3: i960 I^2C Bus Memory Map

Resource	Binary Address	Information
Voltage/Temperature Monitor	**0101100b**	Section 5.4.9
Board ID EEPROM	**1010xxxb** (where **xxx** selects one of eight 256–word blocks)	Section 5.4.10

5.4 i960 Resources Operation

The following subsections describe the operation of the resources available to the i960 processor.

5.4.1 VMEbus Master Access

When the Model 4292/4293 is a VMEbus Master, the i960 processor can access data and perform D16, D32, and D64 transfers in all relevant VMEbus address spaces. See Section 7.2 for more information on VMEbus Master operation. The i960 memory map for VMEbus Master access is provided below.

Table 5–4: VMEbus Master Memory Map

VMEbus Address Space (Address Modifiers)	i960 Memory Address	VMEbus Address
A16/D16 (Non–Privileged, 0x29)	**0x0001 0000 – 0x0001 FFFF**	**0x0000 – 0xFFFF**
A16/D32 (Non–Privileged, 0x29)	**0x0002 0000 – 0x0002 FFFF**	**0x0000 – 0xFFFF**
A24/D16 (Standard, Non–Privileged, Data, 0x39)	**0x0100 0000 – 0x01FF FFFF**	**0x00 0000 – 0xFF FFFF**
A24/D32 (Standard, Non–Privileged, Data, 0x39)	**0x0200 0000 – 0x02FF FFFF**	**0x00 0000 – 0xFF FFFF**
A32/D32 (Extended, Non–Privileged, Data, 0x09)	**0x1000 0000 – 0x6FFF FFFF**	**0x0000 0000 – 0x5FFF FFFF**
A32/D64 (Extended, Non–Privileged, Data, 0x09)	**0x0400 0000 – 0x0FFF FFFF**	**0x0400 0000 – 0x0FFF FFFF**

5.4.2 RS–232/Ethernet Interface

The Intel 82559 supports RS–232 and Ethernet input/output through the Model 4292/4293 front panel connector (Section 2.6.6). To transmit or receive Ethernet signals, read or write to the 82559 Control and Status registers. Refer to the Intel 82559 Fast Ethernet Multifunction PCI/CardBus Controller datasheet for description of these registers. The i960 can access the 82559 registers starting at memory address **0x8020 0000** (or, using PCI I/O access, at memory address **0x9000 1000**). To send or receive RS–232 serial data, read/write to the TL16C550C UART at memory address **0x8040 0000**.

NOTE: This resource is reserved for use exclusively by the i960 firmware.

5.4.3 Universe II Registers

The Tundra Universe II provides programmable registers for VME system configuration and control of Universe II operational characteristics: PCI Configuration Space, VMEbus Configuration and Status Registers, and Universe II Device Specific Status Registers. A reference listing of these registers is provided in Appendix E. Refer to the Tundra Universe II User Manual for detailed description of these registers. The i960 can access these registers starting at memory address **0x8050 0000** (or, using PCI I/O access, at memory address **0x9000 2000**).

5.4 i960 Resources Operation (continued)

5.4.4 PMC Module Access

The PCI Mezzanine Card (PMC) site accepts industry−standard PMC modules. If a PMC module is installed, the i960 can access the PMC local address space resources (I/O, memory, registers, DMA, etc.) starting at memory address **0x7000 0000** and, if PCI I/O access is implemented on the PMC module, at memory address **0x9000 3000**. Each PMC module is supplied with an Operating Manual that describes any programmable resources of that module. Refer to that manual for description of these resources.

When Option 005 RACEway is installed, this PMC module supports both outgoing (Master) and incoming (Slave) RACEway operations. The i960 can access the RACEway address space starting at memory address **0x7000 0000**, and can access the module's PXB++ chip internal configuration registers starting at memory address **0x8060 1000**. There is no PCI I/O−mapped access implemented on the Option 005 PMC module. Refer to Appendix D for description of Pentek's RACE++ PMC module, Option 005.

5.4.5 Global SDRAM

The i960 processor can access Global SDRAM starting at memory address **0xA000 1000**. The available size of this memory is either 16 or 32 MBytes, depending on the setting of PCB jumper **J10**, i960 memory/bus data width (Section 2.2.4). (Note that when this jumper is set for 64−bit data width, the Global SDRAM size is 32 MBytes and the Global Bus FIFOs are not available to the i960.)

The first 4 MBytes of Global SDRAM (**0xA000 1000 – 0xA040 FFFF**) are reserved for the default boot firmware use. The i960 processor initializes and uses this area of Global SDRAM during its boot sequence. Refer to Section 10.4 for description of the i960 boot sequence and Global SDRAM initialization.

5.4.6 Test Bus Controller

The Test Bus Controller (TBC), a TI SN74ACT8990, provides access to each 'C6x processor's JTAG/XDS emulator ports. This device interfaces the i960 to a serial test bus on the Model 4292/4293, compliant with IEEE Standard 1149.1 (JTAG). The test bus may also be controlled by an external XDS510 Emulator (using the board's JTAG/XDS connector, Section 2.3.3) as an alternative to JTAG, but use of these functions is mutually exclusive—that is, when an XDS510 emulator connector is inserted, the TBC is switched off.

The i960 can access the SN74ACT8990 control registers on its 8−bit local bus, starting at memory address **0xFE00 0040**. Each i960 data word on this bus is only one byte (8 bits). Since the ACT8990 has a 16−bit interface, read/write two bytes for every word from/to the Test Bus Controller.

5.4 i960 Resources Operation (continued)

5.4.7 DSP Expansion Bus Host Port

Through the four PLX PCI 9054s, the i960 has access to two registers on each DSP's Expansion Bus (XBus) Host Port: the XBus Data Register (XBD), and the XBus Internal Slave Address Register (XBISA). Refer to the Texas Instruments TMS320C6000 Peripherals User Guide for description of these registers. The XBISA and XBD registers allow the i960 to access any address on each DSP's memory map (Sections 3.3 and 4.3), and support both prefetched and non–prefetched data access to those addresses.

- The XBISA controls the memory location (in the DSP) being accessed by the i960 through the DSP XBus Host Port. The i960 can access each DSP's XBISA registers at the memory addresses listed in Table 5–5, below.
- The XBD contains the data that was read from the memory accessed by the XBus Host Port if the current access is a read, or the data that is written to the memory if the current access is a write. The i960 can access each DSP's XBD registers at the memory addresses listed in Table 5–5, below.

To read or write to a DSP resource, first write the DSP memory address into the XBISA, then read or write the data at that address using the XBD. The i960 can perform single 32–bit, 16–bit, or 8–bit accesses to the XBD. Bursts longer than one word to the XBD should always be 32–bits wide.

Table 5–5: DSP XBus Host Port Registers

DSP Node	DSP	Access Type	XBD i960 Address	XBISA i960 Address
Node A	Master	Non–prefetch	0x8801 0000 – 0x8801 3FFF	0x8801 4000 – 0x8801 7FFF
		Prefetch	0x8810 0000 – 0x8810 3FFF	0x8810 4000 – 0x8810 7FFF
	Slave*	Non–prefetch	0x8801 8000 – 0x8801 BFFF	0x8801 C000 – 0x8801 FFFF
		Prefetch	0x8810 8000 – 0x8810 BFFF	0x8810 C000 – 0x8810 FFFF
Node B	Master	Non–prefetch	0x8804 0000 – 0x8804 3FFF	0x8804 4000 – 0x8804 7FFF
		Prefetch	0x8811 0000 – 0x8811 3FFF	0x8811 4000 – 0x8811 7FFF
	Slave*	Non–prefetch	0x8804 8000 – 0x8804 BFFF	0x8804 C000 – 0x8804 FFFF
		Prefetch	0x8811 8000 – 0x8811 BFFF	0x8811 C000 – 0x8811 FFFF
Node C	Master	Non–prefetch	0x8807 0000 – 0x8807 3FFF	0x8807 4000 – 0x8807 7FFF
		Prefetch	0x8812 0000 – 0x8812 3FFF	0x8812 4000 – 0x8812 7FFF
	Slave*	Non–prefetch	0x8807 8000 – 0x8807 BFFF	0x8807 C000 – 0x8807 FFFF
		Prefetch	0x8812 8000 – 0x8812 BFFF	0x8812 C000 – 0x8812 FFFF
Node D	Master	Non–prefetch	0x880A 0000 – 0x880A 3FFF	0x880A 4000 – 0x880A 7FFF
		Prefetch	0x8813 0000 – 0x8813 3FFF	0x8813 4000 – 0x8813 7FFF
	Slave*	Non–prefetch	0x880A 8000 – 0x880A BFFF	0x880A C000 – 0x880A FFFF
		Prefetch	0x8813 8000 – 0x8813 BFFF	0x8813 C000 – 0x8813 FFFF

* Model 4293 only

5.4 i960 Resources Operation (continued)

5.4.8 PLX PCI 9054 Registers

Each PCI 9054 provides programmable registers that can be accessed by the i960 processor: Configuration, DMA, Mailbox, PCI–to–Local and Local–to–PCI Doorbell, Messaging Queue, Power Management, Hot Swap, and VPD registers. These registers are described in the PLX PCI 9054 Data Book. The i960 can access each node's PCI 9054 registers, using the memory or PCI I/O–mapped addresses indicated in the following table. Also, the default 'C6x firmware provides user functions that can be called using the PCI 9054 Mail–box and Doorbell registers (see Section 10.5).

Table 5–6: PLX PCI 9054 Registers

DSP Node	i960 Memory Address	i960 (PCI I/O) Memory Address
Node A	0x8800 0000 – 0x8800 00FF	0x9001 0000 – 0x9001 00FF
Node B	0x8803 0000 – 0x8803 00FF	0x9001 0300 – 0x9001 03FF
Node C	0x8806 0000 – 0x8806 00FF	0x9001 0600 – 0x9001 06FF
Node D	0x8809 0000 – 0x8809 00FF	0x9001 0900 – 0x9001 09FF

5.4.9 Voltage/Temperature Monitor

The Voltage/Temperature Monitor, an Analog Devices ADM1024, provides constant monitoring of critical voltages and temperatures on the Model 4292/4293 PCB. If the readings fall outside of programmed limits, the ADM1024 asserts an interrupt to the i960 processor (NMI interrupt, Section 8.4), and illuminates the red **Temp** LED on the front panel (Section 2.6.3).



The ADM1024 is programmed with default limits by Pentek's i960 boot firmware, and should **NOT** be modified by the user.

5.4.10 Board ID EEPROM

The serial ID EEPROM contains Model 4292/4293 board identification and configuration information. The size of the ID EEPROM is eight 256–word blocks. See Appendix B for the format and contents of this EEPROM.



The board ID EEPROM is shipped with board identification preset by Pentek, and should **NOT** be modified by the user.

5.5 i960 Control Registers

The i960 I/O processor has access to several 8–bit local control registers that allow it to handle interrupts and control the operation of resources available to it. The following table shows a memory map for these i960 registers. The subsections following this table provide more information about each register.

Table 5–7: i960 Register Memory Map

Memory Address	i960 Register	Access	Information
0xFE00 0000	DSP (SwiftNet) Interrupt Request	R/W	Section 5.5.1
0xFE00 0001	PLX PCI 9054 Interrupt Enables	R/W	Section 5.5.2
0xFE00 0002	DSP (SwiftNet) Interrupt Enables	R/W	Section 5.5.3
0xFE00 0003	Global Bus FIFO Interrupt Enables	R/W	Section 5.5.4
0xFE00 0004	Built–In Self Test #0	R/W	Section 5.5.5
0xFE00 0005	Built–In Self Test #1		
0xFE00 0006	Control	R/W	Section 5.5.6
0xFE00 0007	General Purpose Semaphore	R/W	Section 5.5.7
0xFE00 0008	PLX PCI 9054 Interrupt Status	R.O.	Section 5.5.8
0xFE00 0009	DSP (SwiftNet) Interrupt Flags	R/Clr	Section 5.5.9
0xFE00 000A	Global Bus FIFO Interrupt Status	R.O.	Section 5.5.10

5.5.1 DSP (SwiftNet) Interrupt Request Register

This register allows the i960 processor to assert an interrupt to each 'C6x processor. Each bit sets the 'i960 SwiftNet' interrupt status bit in the Miscellaneous Interrupt Status Registers for the applicable Master DSP (Sections 3.6.5 and 3.6.13), or asserts External Interrupt #6 (**EXT_INT6**) to the applicable Slave DSP on the Model 4293 (Section 4.5.2). These interrupts are reserved for use by SwiftNet—if SwiftNet is not used, they are available for general use.

The following table shows the bit layout of this register. Each bit represents an interrupt from the i960 to one DSP. Set the bit to logic '1' to assert an interrupt to that DSP—the interrupt is asserted when the bit transitions from logic '0' to logic '1'. Clear the bit to logic '0' to clear the interrupt condition.

Table 5–8: DSP (SwiftNet) Interrupt Request Register
R/W @ 0xFE00 0000

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	DSP H	DSP D	DSP G	DSP C	DSP F	DSP B	DSP E	DSP A
Function	0 = No Interrupt 1 = Interrupt							

Slave DSPs E, F, G, and H are available only on Model 4293.
All bits default to the logic '0' state at power up and reset.

5.5 i960 Control Registers (continued)

5.5.2 PLX PCI 9054 Interrupt Enable Register

This register contains interrupt enable bits for the interrupts from each of the PLX PCI 9054 Bus Masters. These enable bits allow each PCI 9054 to interrupt the i960. If an enable bit is set, when the associated status bit in the PLX PCI 9054 Interrupt Status Register (Section 5.5.8) transitions from inactive to active (logic '0' to logic '1'), an interrupt, XINT0, is asserted to the i960.

The following table shows this register's bit layout. Each bit represents an interrupt enable for one PCI 9054 (for Node A, B, C, or D). Set a bit to logic '1' to enable the interrupt. Clear a bit to logic '0' to disable the interrupt.

Table 5–9: PLX PCI 9054 Interrupt Enable Register
R/W @ 0xFE00 0001

	D7 – D4	D3	D2	D1	D0
Bit Name	Reserved	PLX D	PLX C	PLX B	PLX A
Function	Write with zeros, Mask when reading	0 = Interrupt not enabled 1 = Interrupt enabled			
All bits default to the logic '0' state at power up and reset.					

5.5.3 DSP (SwiftNet) Interrupt Enable Register

This register contains interrupt enable bits for interrupts from each 'C6x processor (bit D7 in the Master DSP Miscellaneous Control Register, Section 3.6.8, or a CE3 write by the Slave DSP, Section 4.5.3.2) to the i960. If an enable bit is set in this register, when the associated flag bit in the DSP Interrupt Flag Register (Section 5.5.9) transitions from inactive to active (logic '0' to logic '1'), an interrupt, XINT1, is asserted to the i960. These interrupts are reserved for use by SwiftNet—if SwiftNet is not used, they are available for general use.

The following table shows this register's bit layout. Each bit represents an interrupt enable for one DSP. Set a bit to logic '1' to enable the interrupt. Clear a bit to logic '0' to disable the interrupt.

Table 5–10: DSP (SwiftNet) Interrupt Enable Register
R/W @ 0xFE00 0002

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	DSP H	DSP D	DSP G	DSP C	DSP F	DSP B	DSP E	DSP A
Function	0 = Interrupt not enabled 1 = Interrupt enabled							
Slave DSPs E, F, G, and H are available only on Model 4293. All bits default to the logic '0' state at power up and reset.								

5.5 i960 Control Registers (continued)

5.5.4 Global Bus FIFO Interrupt Enable Register

This register contains interrupt enable bits for the Almost Full and Almost Empty flags from each node's Global Bus FIFO. If an enable bit is set, when the associated bit in the Global Bus FIFO Status Register (Section 5.5.10) transitions from inactive to active (logic '0' to logic '1'), an interrupt, XINT2, is asserted to the i960.

The following table shows this register's bit layout. Each bit represents an interrupt enable for either the Almost Full or the Almost Empty condition from one of the four Global Bus FIFOs: A, B, C, or D. Set a bit to logic '1' to enable the interrupt. Clear a bit to logic '0' to disable the interrupt.

Table 5–11: Global Bus FIFO Interrupt Enable Register
R/W @ 0xFE00 0003

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	FIFO D Almost Full	FIFO D Almost Empty	FIFO C Almost Full	FIFO C Almost Empty	FIFO B Almost Full	FIFO B Almost Empty	FIFO A Almost Full	FIFO A Almost Empty
Function	0 = Interrupt not enabled 1 = Interrupt enabled							

All bits default to the logic '0' state at power up and reset.

5.5.5 Built–In Self Test Registers #0 and #1

The two Built–In Self Test (BIST) registers contain status and control bits for the board's built–in self–tests. These bits are used internally for the board's start–up self–tests (Section 10.4.5), and indicate the results of these self–tests.

The following two tables, below and on the next page, show each register's bit layout. Some of the bits are read only. The subsections following these tables describe the bits of both registers.

Table 5–12: Built–In Self Test Register #0
R/W @ 0xFE00 0004

	D7 – D4	D3*	D2*	D1*	D0
Bit Name	Reserved	Word Test	Ones Test	BIST Status	BIST Start
Function	Write with zeros, Mask when reading	0 = Pass 1 = Fail	0 = Pass 1 = Fail	0 = BIST Complete 1 = BIST In Process	0 = Run BIST on Reset 1 = No BIST on Reset

* These bits are read only.
All bits default to the logic '0' state at power up.

5.5 i960 Control Registers (continued)

5.5.5 Built–In Self Test Registers #0 and #1 (continued)

Table 5–13: Built–In Self Test Register #1
R/W @ 0xFE00 0005

	D7 – D4	D3*	D2*	D1*	D0*
Bit Name	Reserved	Node D	Node C	Node B	Node A
Function	Write with zeros, Mask when reading	0 = Pass 1 = Fail			

* These bits are read only.
All bits default to the logic '0' state at power up.

5.5.5.1 Word Test — Register #0, Bit D3

This read–only bit indicates the results of the Global SDRAM multi–word access test. When read as logic '1', the test failed. When read as logic '0', the test passed.

5.5.5.2 Ones Test — Register #0, Bit D2

This read–only bit indicates the results of the Global SDRAM walking ones test. When read as logic '1', the test failed. When read as logic '0', the test passed.

5.5.5.3 BIST Status — Register #0, Bit D1

This read–only bit indicates the operating status of BIST. When read as logic '1', BIST is in progress. When read as logic '0', BIST has completed.

5.5.5.4 BIST Start — Register #0, Bit D0

This bit controls the BIST execution at a Reset of the board. Set this bit to logic '1' to bypass executing BIST at Reset. Clear this bit to logic '0' to execute BIST at board Reset.

5.5.5.5 Node Tests — Register #1, Bits D3 to D0

These four read–only bits indicate the results of the BIST memory tests for each DSP Node. When read as logic '1', the test failed for that node. When read as logic '0', the test passed.

Refer to Section 10.5.2 for further description of the BIST operation.

5.5 i960 Control Registers (continued)

5.5.6 Control Register

This register contains general board control and status bits. Some of the bits are read only. The following table shows this register's bit layout. The sub–sections following this table describe each register bit.

Table 5–14: Control Register
R/W @ 0xFE00 0006

	D7 – D6	D5 *	D4 *	D3 *	D2 *	D1	D0
Bit Name	Reserved	Jumper J1	Jumper J2	PMC Present	TBC Ready	LED	TBC Reset
Function	Write with zeros, Mask when reading	0 = Jumper 1 = No Jumper		0 = PMC 1 = No PMC	0 = Ready 1 = Testing	0 = Off 1 = On	0 = Ready 1 = Reset

* These bits are read only.
Bits D1 and D0 default to the logic '0' state at power up and reset.

5.5.6.1 Jumper J1, J2 D5, D4

These read–only bits indicate the setting of the Board Power–up Jumpers **J1** and **J2** on the PCB (Section 2.2.1). When read as logic '1', the applicable jumper is not installed. When read as logic '0', the jumper is installed. The default i960 boot program, described in Section 10.4.5, uses these status bits.

5.5.6.2 PMC Present D3

This read–only bit indicates whether or not a PMC module is installed on the 4292/4293. When read as logic '1', a PMC module is not installed. When read as logic '0', a PMC module is installed.

5.5.6.3 TBC Ready D2

This read–only bit indicates the status of the Test Bus Controller (TBC). When read as logic '1', the TBC is performing test functions and is not ready. When read as logic '0', the TBC is ready.

5.5.6.4 LED D1

This bit controls the front panel **Fault** LED (Section 2.6.4). This bit is OR'ed with the processor fault bit from the i960. Set this bit to logic '1' to illuminate the LED. Clearing this bit to logic '0' will turn off the LED, unless the i960 fault indication is set.

5.5.6.5 TBC Reset D0

This bit controls the Test Bus Controller (TBC) reset state. Set this bit to logic '1' to hold the TBC in its reset state. Clear this bit to logic '0' to release the reset and return the TBC to an operable state.

5.5 i960 Control Registers (continued)

5.5.7 General Purpose Semaphore Register

The General Purpose Semaphore Register provides three semaphores that allow the i960 processor or any Master 'C6x processor to request and monitor ownership of common resources on the Model 4292/4293. Each Master 'C6x processor and the i960 processor has a copy of this register (see Section 3.6.9 for the 'C6x processor copies of this register).

Each semaphore has two bits: one bit to request ownership of the semaphore (Requestn) by the processor, and one read–only bit that grants ownership of the semaphore (Grantn) by the board's arbitration logic. The following table shows the semaphore register bit layout. The subsections following the table describe each set of bits.

Table 5–15: General Purpose Semaphore Register
R/W @ 0xFE00 0007

	D7 – D6	D5	D4	D3	D2	D1	D0
Bit Name	Reserved	Grant2	Grant1	Grant0	Request2	Request1	Request0
Function	Write with zeros, Mask when reading	Read only: 1 = Semaphore is owned 0 = Semaphore is available			Read/Write: 1 = Request ownership of Semaphore 0 = Release the Semaphore		
All bits default to the logic '0' state at power up and reset.							

5.5.7.1 Grant2, Grant1, Grant0 D5, D4, D3

These read–only bits indicate the status of each semaphore. When read as logic '1', ownership of this semaphore has been granted to the i960 processor by the board's arbitration logic. When read as logic '0', this semaphore is available.

5.5.7.2 Request2, Request1, Request0 D2, D1, D0

These bits request ownership of each semaphore. Set a bit to logic '1' to request ownership of that semaphore. Clear the bit to logic '0' to release the semaphore for ownership by another processor.

Two of the semaphores are pre–assigned to global resources, as follows.

- Semaphore 0 is assigned to the Universe II VMEbus Master Access.
- Semaphore 1 is assigned to the Global SDRAM.
- Semaphore 2 is user–defined—you may use this semaphore for requesting and monitoring use of any resource.

5.5 i960 Control Registers (continued)

5.5.7 General Purpose Semaphore Register (continued)

To use a semaphore, the processor first requests ownership of the semaphore by setting one of the 'Request' bits (D5, D4, or D3) to logic '1'.

- If that semaphore is not in use, the associated 'Grant' bit (D2, D1, or D0) is set to logic '1' by the board's arbitration logic, and the requesting processor now 'owns' the semaphore.

- If the semaphore is already 'owned' by another processor, the Grant bit will remain logic '0' until the processor that owns it releases it.

After using the resource, the owning processor must clear the applicable 'Request' bit to logic '0' to release the semaphore for use by another processor.

5.5.8 PLX PCI 9054 Interrupt Status Register

This read–only register contains status bits for interrupts from each PLX PCI 9054 Bus Master. The status bit is a buffered version of each PCI 9054's PCI Interrupt (INTA), and may be monitored by software to verify that an interrupting condition has been properly cleared.

When a status bit transitions from inactive to active (logic '0' to logic '1'), interrupt XINT0 is asserted to the i960 if it is enabled by the PLX PCI 9054 Interrupt Enable Register (Section 5.5.2).

The following table shows this register's bit layout. Each bit represents an interrupt from one of the PCI 9054 Bus Masters: A, B, C, or D. When read as logic '1', the interrupt is active. When read as logic '0', the interrupt is not active.

Table 5–16: PLX PCI 9054 Interrupt Status Register
R.O. @ 0xFE00 0008

	D7 – D4	D3	D2	D1	D0
Bit Name	Reserved	PLX D	PLX C	PLX B	PLX A
Function	Mask when reading	0 = Interrupt not active 1 = Interrupt active			
When reset, including power–up, the state of this register is unknown.					

5.5 i960 Control Registers (continued)

5.5.9 DSP (SwiftNet) Interrupt Flag Register

This register contains flag bits for interrupts from each 'C6x processor to the i960. Each flag bit is a latched version of each DSP's interrupt request (from bit D7 in the Master DSP Miscellaneous Control Register, Section 3.6.8, or a CE3 write by the Slave DSP on the Model 4293, Section 4.5.3.2). These interrupts are reserved for use by SwiftNet—if SwiftNet is not used, they are available for general use.

Each flag bit in this register is a read/clear bit. When read as logic '1', a DSP interrupt request has been latched. When read as logic '0', the DSP interrupt is inactive. A flag bit remains latched in this register until you clear it by writing logic '1' to that bit.

When a flag bit in this register transitions from inactive to active (logic '0' to logic '1'), interrupt XINT1 is asserted to the i960 if it is enabled by the DSP Interrupt Enable Register (Section 5.5.3).

The following table shows this register's bit layout. Each flag bit represents an interrupt request from one of the DSPs.

Table 5–17: DSP (SwiftNet) Interrupt Flag Register
R/Clr @ 0xFE00 0009

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	DSP H	DSP D	DSP G	DSP C	DSP F	DSP B	DSP E	DSP A
Function	**READ**: 0 = Interrupt not active; 1 = Interrupt latched **WRITE**: 0 = Normal operation; 1 = Clear latch							

Slave DSPs E, F, G, and H are available only on Model 4293.

IMPORTANT! When reset, including power–up, the state of this register is unknown. You should clear this register before using it, by writing '1's into all bits.

5.5 i960 Control Registers (continued)

5.5.10 Global Bus FIFO Interrupt Status Register

This read–only register contains status bits for the Almost Full and Almost Empty boundary flag conditions from the Global Bus FIFOs. Each status bit is a buffered version of the associated FIFO flag condition, and may be moni–tored by software to verify that a flag status has been properly cleared.

When a status bit transitions from inactive to active (logic '0' to logic '1'), interrupt XINT2 is asserted to the i960 if it is enabled by the Global Bus FIFO Interrupt Enable Register (Section 5.5.4).

The following table shows this register's bit layout. Each bit represents a sta–tus for either the Almost Full or the Almost Empty flag condition from one of the four Global Bus FIFOs: A, B, C, or D. When read as logic '1', the FIFO flag is active. When read as logic '0', the FIFO flag is not active.

Table 5–18: Global Bus FIFO Interrupt Status Register
R.O. @ 0xFE00 000A

	D7	D6	D5	D4	D3	D2	D1	D0
Bit Name	FIFO D Almost Full	FIFO D Almost Empty	FIFO C Almost Full	FIFO C Almost Empty	FIFO B Almost Full	FIFO B Almost Empty	FIFO A Almost Full	FIFO A Almost Empty
Function	0 = FIFO flag not active 1 = FIFO flag active							
When reset, including power–up, the state of this register is unknown.								

The Global Bus FIFO flag levels are programmed by the associated Master 'C6x processor. See Chapter 9 for more information on the operation and programming of the FIFO flags.

Chapter 6: PCI Bus Memory Maps and Resources

6.1 Overview

The Model 4292/4293 supports two PCI bus structures, the Primary and Secondary PCI Buses. The i960 I/O processor acts as the bridge between the two PCI buses. The Secondary PCI Bus provides access to each node's PLX PCI 9054 PCI Bus Master, and to each DSP's Expansion Bus through the PLX PCI 9054s. The Primary PCI Bus provides access to the i960 Messaging Unit and the PMC site, and Master access to the VMEbus. Resources on the Global Bus are also available using the i960 PCI bus bridge.

This chapter presents the memory map and operation of board resources available as PCI memory−mapped and I/O resources to a device that masters either PCI bus. In addition, when accessed by an external VMEbus Master, the Universe II provides access to PCI bus resources in PCI Configuration Space.

6.2 PCI Configuration Space

PCI bus devices can be programmed in the Universe II through PCI Configuration Space or through a VMEbus access to make the resource available in PCI memory space and/or in PCI I/O space. PCI Configuration Cycles can be generated by accessing a VMEbus Slave image whose Local Address Space field (LAS) is set for Configuration Space. Refer to the Tundra Universe II User Manual for description of the PCI Configuration Cycles. The following PCI Configuration Space map shows the resources available using PCI Configuration Cycles.

Table 6−1: PCI Configuration Space Addresses

Resource	PCI Address Line	VMEbus Address (VA15:11)
Universe II	12	**00001**
PMC Site	13	**00010**
82559 Ethernet Controller	14	**00011**
i960 I/O Processor	15	**00100**
DSP Node A PLX PCI 9054	16	**00101**
DSP Node B PLX PCI 9054	17	**00110**
DSP Node C PLX PCI 9054	18	**00111**
DSP Node D PLX PCI 9054	19	**01000**

NOTE: This PCI Configuration Space map is provided for reference only. The default boot sequence (Chapter 10) configures these PCI devices for the PCI memory and I/O maps shown in the following sections—the user does not need to configure these resources using PCI Configuration Space.

6.3 PCI Bus Memory Map

The following memory map shows the resources available to any device that masters either PCI bus, the path to each resource, the PCI memory–mapped address of that resource, and a reference to more information.

Table 6–2: PCI Bus Memory Map

Resource	Path	PCI Memory Address	Information
Node A PLX PCI 9054 Registers	Secondary PCI Bus	0x0000 0000 – 0x0000 00FF	Section 6.5.1
Node A Host Port Registers (NP)		0x0001 0000 – 0x0001 FFFF	Section 6.5.2
Node B PLX PCI 9054 Registers		0x0003 0000 – 0x0003 00FF	Section 6.5.1
Node B Host Port Registers (NP)		0x0004 0000 – 0x0004 FFFF	Section 6.5.2
Node C PLX PCI 9054 Registers		0x0006 0000 – 0x0006 00FF	Section 6.5.1
Node C Host Port Registers (NP)		0x0007 0000 – 0x0007 FFFF	Section 6.5.2
Node D PLX PCI 9054 Registers		0x0009 0000 – 0x0009 00FF	Section 6.5.1
Node D Host Port Registers (NP)		0x000A 0000 – 0x000A FFFF	Section 6.5.2
Node A Host Port Registers (P)		0x0010 0000 – 0x0010 FFFF	Section 6.5.2
Node B Host Port Registers (P)		0x0011 0000 – 0x0011 FFFF	
Node C Host Port Registers (P)		0x0012 0000 – 0x0012 FFFF	
Node D Host Port Registers (P)		0x0013 0000 – 0x0013 FFFF	
Reserved	–	0x0014 0000 – 0x0FFF FFFF	–
i960 Messaging Unit	Primary PCI Bus	0x1000 0000 – 0x1000 0FFF	Section 6.5.3
Global SDRAM Reserved [1]	Global Bus	0x1000 1000 – 0x1040 FFFF	Section 10.4.1
Global SDRAM User Area	Global Bus, 32–bit mode 64–bit mode	0x1041 0000 – 0x10FF FFFF 0x1041 0000 – 0x11FF FFFF	Section 6.5.4
Node A Global Bus FIFO	Global Bus, 32–bit mode [2]	0x1100 0000 – 0x1107 FFFF	Section 9.4.2
Node C Global Bus FIFO		0x1108 0000 – 0x110F FFFF	
Node D Global Bus FIFO		0x1110 0000 – 0x1117 FFFF	
Node B Global Bus FIFO		0x1118 0000 – 0x111F FFFF	
Reserved	–	0x1200 0000 – 0x181F FFFF	–
82559 Ethernet Controller	Primary PCI Bus	0x1820 0000 – 0x1833 0FFF	Section 6.5.5
Universe II Registers		0x1850 0000 – 0x1850 0FFF	Section 6.5.6
PXB Registers (Option 005 only)		0x1860 0000 – 0x1860 FFFF	Section 6.5.8
Reserved	–	0x1861 0000 – 0x8000 FFFF	–
VMEbus Master Access	Primary PCI Bus	0x8001 0000 – 0xEFFF FFFF	Section 6.5.7
PMC Module Access RACEway Access (Option 005 only)		0xF000 0000 – 0xFFFF FFFF	Section 6.5.8

1 – This area of Global SDRAM is reserved for Pentek boot firmware and should not be used.

2 – If PCB jumper **J10** is set for 64–bit data width (Section 2.2.4), the four Global Bus FIFOs are not available.

(NP) = Non–prefetched data access; (P) = Prefetched data access

6.4 PCI Bus I/O Map

The following map shows the PCI I/O addresses for resources on each PCI bus, and a reference to more information.

Table 6–3: PCI Bus I/O Map

Resource	Path	PCI I/O Address	Information
Node A PLX PCI 9054 Registers	Secondary PCI Bus	0x0000 – 0x00FF	Section 6.5.1
Node B PLX PCI 9054 Registers		0x0300 – 0x03FF	
Node C PLX PCI 9054 Registers		0x0600 – 0x06FF	
Node D PLX PCI 9054 Registers		0x0900 – 0x09FF	
82559 Ethernet Controller	Primary PCI Bus	0x1000 – 0x1FFF	Section 6.5.5
Universe II Registers		0x2000 – 0x2FFF	Section 6.5.6
PMC Module Access *		0x3000 – 0x3FFF	Section 6.5.8
* Module dependent—I/O–mapped resources may not be present on all PCI devices.			

6.5 PCI Bus Resources Operation

The following subsections describe the basic operation of the Model 4292/4293 resources available to a device that masters either PCI bus.

6.5.1 PLX PCI 9054 Registers

The PLX PCI 9054 provides programmable registers that are accessible from its PCI Bus: Configuration, DMA, Mailbox, PCI–to–Local and Local–to–PCI Doorbell, Messaging Queue, Power Management, Hot Swap, and VPD registers. These registers are described in the PLX PCI 9054 Data Book. A device that masters either PCI bus can access each node's PCI 9054 registers, at the PCI memory or I/O addresses indicated in the following table.

Table 6–4: PLX PCI 9054 Registers

DSP Node	PCI Memory Address	PCI I/O Address
Node A	0x0000 0000 – 0x0000 00FF	0x0000 – 0x00FF
Node B	0x0003 0000 – 0x0003 00FF	0x0300 – 0x03FF
Node C	0x0006 0000 – 0x0006 00FF	0x0600 – 0x06FF
Node D	0x0009 0000 – 0x0009 00FF	0x0900 – 0x09FF

In addition, the default 'C6x boot firmware provides a variety of functions that can be called using the PCI 9054 Mailbox and Doorbell registers. These 'C6x firmware functions are described in Section 10.5.

6.5 PCI Bus Resources Operation (continued)

6.5.2 DSP Expansion Bus Host Port

A PCI Bus Master has access to each DSP's XBus Host Port, which contains two registers: the XBus Internal Slave Address Register (XBISA) and the XBus Data Register (XBD). Definitions of these registers are provided in the Texas Instruments TMS320C6000 Peripherals User Guide. The XBISA and XBD registers allow a PCI Bus Master to access any address on each DSP's memory map (Sections 3.3 and 4.3), and support both prefetched and non–prefetched data access to those addresses.

- The XBISA controls the memory location (in the DSP) being accessed through the XBus Host Port. A PCI Bus Master can access each DSP's XBISA register at the PCI memory addresses listed in Table 6–5 below.

- The XBD contains the data that was read from the DSP memory location if the access is a read, or the data that is written to the memory if the access is a write. A PCI Bus Master can access each DSP's XBD register at the PCI memory addresses listed in Table 6–5 below.

To read or write to a DSP resource, first write the desired DSP memory address into the XBISA, then read or write the data at that address with the XBD. The host can perform single 32–bit, 16–bit, or 8–bit accesses to the XBD. Bursts longer than one word to the XBD should always be 32–bits wide.

Table 6–5: DSP XBus Host Port Registers

DSP Node	DSP	Access Type	XBD PCI Address	XBISA PCI Address
Node A	Master	Non–prefetch	0x0001 0000 – 0x0001 3FFF	0x0001 4000 – 0x0001 7FFF
		Prefetch	0x0010 0000 – 0x0010 3FFF	0x0010 4000 – 0x0010 7FFF
	Slave*	Non–prefetch	0x0001 8000 – 0x0001 BFFF	0x0001 C000 – 0x0001 FFFF
		Prefetch	0x0010 8000 – 0x0010 BFFF	0x0010 C000 – 0x0010 FFFF
Node B	Master	Non–prefetch	0x0004 0000 – 0x0004 3FFF	0x0004 4000 – 0x0004 7FFF
		Prefetch	0x0011 0000 – 0x0011 3FFF	0x0011 4000 – 0x0011 7FFF
	Slave*	Non–prefetch	0x0004 8000 – 0x0004 BFFF	0x0004 C000 – 0x0004 FFFF
		Prefetch	0x0011 8000 – 0x0011 BFFF	0x0011 C000 – 0x0011 FFFF
Node C	Master	Non–prefetch	0x0007 0000 – 0x0007 3FFF	0x0007 4000 – 0x0007 7FFF
		Prefetch	0x0012 0000 – 0x0012 3FFF	0x0012 4000 – 0x0012 7FFF
	Slave*	Non–prefetch	0x0007 8000 – 0x0007 BFFF	0x0007 C000 – 0x0007 FFFF
		Prefetch	0x0012 8000 – 0x0012 BFFF	0x0012 C000 – 0x0012 FFFF
Node D	Master	Non–prefetch	0x000A 0000 – 0x000A 3FFF	0x000A 4000 – 0x000A 7FFF
		Prefetch	0x0013 0000 – 0x0013 3FFF	0x0013 4000 – 0x0013 7FFF
	Slave*	Non–prefetch	0x000A 8000 – 0x000A BFFF	0x000A C000 – 0x000A FFFF
		Prefetch	0x0013 8000 – 0x0013 BFFF	0x0013 C000 – 0x0013 FFFF

* Model 4293 only

6.5 PCI Bus Resources Operation (continued)

6.5.3 i960 Messaging Unit

The i960 Messaging Unit (MU) has four messaging mechanisms that allow a PCI Bus Master and the i960 I/O processor to communicate through command passing and interrupt generation. The four mechanisms are: Message Registers, Doorbell Registers, Circular Queues, and Index Registers. Definition of these registers and MU programming are provided in the Intel i960® RM/RN I/O Processor Developer's Manual. A PCI Bus Master can access these registers starting at PCI memory address **0x1000 0000**.

6.5.4 Global SDRAM

A device that masters either PCI bus can access the Global SDRAM User Data area starting at PCI memory address **0x1041 0000**. The available size of this memory is either 16 or 32 MBytes, depending on the setting of jumper **J10**, i960 bus data width (Section 2.2.4). However, the first 4 MBytes of Global SDRAM (**0x1000 1000** to **0x1040 FFFF**) is reserved for use by the default boot firmware (Section 10.4.1) and should not be used.

6.5.5 Ethernet Interface

The Intel 82559 controller supports input/output of RS–232 and Ethernet signals, using the front panel connector (Section 2.6.6). To transmit and receive Ethernet signals, you must read and write to the 82559 Control and Status registers. Refer to the Intel 82559 Fast Ethernet Multifunction PCI/CardBus Controller datasheet for description of these registers.

NOTE: This resource is reserved for use exclusively by the i960 firmware.

6.5.6 Universe II Registers

The Tundra Universe II provides programmable registers for VME system configuration and for control of Universe II operational characteristics. These registers are: PCI Configuration Space, VMEbus Configuration and Status Registers, and Universe II Device Specific Status Registers. A quick reference listing of these registers is provided in Appendix E. Refer to the Tundra Universe II User Manual for detailed description of these registers.

A PCI Bus Master can access these Universe II registers starting at PCI memory address **0x1850 0000**, or PCI I/O address **0x2000**.

6.5 PCI Bus Resources Operation (continued)

6.5.7 VMEbus Master Access

When the Model 4292/4293 is a VMEbus Master, any device that masters either PCI bus also has Master access to the VMEbus. That device can access data and perform D16, D32, and D64 block transfers in all relevant VMEbus address spaces. See Section 7.2 for more information on VMEbus Master operation. The VMEbus Master Memory Map for a device that masters either PCI bus is provided in the following table.

Table 6–6: VMEbus Master Memory Map

VMEbus Address Space (Address Modifiers)	PCI Memory Addresses	VMEbus Addresses
A16/D16 (Non–Privileged, 0x29)	0x8001 0000 – 0x8001 FFFF	0x0000 – 0xFFFF
A16/D32 (Non–Privileged, 0x29)	0x8002 0000 – 0x8002 FFFF	0x0000 – 0xFFFF
A24/D16 (Standard, Non–Privileged, Data, 0x39)	0x8100 0000 – 0x81FF FFFF	0x00 0000 – 0xFF FFFF
A24/D32 (Standard, Non–Privileged, Data, 0x39)	0x8200 0000 – 0x82FF FFFF	0x00 0000 – 0xFF FFFF
A32/D32 (Extended, Non–Privileged, Data, 0x09)	0x9000 0000 – 0xEFFF FFFF	0x0000 0000 – 0x5FFF FFFF
A32/D64 (Extended, Non–Privileged, Data, 0x09)	0x8400 0000 – 0x8FFF FFFF	0x0400 0000 – 0x0FFF FFFF

6.5.8 PMC Module Access

The PCI Mezzanine Card (PMC) site accepts industry–standard PMC modules. If a PMC module is installed, a PCI Bus Master can access PMC local address space resources (I/O, memory, registers, DMA, etc.) starting at PCI memory address **0xF000 0000** and, if PCI I/O–mapped access is implemented on the PMC module, starting at PCI I/O address **0x3000**.

Each PMC module is supplied with an Operating Manual that describes any programmable resources of that module. Refer to that manual for description of these resources.

When Option 005 RACEway, is installed, this PMC module supports both outgoing (Master) and incoming (Slave) RACEway operations. A PCI Bus Master can access the RACEway address space starting at memory address **0xF000 0000**, and can access the module's PXB++ chip internal configuration registers starting at memory address **0x1860 0000**. There is no PCI I/O–mapped access implemented on the Option 005 PMC module. Refer to Appendix D for information on operation of this optional RACEway interface.

Chapter 7: VME Interface

7.1 Overview

The Model 4292/4293 provides a full−featured Master/Slave VME64 interface. As a VMEbus Master, the board can access the entire VME address space for A32, A24, and A16 modes of operation. As a result, it can also access any external slave. As a VME−bus Slave, the board presents memory images in A32, A24, and A16 space to any exter−nal VMEbus Master. Using the Tundra Universe II as the interface, the board supports 64−bit master/slave block transfers in addition to 32−bit and 16−bit block transfers. The board also supports 32−bit, 16−bit, and 8−bit single cycle transfers as a VMEbus Master and Slave. As a VMEbus system controller, the board can act as a bus arbiter for multiple master systems. The Model 4292/4293 also supports VMEbus interrupt gen−eration and handling (see Section 8.6 for description of VMEbus interrupt operation).

7.2 VMEbus Master Access

As a VMEbus Master, the Model 4292/4293 can access 32−bit, 16−bit, or 8−bit data and perform D32 and D64 block transfers in all relevant address spaces. The default VME−bus Master Memory Map for master access and block transfers is shown in Table 7−1, below. The user can configure the board's VMEbus master interface from either the i960's serial port or from VME A16/A24 space. Pentek's ReadyFlow® Board Support Libraries provides routines to configure the Universe II from the VMEbus. The Moni−tor program (see Chapter 11) provides configuration commands for specifying a cus−tom VMEbus Master Memory Map. The custom memory map is stored in i960 FLASH memory and restored after a board reset.

Table 7−1: VMEbus Master Memory Map

VMEbus Address Space (Address Modifiers)	i960 Memory Address	PCI Primary/ Secondary Bus Address	VMEbus Address
A16/D16 (Non−Privileged, 0x29)	0x0001 0000 – 0x0001 FFFF	0x8001 0000 – 0x8001 FFFF	0x0000 0000 – 0x0000 FFFF
A16/D32 (Non−Privileged, 0x29)	0x0002 0000 – 0x0002 FFFF	0x8002 0000 – 0x8002 FFFF	0x0000 0000 – 0x0000 FFFF
A24/D16 (Standard, Non−Privileged, Data, 0x39)	0x0100 0000 – 0x01FF FFFF	0x8100 0000 – 0x81FF FFFF	0x0000 0000 – 0x00FF FFFF
A24/D32 (Standard, Non−Privileged, Data, 0x39)	0x0200 0000 – 0x02FF FFFF	0x8200 0000 – 0x82FF FFFF	0x0000 0000 – 0x00FF FFFF
A32/D32 (Extended, Non−Privileged, Data, 0x09)	0x1000 0000 – 0x6FFF FFFF	0x9000 0000 – 0xEFFF FFFF	0x0000 0000 – 0x5FFF FFFF
A32/D64 (Extended, Non−Privileged, Data, 0x09)	0x0400 0000 – 0x0FFF FFFF	0x8400 0000 – 0x8FFF FFFF	0x0400 0000 – 0x0FFF FFFF

7.3 VMEbus Slave Access

As a VMEbus Slave, the Tundra Universe II is responsible for decoding the address and controlling VMEbus access from an external VMEbus Master to various on–board resources. The Global SDRAM, each DSP's Expansion Bus Host Port, each node's Global Bus FIFO, the PCI bus resources, and the PMC site are accessible from the VMEbus. The board's slave interface supports single accesses and block transfers of D16, D32, and D64 data sizes.

The Model 4292/4293 is shipped from the factory with only VMEbus A16 address space slave access enabled. Jumpers on the PCB configure the board to respond as a Slave in either A16 or A24 address space (Section 2.2.6), and set the A16/A24 slave base address (Section 2.2.8). The factory default setting of these jumpers is for A16 address mode, starting at A16 slave base address **0x0000**. In either A16 or A24 address modes, the Model 4292/4293 occupies 4 kBytes of VME space for the default configuration.



If you want an external VMEbus Master to access the board's A24 memory space in A24 address mode, you must set the A24 slave base address (jumper block **JB1**, Section 2.2.8) to either **0x00 0000** or **0x80 0000**.

The Model 4292/4293 is shipped with VME slave access enabled to the Universe II registers only (Default VMEbus A16/A24 Slave Memory Map, Section 7.3.1). You can enable VME slave access to other board resources for A24/A32 address modes using the i960 serial port or writing to the Universe II registers from the VMEbus.

- Pentek's ReadyFlow Board Support Libraries for the Model 4292/4293 provides routines to configure the Universe II registers from the VMEbus. Refer to the ReadyFlow Software Support Documentation (Pentek part #801.42920) for description of these capabilities.

- The Model 4292/4293 Monitor program provides commands for configuring VME slave access to various board resources from the i960 serial port at boot time. Refer to Chapter 11 for description of these capabilities.

When you enable extended VME A24/A32 slave access using the default ReadyFlow or Monitor program parameters, an Extended VMEbus Slave Memory Map is enabled that provides access to various Model 4292/4293 resources from an external VMEbus Master (see Extended VMEbus Slave Memory Map, Section 7.3.2).

You can specify a custom VMEbus slave memory map using the ReadyFlow or Monitor program commands. If you create a custom memory map from the Monitor, the custom memory map is stored in i960 FLASH memory and restored after a board reset. The Monitor program also provides an option to restore the factory default configuration parameters, if needed.

7.3 VMEbus Slave Access (continued)

7.3.1 Default VMEbus A16/A24 Slave Memory Map

At board power on, the Model 4292/4293 provides a default A16/A24 Slave Memory Map that allows access only to the Tundra Universe II registers from an external VMEbus Master. PCB jumper **J20** (Section 2.2.6) configures the board to respond as a Slave in either A16 or A24 space—the factory default setting is for A16 address space. The following table shows the default VME–bus A16/A24 Slave Memory Map.

Table 7–2: Default VMEbus A16/A24 Slave Memory Map

VMEbus Address Space	Universe II Registers Default VMEbus Address
A16	VME_base + **0x0000 – 0x0FFF**
A24	VME_base + **0x00 0000 – 0x00 0FFF**
These addresses are offsets relative to the A16/A24 slave base address 'VME_base', which is set by PCB jumper block **JB1**, Section 2.2.8.	

NOTE: The choice of A16 or A24 is determined by the setting of jumper **J20** on the PCB, Section 2.2.6. The factory default setting is A16.

The Tundra Universe II provides user–programmable registers for VME sys–tem configuration and for control of Universe II operational characteristics: PCI Configuration Space, VMEbus Configuration and Status Registers, and Universe II Device Specific Status Registers. A reference list of these registers is provided in Appendix E. Refer to the Tundra Universe II User Manual for detailed descriptions of these registers.

The user can configure A24 or A32 slave access to various resources on the Model 4292/4293 from VMEbus A16 or A24 space using these Universe II registers. Pentek's ReadyFlow Board Support Libraries provides routines to configure the Universe II registers from an external VMEbus master using the default VMEbus A16 or A24 space.

The Universe II Mailbox Registers are also used by the board's default boot firmware (Section 10.4.2). The following table shows their addresses.

Table 7–3: Universe II Mailbox Registers

Register	A16 VMEbus Address	A24 VMEbus Address
Mailbox 0 Register	VME_base + **0x0348**	VME_base + **0x00 0348**
Mailbox 1 Register	VME_base + **0x034C**	VME_base + **0x00 034C**
Mailbox 2 Register	VME_base + **0x0350**	VME_base + **0x00 0350**
Mailbox 3 Register	VME_base + **0x0354**	VME_base + **0x00 0354**

7.3 VMEbus Slave Access (continued)

7.3.2 Extended VMEbus Slave Memory Map

You can enable extended VMEbus A24/A32 slave access to various board resources using the Universe II registers. If you enable the 'VMEbus Slave Extended Memory Map' option using the default Monitor (Section 11.5.24) or ReadyFlow parameters, an extended VMEbus A24/A32 memory map is enabled, shown in Table 7–4 below. VMEbus addresses in this table are offset from the A24/A32 slave base address 'VME_slave' set by the Monitor or ReadyFlow (the default is **0x0**); the equivalent PCI bus address of each resource is included for reference.

Table 7–4: Extended VMEbus A24/A32 Slave Memory Map

Resource	Path	VMEbus Address Offset	PCI Memory Address	Information
Universe II Registers [1]	VME Bus	VME_slave + **0x00 0000 – 0x00 0FFF**	N/A	Section 7.3.2.1
Node A PLX PCI 9054 Registers	PCI Bus I/O	VME_slave + **0x01 0000 – 0x01 00FF**	PCI I/O Addr: **0x0000 – 0x00FF**	Section 7.3.2.2
Node B PLX PCI 9054 Registers		VME_slave + **0x01 0300 – 0x01 03FF**	PCI I/O Addr: **0x0300 – 0x03FF**	
Node C PLX PCI 9054 Registers		VME_slave + **0x01 0600 – 0x01 06FF**	PCI I/O Addr: **0x0600 – 0x06FF**	
Node D PLX PCI 9054 Registers		VME_slave + **0x01 0900 – 0x01 09FF**	PCI I/O Addr: **0x0900 – 0x09FF**	
PMC Access		VME_slave + **0x01 3000 – 0x01 3FFF**	PCI I/O Addr: **0x3000 – 0x3FFF**	Section 7.3.2.3
Node A Global Bus FIFO	Global Bus, 32-bit mode [2]	VME_slave + **0x02 0000 – 0x02 FFFF**	**0x1100 0000 – 0x1100 FFFF**	Section 9.4.4
Node C Global Bus FIFO		**Not Available** [3]	**0x1108 0000 – 0x1108 FFFF** [3]	
Node D Global Bus FIFO			**0x1110 0000 – 0x1110 FFFF** [3]	
Node B Global Bus FIFO			**0x1118 0000 – 0x1118 FFFF** [3]	
i960 Messaging Unit	Primary PCI Bus	VME_slave + **0x03 0000 – 0x04 FFFF**	**0x1000 0000 – 0x1001 FFFF**	Section 7.3.2.4
Node A PLX PCI 9054 Registers	Secondary PCI Bus	VME_slave + **0x10 0000 – 0x10 00FF**	**0x0000 0000 – 0x0000 00FF**	Section 7.3.2.2
Node A Host Port Registers (NP)		VME_slave + **0x11 0000 – 0x11 FFFF**	**0x0001 0000 – 0x0001 FFFF**	Section 7.3.2.5
Node B PLX PCI 9054 Registers		VME_slave + **0x13 0000 – 0x13 00FF**	**0x0003 0000 – 0x0003 00FF**	Section 7.3.2.2
Node B Host Port Registers (NP)		VME_slave + **0x14 0000 – 0x14 FFFF**	**0x0004 0000 – 0x0004 FFFF**	Section 7.3.2.5
Node C PLX PCI 9054 Registers		VME_slave + **0x16 0000 – 0x16 00FF**	**0x0006 0000 – 0x0006 00FF**	Section 7.3.2.2
Node C Host Port Registers (NP)		VME_slave + **0x17 0000 – 0x17 FFFF**	**0x0007 0000 – 0x0007 FFFF**	Section 7.3.2.5
Node D PLX PCI 9054 Registers		VME_slave + **0x19 0000 – 0x19 00FF**	**0x0009 0000 – 0x0009 00FF**	Section 7.3.2.2
Node D Host Port Registers (NP)		VME_slave + **0x1A 0000 – 0x1A FFFF**	**0x000A 0000 – 0x000A FFFF**	Section 7.3.2.5
Node A Host Port Registers (P)		VME_slave + **0x1C 0000 – 0x1C FFFF**	**0x0010 0000 – 0x0010 FFFF**	Section 7.3.2.5
Node B Host Port Registers (P)		VME_slave + **0x1D 0000 – 0x1D FFFF**	**0x0011 0000 – 0x0011 FFFF**	
Node C Host Port Registers (P)		VME_slave + **0x1E 0000 – 0x1E FFFF**	**0x0012 0000 – 0x0012 FFFF**	
Node D Host Port Registers (P)		VME_slave + **0x1F 0000 – 0x1F FFFF**	**0x0013 0000 – 0x0013 FFFF**	
PMC Access RACEway Access (Option 005)	Primary PCI Bus	VME_slave + **0x20 0000 – 0x3F FFFF**	**0xF000 0000 – 0xF01F FFFF**	Section 7.3.2.3
Global SDRAM (A24)	Global Bus, 32/64-bit mode	VME_slave + **0x41 0000 – 0x7F FFFF**	**0x1041 0000 – 0x107F FFFF**	Section 7.3.2.6
Global SDRAM (A32)	Global Bus, 32-bit	VME_slave + **0x0041 0000 – 0x00FF FFFF**	**0x1041 0000 – 0x10FF FFFF**	
	64-bit	VME_slave + **0x0041 0000 – 0x01FF FFFF**	**0x1041 0000 – 0x11FF FFFF**	

1 – Universe II Registers are available in this map only if PCB jumper **J20** is set to A24 address space (Section 2.2.6).
2 – If PCB jumper **J10** is set for 64–bit data width (Section 2.2.4), the four Global Bus FIFOs are not available.
3 – Global Bus FIFOs B, C, & D are not available with default settings—use Universe II registers to change defaults.
(NP) = Non–prefetched data access; (P) = Prefetched data access

7.3 VMEbus Slave Access (continued)

7.3.2 Extended VMEbus Slave Memory Map (continued)

The following sections describe the operation of Model 4292/4293 resources from an external VMEbus Master using the Extended VMEbus Slave Memory Map shown in Table 7–4. All VMEbus addresses are shown as offsets relative to the VMEbus A24/A32 slave base address 'VME_slave' set by the Monitor or ReadyFlow.

7.3.2.1 Universe II Registers

The Tundra Universe II provides programmable registers for VME system configuration and for control of Universe II operational characteristics: PCI Configuration Space, VMEbus Configuration and Status Registers, and Universe II Device Specific Status Registers. A reference list of these registers is provided in Appendix E. Refer to the Tundra Universe II User Manual for detailed descriptions of these registers.

These registers are available in the Extended VMEbus Slave Memory Map only when PCB jumper **JB20** (Section 2.2.6) is set for A24 address space. Using the Extended VMEbus Memory Map, an external VMEbus Master can access these registers starting at the A24 VMEbus slave address VME_slave + **0x00 0000**.

7.3.2.2 PLX PCI 9054 Registers

Each PLX PCI 9054 provides programmable registers accessible from the PCI Bus: Configuration, DMA, Mailbox, PCI–to–Local and Local–to–PCI Doorbell, Messaging Queue, Power Management, Hot Swap, and VPD registers. These registers are described in the PLX PCI 9054 Data Book.

Using the Extended VMEbus Memory Map, an external VMEbus Master can access each node's PCI 9054 registers, using either PCI memory or I/O–mapped addresses, at the VMEbus addresses indicated in the following table.

Table 7–5: PLX PCI 9054 Registers

DSP Node	VMEbus Address (PCI Memory)	VMEbus Address (PCI I/O)
Node A	VME_slave + **0x10 0000 – 0x10 00FF**	VME_slave + **0x01 0000 – 0x01 00FF**
Node B	VME_slave + **0x13 0000 – 0x13 00FF**	VME_slave + **0x01 0300 – 0x01 03FF**
Node C	VME_slave + **0x16 0000 – 0x16 00FF**	VME_slave + **0x01 0600 – 0x01 06FF**
Node D	VME_slave + **0x19 0000 – 0x19 00FF**	VME_slave + **0x01 0900 – 0x01 09FF**

7.3 VMEbus Slave Access (continued)

7.3.2 Extended VMEbus Slave Memory Map (continued)

7.3.2.3 PMC Access

Using the Extended VMEbus Memory Map, an external VMEbus Master can access the PMC local address space resources, starting at VMEbus slave memory address VME_slave + **0x20 0000**, or using PCI I/O–mapped addresses, starting at VMEbus slave address VME_slave + **0x01 3000**.

Each PMC module is supplied with an Operating Manual that describes any programmable resources of that module. Refer to that manual for description of these resources.

When Option 005, the RACEway interface, is installed, this PMC module supports both outgoing (Master) and incoming (Slave) RACEway operations. An external VMEbus Master can access the RACEway address space starting at VMEbus slave memory address VME_slave + **0x20 0000**. There is no PCI I/O–mapped access implemented on the Option 005 PMC module. Refer to Appendix D for further information on operation of this RACEway interface.

7.3.2.4 i960 Messaging Unit

The i960 Messaging Unit (MU) has four messaging mechanisms that allow an external VMEbus Master to communicate with the i960 I/O processor through command passing and interrupt generation. The four mechanisms are: Message Registers, Doorbell Registers, Circular Queues, and Index Registers. Definition of these registers and MU programming are provided in the Intel i960® RM/RN I/O Processor Developer's Manual.

Using the Extended VMEbus Memory Map, an external VMEbus Master can access these registers starting at VMEbus slave memory address VME_slave + **0x03 0000**.

In addition, the Command Processor application (Chapter 12) provides several data transfer, interrupt, and network functions that can be called by an external VMEbus Master using the i960 MU.

7.3 VMEbus Slave Access (continued)

7.3.2 Extended VMEbus Slave Memory Map (continued)

7.3.2.5 DSP Expansion Bus Host Port

An external VMEbus Master can access two registers on each DSP's XBus Host Port: the Expansion Bus Data Register (XBD) and Expansion Bus Internal Slave Address Register (XBISA). Refer to Texas Instruments' TMS320C6000 Peripherals User Guide for description of these registers. The XBISA and XBD registers allow access to any address on any DSP's memory map (Sections 3.3 and 4.3), and support both prefetched and non–prefetched data access.

- The XBISA controls the DSP memory location being accessed. A VMEbus Master can access the XBISA registers at the VMEbus slave memory addresses listed in Table 7–6 below.
- The XBD contains the data that was read from DSP memory if the access is a read, or the data that is written to DSP memory if the access is a write. A VMEbus Master can access the XBD registers at the VMEbus slave memory addresses in Table 7–6.

To read or write to a DSP resource from the VMEbus, first write the desired DSP memory address to the XBISA, then read or write the data at that address with the XBD. The VMEbus Master can perform single 32–bit, 16–bit, or 8–bit accesses to the XBD. Bursts longer than one word to the XBD should always be 32–bits wide.

Table 7–6: DSP XBus Host Port Registers

DSP Node	DSP	Access Type	XBD VMEbus Address	XBISA VMEbus Address
Node A	Master	Non–prefetch	VME_slave + **0x11 0000 – 0x11 3FFF**	VME_slave + **0x11 4000 – 0x11 7FFF**
		Prefetch	VME_slave + **0x1C 0000 – 0x1C 3FFF**	VME_slave + **0x1C 4000 – 0x1C 7FFF**
	Slave*	Non–prefetch	VME_slave + **0x11 8000 – 0x11 BFFF**	VME_slave + **0x11 C000 – 0x11 FFFF**
		Prefetch	VME_slave + **0x1C 8000 – 0x1C BFFF**	VME_slave + **0x1C C000 – 0x1C FFFF**
Node B	Master	Non–prefetch	VME_slave + **0x14 0000 – 0x14 3FFF**	VME_slave + **0x14 4000 – 0x14 7FFF**
		Prefetch	VME_slave + **0x1D 0000 – 0x1D 3FFF**	VME_slave + **0x1D 4000 – 0x1D 7FFF**
	Slave*	Non–prefetch	VME_slave + **0x14 8000 – 0x14 BFFF**	VME_slave + **0x14 C000 – 0x14 FFFF**
		Prefetch	VME_slave + **0x1D 8000 – 0x1D BFFF**	VME_slave + **0x1D C000 – 0x1D FFFF**
Node C	Master	Non–prefetch	VME_slave + **0x17 0000 – 0x17 3FFF**	VME_slave + **0x17 4000 – 0x17 7FFF**
		Prefetch	VME_slave + **0x1E 0000 – 0x1E 3FFF**	VME_slave + **0x1E 4000 – 0x1E 7FFF**
	Slave*	Non–prefetch	VME_slave + **0x17 8000 – 0x17 BFFF**	VME_slave + **0x17 C000 – 0x17 FFFF**
		Prefetch	VME_slave + **0x1E 8000 – 0x1E BFFF**	VME_slave + **0x1E C000 – 0x1E FFFF**
Node D	Master	Non–prefetch	VME_slave + **0x1A 0000 – 0x1A 3FFF**	VME_slave + **0x1A 4000 – 0x1A 7FFF**
		Prefetch	VME_slave + **0x1F 0000 – 0x1F 3FFF**	VME_slave + **0x1F 4000 – 0x1F 7FFF**
	Slave*	Non–prefetch	VME_slave + **0x1A 8000 – 0x1A BFFF**	VME_slave + **0x1A C000 – 0x1A FFFF**
		Prefetch	VME_slave + **0x1F 8000 – 0x1F BFFF**	VME_slave + **0x1F C000 – 0x1F FFFF**

* Model 4293 only

7.3 VMEbus Slave Access (continued)

7.3.2 Extended VMEbus Slave Memory Map (continued)

7.3.2.6 Global SDRAM

Using the Extended VMEbus Memory Map, an external VMEbus Master can access the Global SDRAM, starting at VMEbus A24 slave address VME_slave + **0x41 0000**, or at VMEbus A32 slave address VME_slave + **0x0041 0000**. These addresses are at the start of the 'User Data' area of Global SDRAM (equivalent to PCI bus memory address **0x1041 0000**)—the first 4 MBytes are reserved for the default boot firmware use (Section 10.4.1) and cannot be accessed from the VMEbus.

To read or write data to this memory, access any longword address within the available memory space. The total size of Global SDRAM is 16 or 32 MBytes, depending on the setting of PCB jumper **J10**, i960 bus data width (Section 2.2.4). However, in the Extended VMEbus Memory Map, the size of Global SDRAM available to an external VMEbus Master is different when addressed from VMEbus A24 space than from A32 space, as shown in the following table.

Table 7–7: VMEbus Global SDRAM Size/Addresses

VMEbus Address Space	i960 Bus Data Width	Global SDRAM Available Size	VMEbus Address
A24	32–bit	4 MBytes	VME_slave + **0x41 0000 – 0x7F FFFF**
	64–bit		
A32	32–bit	12 MBytes	VME_slave + **0x0041 0000 – 0x00FF FFFF**
	64–bit	28 MBytes	VME_slave + **0x0041 0000 – 0x01FF FFFF**

Chapter 8: Interrupt Operation

8.1 Overview

This chapter describes the operation of interrupts on the Model 4292/4293. Interrupts can be asserted from each 'C6x processor, each FIFO, each PCI 9054, each VIM site, the i960 processor, the VME interface, the PMC site, the RS–232/Ethernet interface, and the Voltage/Temperature monitor. Each 'C6x processor, each PCI 9054, the i960, and the VME interface can respond to interrupts. Figure 8–1, on the next page, provides an illustration of these interrupts paths.

8.2 General Interrupt Operation

Each Master 'C6x processor can assert interrupts from a DSP register and from its Expansion Bus (XBus) Host Port Interface. Each Master 'C6x can respond to interrupts received on four external interrupt lines: **EXT_INT4**, **EXT_INT5**, **EXT_INT6**, or **EXT_INT7**. Separate DSP External Interrupt Enable Registers allow you to select which of these four external interrupts you wish to use for each external interrupt source.

Section 8.3 provides a summary of the Master 'C6x interrupts. Refer to Section 3.6 for descriptions of the DSP interrupt registers. Refer to Section 4.5 for description of the Slave 'C6x interrupts. Refer to the Texas Instruments TMS320C6000 Peripherals Reference Guide for description of 'C6x interrupt programming.

The i960 processor can assert interrupts from a programmable register and from four PCI Interrupt outputs **P_INTA** to **P_INTD**. The i960 can respond to interrupts received on several external interrupt lines: **XINT0**, **XINT1**, **XINT2**, **XINT3**, **XINT4**, **XINT5**, or **NMI**. These interrupt lines are assigned to the following external interrupt sources:

XINT0: OR of four PLX PCI 9054 INTAs (via PLD status register)
XINT1: OR of four 'C6x processor register interrupts (via PLD status register)
XINT2: OR of four Global Bus FIFO flags (via PLD status register)
XINT3: VME Interface Tundra Universe II LINT1
XINT4: PMC module INTA
XINT5: RS–232/Ethernet Intel 82559 INTA
NMI: Voltage/Temperature Monitor ADM1024 INT

Section 8.4 provides a summary of all interrupt sources and destinations for the i960 processor. Refer to Section 5.5 for descriptions of the i960 interrupt registers. Refer to the Intel i960 RN I/O Processor Developer's Manual for description of the i960 interrupt programming.

Section 8.5 describes the operation of the PCI Bus Doorbell interrupts.

Section 8.6 describes the operation of VMEbus interrupts.

8.2 General Interrupt Operation (continued)

The interrupt interconnections for the Model 4292/4293 are illustrated below.

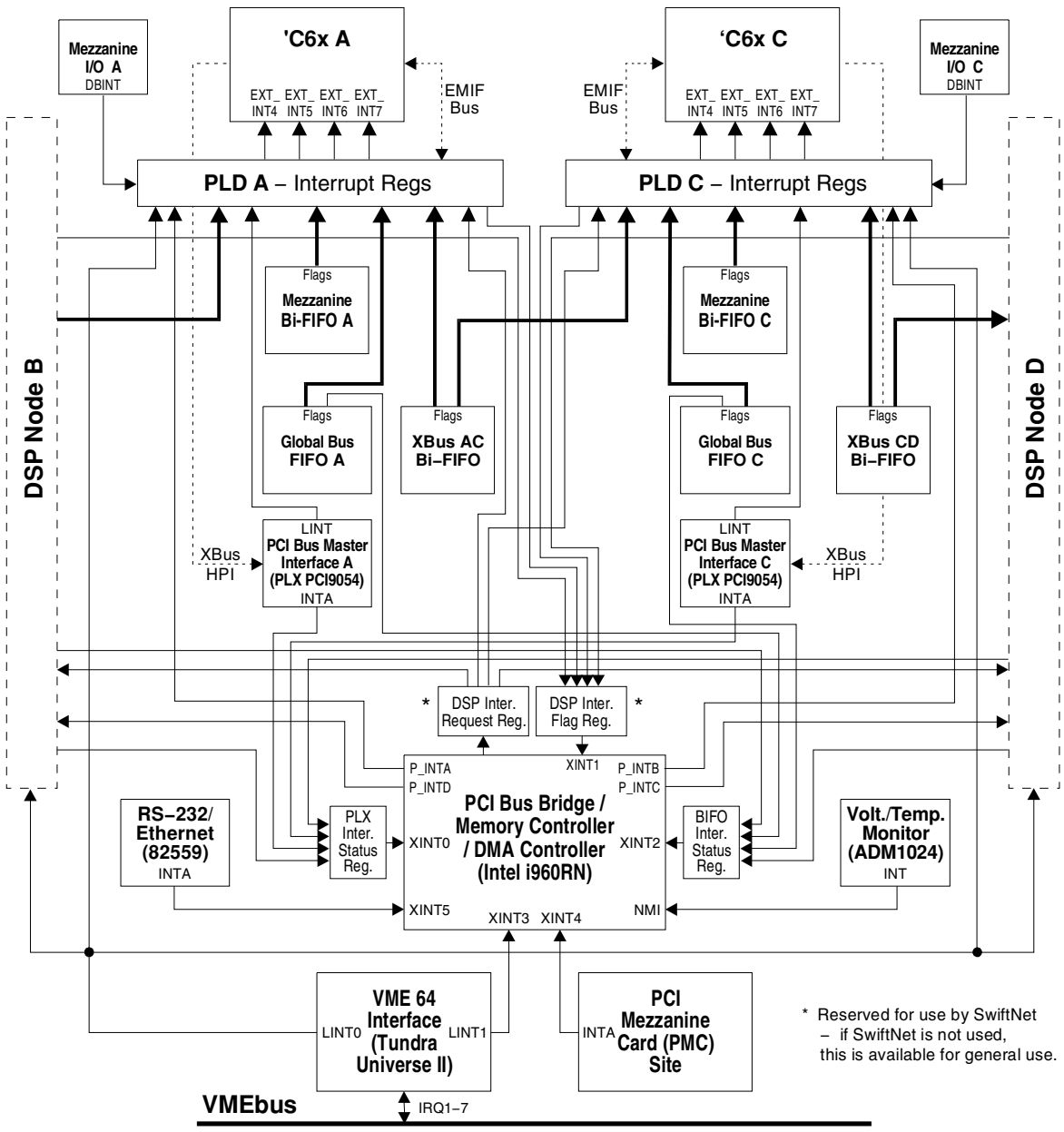


Figure 8–1: Model 4292/4293 Interrupt Connections
(Slave 'Cx6s not illustrated – Nodes B and D not detailed)

See Figure 4–1, in Section 4.5, for illustration of the Slave 'C6x processor interrupt paths for the Model 4293. See also Section 9.5 for detailed illustrations of the FIFO interrupt logic for one of the FIFOs.

8.3 Master 'C6x Processor Interrupts

The following two tables identify the outgoing (Table 8–1) and incoming (Table 8–2) interrupt paths between each Master 'C6x processor and other devices on the board, and provide references to the board resources to use for asserting or responding to these interrupts. See Section 4.5 for the interrupt paths for the Slave 'Cx6 processors. See Section 8.4 for the interrupt paths for the i960 processor.

Table 8–1: Master DSP Outgoing Interrupts

'C6x Interrupt Source	Interrupt Path	Interrupt Destination
'C6x XBus Host Port (Section 3.5)	XBus HPI, to PCI Doorbell Interrupt	PCI INTA, to i960 (Section 8.5.1)
	HPI, to Universe II Mailbox register	VMEbus (Section 8.6.2)
Miscellaneous Control Register (Section 3.6.8)	DSP register to i960 register	i960, DSP Interrupt Flag Register (Section 5.5.9) [1]

1 – Reserved for use by SwiftNet—if SwiftNet is not used, this is available for general use.

Table 8–2: Master DSP Incoming Interrupts

Interrupt Source	Interrupt Path	'C6x Interrupt Destination
All FIFOs (Chapter 9)	FIFO flags, to DSP registers	FIFO Status Registers (Section 3.6.3) [4]
i960, DSP Interrupt Request Register (Section 5.5.1) [1]	i960 register, to DSP register	Miscellaneous Interrupt Status Register (Section 3.6.5) and Miscellaneous Interrupt Status Register #2 (Section 3.6.13) [4]
i960 PCI Interrupt [2]	i960 P_INTx, to DSP register	
VME Interface (Section 8.6.1)	Universe II LINT0, to DSP register	
PCI 9054 Local Interrupt (Section 8.5.2)	PCI 9054 LINT, to DSP register	
VIM module [3]	VIM DBINT, to DSP register	

1 – Reserved for use by SwiftNet—if SwiftNet is not used, this is available for general use.

2 – Refer to Intel i960 RN I/O Processor Developer's Manual for description of the i960 PCI interrupt programming.

3 – Each Pentek VIM module is supplied with an Operating Manual that provides register and interrupt programming instructions for that module.

4 – There are two levels of 'C6x interrupt enabling registers:

- The first level, FIFO Interrupt Enable Registers (Section 3.6.1), enables the FIFO source interrupts (from the FIFO flags).
- The second level, External Interrupt Enable Registers (Section 3.6.6), steers (enables) each FIFO register's interrupt and each miscellaneous interrupt to a unique 'C6x processor external interrupt, **`EXT_INT4`**, **`EXT_INT5`**, **`EXT_INT6`**, or **`EXT_INT7`**. These four registers let you select which of the four 'C6x external interrupts to use for each FIFO register and miscellaneous interrupt (see also Section 9.5).

8.4 i960 Processor Interrupts

The following two tables identify the outgoing (Table 8–3) and incoming (Table 8–4) interrupt paths between the i960 processor and other devices on the board, and provide references to the board resources to use for asserting or responding to these interrupts. See Section 8.3 for the interrupt paths for the Master 'C6x processors.

Table 8–3: i960 Processor Outgoing Interrupts

i960 Interrupt Source	Interrupt Path	Interrupt Destination
DSP Interrupt Request Register (Section 5.5.1) [1]	i960 register, to DSP register	DSP x, Miscellaneous Interrupt Status Register (Section 3.6.5) & Miscellaneous Interrupt Status Register #2 (Section 3.6.13)
i960 PCI Interrupt	i960 P_INTx, to DSP x register [2]	

1 – Reserved for use by SwiftNet—if SwiftNet is not used, this is available for general use.

2 – Refer to Intel i960 RN I/O Processor Developer's Manual for description of PCI interrupt programming.
Each i960 P_INTx interrupt line is routed to (the DSP register of) each Node as follows:
P_INTA to Node A
P_INTB to Node C
P_INTC to Node D
P_INTD to Node B

Table 8–4: i960 Processor Incoming Interrupts

Interrupt Source	Interrupt Path	i960 Interrupt Destination
PCI 9054, PCI Interrupt (Section 8.5.1)	PCI 9054 INTA, to i960 register	PLX Interrupt Status Register (Section 5.5.8), XINT0
DSP x, Miscellaneous Control Register (Section 3.6.8)	DSP register to i960 register	DSP Interrupt Flag Register (Section 5.5.9), XINT1 [1]
Global Bus FIFOs (Chapter 9)	FIFO flags, to i960 register	FIFO Interrupt Status Register (Section 5.5.10), XINT2
VME Interface (Section 8.6.1)	Universe II LINT1, to i960	XINT3 [3]
PMC [2]	PMC module INTA, to i960	XINT4 [3]
RS–232/Ethernet (Section 5.4.2)	Intel 82559 INTA, to i960	XINT5 [3]
Volage./Temperature Monitor (Section 5.4.9)	ADM1024 INT, to i960	NMI [3]

1 – Reserved for use by SwiftNet—if SwiftNet is not used, this is available for general use.

2 – Each Pentek PMC module is supplied with an Operating Manual that provides register and interrupt programming instructions for that module.

3 – Refer to Intel i960 RN I/O Processor Developer's Manual for description of the i960 interrupt programming.

8.5 PCI Doorbell Interrupts

Each PLX PCI 9054 has two 32–bit Doorbell Interrupt/Status registers. One is assigned to the PCI Bus interface, the other is assigned to the Local Bus interface. A local (Master 'C6x) processor can assert a PCI Bus interrupt by writing any number other than all zeroes to the Local–to–PCI Doorbell register bits. A PCI Master can assert a Local Bus interrupt by writing any number other than all zeroes to the PCI–to–Local Doorbell register bits. The PCI Interrupt and Local Interrupt remain asserted until all bits are cleared to zero in the respective register.

8.5.1 Local–to–PCI Doorbell Register

Each Master 'C6x processor can access the doorbell register for the PCI 9054 on its DSP node, using its XBus Host Port Interface (see Section 3.5.3).

When a Master 'C6x processor writes to the Local–to–PCI Doorbell Register, a PCI interrupt (**INTA**) is asserted to the i960. The i960 processor can then read this PCI register to determine which doorbell bit was asserted. The interrupting 'C6x processor sets a doorbell by writing a '1' to a particular bit. The i960 processor clears a doorbell bit by writing a '1' to that bit position.

8.5.2 PCI–to–Local Doorbell Register

Any device that masters the PCI bus can access the doorbell register for the PCI 9054 on any DSP node (see Sections 6.5.1 and 7.3.2.2).

When a PCI Master writes to the PCI–to–Local Doorbell Register, a Local interrupt (**LINT**) is asserted to the associated Master 'C6x processor. The 'C6x processor can then read this PCI register to determine which doorbell bit was asserted. The PCI Master sets a doorbell by writing a '1' to a particular bit. The 'C6x processor clears a doorbell bit by writing a '1' to that bit position.

The PCI 9054 Interrupt Programming and the Doorbell and Interrupt Control Registers are described in the PLX PCI 9054 Data Book.

The Model 4292/4293 boot firmware provides several 'C6x functions that can be called using the PCI Mailbox and Doorbell Registers. The operation of these firmware functions is described in Section 10.5.

8.6 VMEbus Interrupts

The Model 4292/4293 is capable of acting as a VMEbus interrupt handler. The seven VME Interrupt Request signals (IRQ1 to IRQ7) are routed to the Tundra Universe II, which in turn can map a direct interrupt to each 'C6x processor and/or the i960 processor from these signals. Also, any processor on the Model 4292/4293 can assert a VMEbus interrupt, on any of the seven IRQ levels, by setting the appropriate bit in the Universe II's internal Mailbox Registers. A reference list of the Tundra Universe II registers is provided in Appendix E.

8.6.1 Handling VMEbus Interrupts

As a VMEbus interrupt handler, the Universe II can monitor any or all of the VMEbus interrupt levels, IRQ1 to IRQ7. Each interrupt is enabled through the Universe II's LINT_EN register. Once enabled, assertion of any VMEbus interrupt level causes the internal interrupt handler circuitry to request ownership of the Universe II's VMEbus Master Interface. The Universe II latches the first interrupt that appears on the VMEbus and performs an interrupt acknowledge for the first interrupt it detected. Once the IACK cycle is complete, an interrupt is generated to the PCI bus on one of the LINT0 to LINT7 lines, depending on the mapping for that VMEbus level in the LINT_MAP0 register. The interrupt is then cleared and the VMEbus interrupt level is re-armed. Refer to the Tundra Universe II User Manual for description of the Universe II VMEbus interrupt handling.

The LINT0 interrupt line from the Universe II is routed to all Master DSPs' Miscellaneous Interrupt Status Registers (Sections 3.6.5 and 3.6.13), where it can assert an interrupt to any 'C6x processor if the appropriate External Interrupt Enable Register bit is set (Section 3.6.6). The Universe II LINT1 interrupt line is routed to the XINT3 interrupt input of the i960 processor.

8.6.2 Interrupting the VMEbus

The Universe II can be programmed to assert an interrupt on the VMEbus when one of its Mailbox Registers is accessed. The mailboxes support read and write accesses, and may be enabled by the LINT_EN register to generate interrupts on the VMEbus when they are written to. When enabled, once the mailbox interrupt has been received, the Universe II asserts the appropriate VMEbus interrupt output signal. Refer to the Tundra Universe II User Manual for description of Universe II VMEbus interrupt generation.

Using Pentek's ReadyFlow, the i960 processor has primary control over the Universe II registers. All user interrupts to the VMEbus should be generated using ReadyFlow calls to appropriate i960 functions.

However, if needed, a Master 'C6x processor can directly issue a VMEbus interrupt, on any IRQ level, by setting the appropriate bit in the Universe II Mailbox Register using its XBus Host Port Interface (Section 3.5.2).

Chapter 9: FIFO Operation

9.1 Overview

This chapter describes the operation of the Model 4292/4293 Bi–FIFOs. On each Master 'C6x processor, the XBus I/O Port is connected to four Bi–FIFOs that provide high–throughput data paths and messaging functions for the processors. The standard size of each Bi–FIFO is 2k x 32 (1k 32–bit words in each direction); for Option 320 each Bi–FIFO is 32k x 32 (16k 32–bit words in each direction). Two Bi–FIFOs provide bi–directional interprocessor communications using the XBus interface; one Bi–FIFO provides bi–directional access to the VIM site associated with each DSP node; and one Bi–FIFO provides a data path from the Global Bus resources (including VMEbus). The following figure illustrates the connections between each FIFO and each DSP.

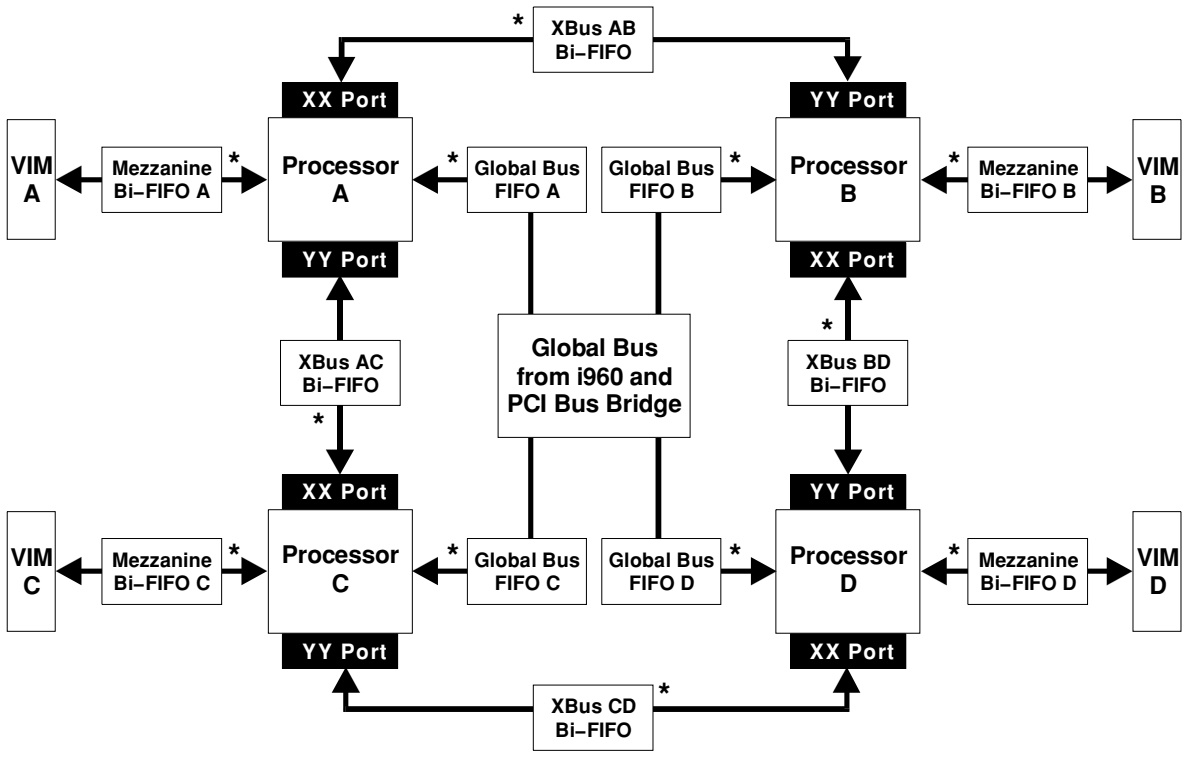


Figure 9–1: Model 4292/4293 FIFO Connections
(* denotes the side of the FIFO to reset and initialize)

NOTE: In this manual the two XBus Inter–Processor (I/P) Bi–FIFO ports on each processor are labeled as XX and YY. This I/P Bi–FIFO port identifier (XX or YY) represents a unique connection path, where the XX port interface denotes the processor that has control over that Bi–FIFO.

9.2 Resetting the FIFOs

Each FIFO may be individually reset by the controlling Master 'C6x processor using its DSP Miscellaneous Control Register (see Section 3.6.8, and Table 9–1 below). Bit D2 resets the Global Bus FIFO, bit D3 resets the XBus Inter–Processor XX Bi–FIFO, and bit D4 resets the Mezzanine Bi–FIFO. If a reset bit is set but not cleared, the FIFO is held in its reset state (see note below).

Table 9–1: DSP Miscellaneous Control Register (MCR)
R/W @ 0x0220 0064

	D31 – D10					D9	D8
Bit Name	Reserved					Slave Interrupt Request	VIM Module Clock
Function	Write with zeros, Mask when reading					(See Section 3.6.8)	
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1 – D0**
Bit Name	i960 Interrupt Request	VIM Module Reset	Reserved	Mezzanine BI-FIFO Reset	XBus XX BI-FIFO Reset	Global Bus FIFO Reset	Reserved
Function	(See Section 3.6.8)		Write zeros, Mask read	0 = Normal 1 = Reset	0 = Normal 1 = Reset	0 = Normal 1 = Reset	Write zeros, Mask read
All bits default to the logic '0' state at power up and reset.							



When you set a reset bit in this register to logic '1' to reset a FIFO, the FIFO is held in a reset state—you must clear that bit to logic '0' to return the FIFO to normal operation.

Figure 9–1, on the prior page, indicates which side of each FIFO is the control side—that is, the side that is used for reset and initialization (see Section 9.3).

- For the Global Bus FIFOs and Mezzanine Bi–FIFOs, the control side is the side connected to the associated 'C6x processor XBus I/O Port.

- For XBus Inter–Processor Bi–FIFOs, the processor with the XX port of the Bi–FIFO link has control over that FIFO. For example, Processor A can reset and initialize XBus Bi–FIFO AC, but Processor C cannot (Processor C controls XBus Bi–FIFO CD).

  Each Master 'C6x processor is connected to two XBus Inter–Processor Bi–FIFOs using its XX and YY ports (see Figure 9–1, prior page). However, reset and initialization of the I/P Bi–FIFOs can be done only from the XX port interface. The XX port of each processor is only attached to one XBus I/P Bi–FIFO, and this processor is responsible for that FIFO's reset (above) and initialization (next page).

As part of the Model 4292/4293 initialization, the 'C6x processor boot code resets all FIFO paths (Section 10.4.3). See also the reset logic diagram in Section 2.7 for illustration of the reset logic at power–up or reset of the board.

9.3 Initializing the FIFOs

After a FIFO is reset (and reset is cleared for operation), the controlling Master 'C6x processor must initialize that FIFO's programmable boundary flag levels (Almost Full and Almost Empty). The 'C6x accomplishes this through the first four writes to the same side of the FIFO responsible for its reset (see Section 9.2). (For the XBus I/P Bi–FIFOs, this is the XX port.) Even if the flags are not used, four dummy writes must occur before data transfers, since after a reset the first four writes to the FIFO are always loaded into its flag registers. The valid programming range for the flag levels is from (decimal) 1 to 1020 32–bit words (1 to 16,380 words for Option 320).

To program the flags, the first four writes to the FIFO should be:

Word 1 = **1024 - w** (Outgoing FIFO Almost Full)
Word 2 = **x** (Outgoing FIFO Almost Empty)
Word 3 = **1024 - y** (Incoming FIFO Almost Full)
Word 4 = **z** (Incoming FIFO Almost Empty)

Where:

w = Outgoing FIFO Almost Full setting
(also, Incoming Almost Full setting for YY port of I/P Bi–FIFO)

x = Outgoing FIFO Almost Empty setting
(also, Incoming Almost Empty setting for YY port of I/P Bi–FIFO)

y = Incoming FIFO Almost Full setting
(also, Outgoing Almost Full setting for YY port of I/P Bi–FIFO)

z = Incoming FIFO Almost Empty setting
(also, Outgoing Almost Empty setting for YY port of I/P Bi–FIFO)

1024 = size of the standard FIFO buffer (32–bit words) *

* With Option 320, the FIFO buffer size is **16384** (32–bit words).

NOTE: Although the Global Bus FIFOs have no Outgoing FIFO capability, you must still write a value to the first two words (**w** and **x**).

Write the binary value of each word into the lowest bits of the word, where bit D0 is the least significant bit of the binary number. To write these four words, perform a DMA write, in synchronous FIFO mode, to the FIFO Data address on the 'C6x XBus I/O Port. See Table 9–2, on the following page, for the XBus I/O Port addresses of each FIFO.

After the first four writes to establish the flag levels, the FIFO is ready to accept data and messages, as described in Section 9.4 on the following pages. FIFO read and write operations will set and clear the appropriate FIFO status flags at the associated 'C6x and i960 processors when the data reaches the programmed boundary flag levels. These FIFO status flags can be enabled to assert interrupts to the respective processors, as described in Section 9.5.

9.4 FIFO Data and Mailbox Transfers

The following subsections describe each type of access to the FIFOs.

9.4.1 Master 'C6x FIFOs Operation

To read or write data to a FIFO from the Master 'C6x processor, perform a DMA read or write, in synchronous FIFO mode, to the FIFO Data address on the 'C6x XBus I/O Port. The following table lists the XBus I/O Port addresses of each FIFO. (Note that the Global Bus FIFO has only incoming data capability and does not have Mailbox capability.)

Table 9–2: 'C6x FIFOs XBus I/O Port Address Map

Resource	'C6x Memory Address
Global Bus FIFO Data (Incoming, read only)	0x4000 0000
Reserved	0x4000 0008 – 0x4FFF FFFF
XBus Inter–Processor XX Bi–FIFO Data	0x5000 0000
XBus Inter–Processor XX Bi–FIFO Mailbox	0x5000 0020
Reserved	0x5000 0024 – 0x5FFF FFFF
XBus Inter–Processor YY Bi–FIFO Data	0x6000 0000
XBus Inter–Processor YY Bi–FIFO Mailbox	0x6000 0020
Reserved	0x6000 0024 – 0x6FFF FFFF
Mezzanine Bi–FIFO Data	0x7000 0000
Mezzanine Bi–FIFO Mailbox	0x7000 0020
Reserved	0x7000 0024 – 0x7FFF FFFF

FIFO data reads are from the inbound FIFO and writes are to the outbound FIFO. Read and write operations will cause the appropriate 'C6x status flags to be set and cleared when the data reaches the programmed boundary levels. See Section 3.6.3 for the DSP FIFO Status Register bits set by these flags. See Section 9.5 for description of the FIFO interrupts that can be asserted from these flags.



Option 304 Only – For a Mezzanine or XBus Inter–Processor FIFO write, the first word of the data is not sent to the other side of the FIFO, so you must write one more word than the number or words you want to send. For example, to send 512 words to a FIFO, you must perform a DMA write of 513 words with a dummy first word.

Writes to the FIFO Mailbox address send data to the opposite side of the FIFO and set the mailbox flag, and read operations read the incoming Mailbox and reset the local mailbox flag. See Section 3.6.3 for the DSP FIFO Status Register bits set by these flags.

9.4 FIFO Data and Mailbox Transfers (continued)

9.4.1 Master 'C6x FIFOs Operation (continued)

The Global Bus sides of all four Global Bus FIFOs are available to each Master DSP from its XBus Host Port Interface (HPI) (see Section 3.5). This access is write–only from each 'C6x, as shown in the following table.

Table 9–3: 'C6x Global Bus FIFO XBus HPI Address Map

Resource	'C6x XBus HPI Address
Node A Global Bus FIFO (write only)	0x1100 0000 – 0x1107 FFFF
Node B Global Bus FIFO (write only)	0x1118 0000 – 0x111F FFFF
Node C Global Bus FIFO (write only)	0x1108 0000 – 0x110F FFFF
Node D Global Bus FIFO (write only)	0x1110 0000 – 0x1117 FFFF



When PCB jumper **J10** is set for 64–bit mode (Section 2.2.4), the four Global Bus FIFOs are not available to a 'C6x on its XBus HPI.

9.4.2 PCI Global Bus FIFOs Operation

To write data to a Global Bus FIFO from a device that masters either PCI bus, access the appropriate FIFO's PCI bus memory address. This access is write–only from a PCI bus device, as shown in the following table.

Table 9–4: PCI Global Bus FIFO Address Map

Resource	PCI Bus Memory Address
Node A Global Bus FIFO (write only)	0x1100 0000 – 0x1107 FFFF
Node B Global Bus FIFO (write only)	0x1118 0000 – 0x111F FFFF
Node C Global Bus FIFO (write only)	0x1108 0000 – 0x110F FFFF
Node D Global Bus FIFO (write only)	0x1110 0000 – 0x1117 FFFF



When PCB jumper **J10** is set for 64–bit mode (Section 2.2.4), the four Global Bus FIFOs are not available to a PCI bus device.

FIFO writes from the PCI bus will cause the appropriate DSP Incoming FIFO status flags to be set and cleared for the associated 'C6x processor when the data reaches the boundary levels. See Section 3.6.3 for the DSP FIFO Status Register bits set by these flags.

9.4 FIFO Data and Mailbox Transfers (continued)

9.4.3 i960 Global Bus FIFOs Operation

To write data to a Global Bus FIFO from the i960 processor, access the appropriate FIFO's i960 memory address. This access is write–only from the i960, as shown in the following table.

Table 9–5: i960 Global Bus FIFO Address Map

Resource	i960 Memory Address
Node A Global Bus FIFO (write only)	0xA100 0000 – 0xA107 FFFF
Node B Global Bus FIFO (write only)	0xA118 0000 – 0xA11F FFFF
Node C Global Bus FIFO (write only)	0xA108 0000 – 0xA10F FFFF
Node D Global Bus FIFO (write only)	0xA110 0000 – 0xA117 FFFF



When PCB jumper **J10**, i960 memory/bus data width, is set for 64–bit mode (Section 2.2.4), the four Global Bus FIFOs are not available to the i960 processor.

FIFO writes from the i960 processor will cause the appropriate i960 FIFO status flags to be set and cleared when the data reaches the programmed boundary levels. See Section 5.5.10 for the i960 FIFO Status Register bits set by these flags. See also Section 9.5 for description of the FIFO interrupts that can be asserted to the i960.

FIFO writes from the i960 processor will also cause the appropriate DSP Incoming FIFO status flags to be set and cleared for the associated 'C6x processor when the data reaches the boundary levels. See Section 3.6.3 for the DSP FIFO Status Register bits set by these flags.

9.4 FIFO Data and Mailbox Transfers (continued)

9.4.4 VMEbus Global Bus FIFOs Operation

To write data to a Global Bus FIFO from an external VMEbus Master, access the FIFO's VMEbus memory address. This access is write–only from the VMEbus. The following table shows the VMEbus slave addresses for each node's Global Bus FIFO. These addresses are shown as offsets relative to the VMEbus Slave base address 'VME_slave' using the Extended VMEbus Slave Memory Map (Section 7.3.2).

Table 9–6: VMEbus Global Bus FIFO Address Map

Resource	VMEbus Address Offset
Node A Global Bus FIFO (write only)	VME_slave + **0x02 0000 – 0x02 FFFF**
Node B Global Bus FIFO (write only)	Not Available *
Node C Global Bus FIFO (write only)	
Node D Global Bus FIFO (write only)	
* Using the default Extended VMEbus Slave Memory Map, Global Bus FIFOs B, C, and D are not available from the VMEbus.	



1) Using the default Extended VMEbus Slave parameters, only the Node A Global Bus FIFO is available from an external VMEbus Master. With these defaults, VMEbus address VME_slave + **0x02 0000** corresponds to PCI bus memory address **0x1100 0000**, which is the start address of the Node A FIFO. To access any other Global Bus FIFO from an external VMEbus Master, you must reprogram this VMEbus address space, using the Universe II registers, to start at the desired Global Bus FIFO's PCI bus memory address (see Table 9–4 for these PCI addresses).

2) When PCB jumper **J10**, i960 memory/bus data width, is set for 64–bit mode (Section 2.2.4), the four Global Bus FIFOs are not available from the VMEbus.

FIFO writes from the VMEbus will cause the appropriate DSP Incoming FIFO status flags to be set and cleared for the associated 'C6x processor, and the i960 FIFO status flags to be set and cleared for the i960 processor, when the data reaches the programmed boundary levels. See Section 3.6.3 for the DSP FIFO Status Register bits set by these flags; see Section 5.5.10 for the i960 FIFO Status Register bits set by these flags. See also Section 9.5 for description of the FIFO interrupts that can be asserted.

9.5 FIFO Interrupts

FIFO reads and writes will set and clear the appropriate FIFO status flags for the asso–ciated ’C6x and i960 processors when the data reaches the programmed boundary lev–els described in Section 9.3. You can enable these FIFO flags to assert interrupts to the respective processors using the interrupt enable registers, Sections 3.6.1 and 5.5.4.

The following figure illustrates the interrupt logic for the Global Bus A FIFO, for Model 4292/4293 boards with FPGA firmware revision 0. For boards with FPGA firmware revision 1 or greater, see Figures 9–3 and 9–4 on the next two pages.

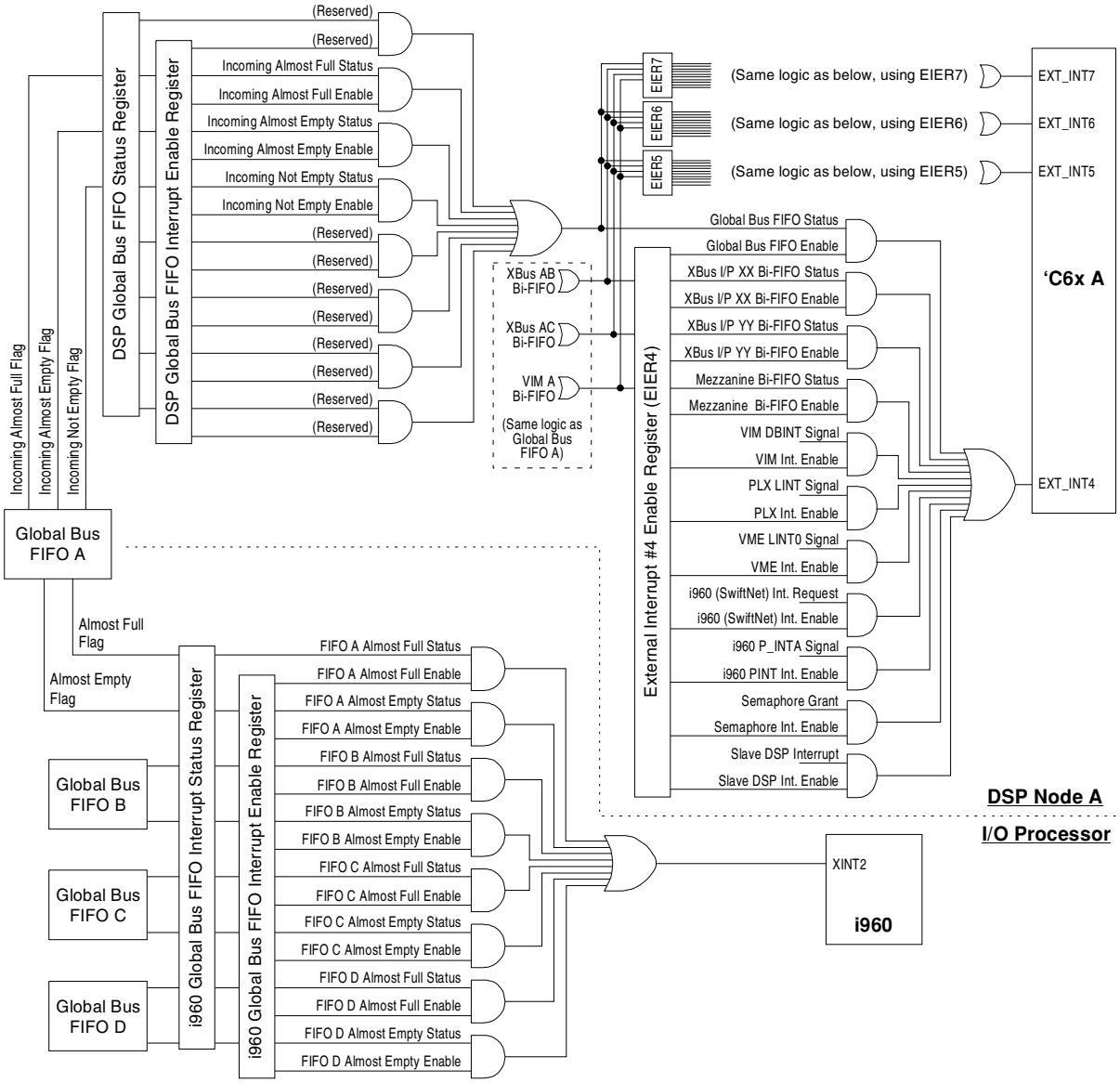


Figure 9–2: FIFO Interrupt Logic – Rev 0
(For boards with FPGA firmware revision 0)

9.5 FIFO Interrupts (continued)

The following figure illustrates the FIFO interrupt logic for Model 4292/4293 boards with FPGA firmware version 1 or greater, showing the DSP and i960 registers involved with the FIFO flag/interrupt flow. See Figure 9–4 on the next page for a detailed logic flow of the Global Bus FIFO A flags.

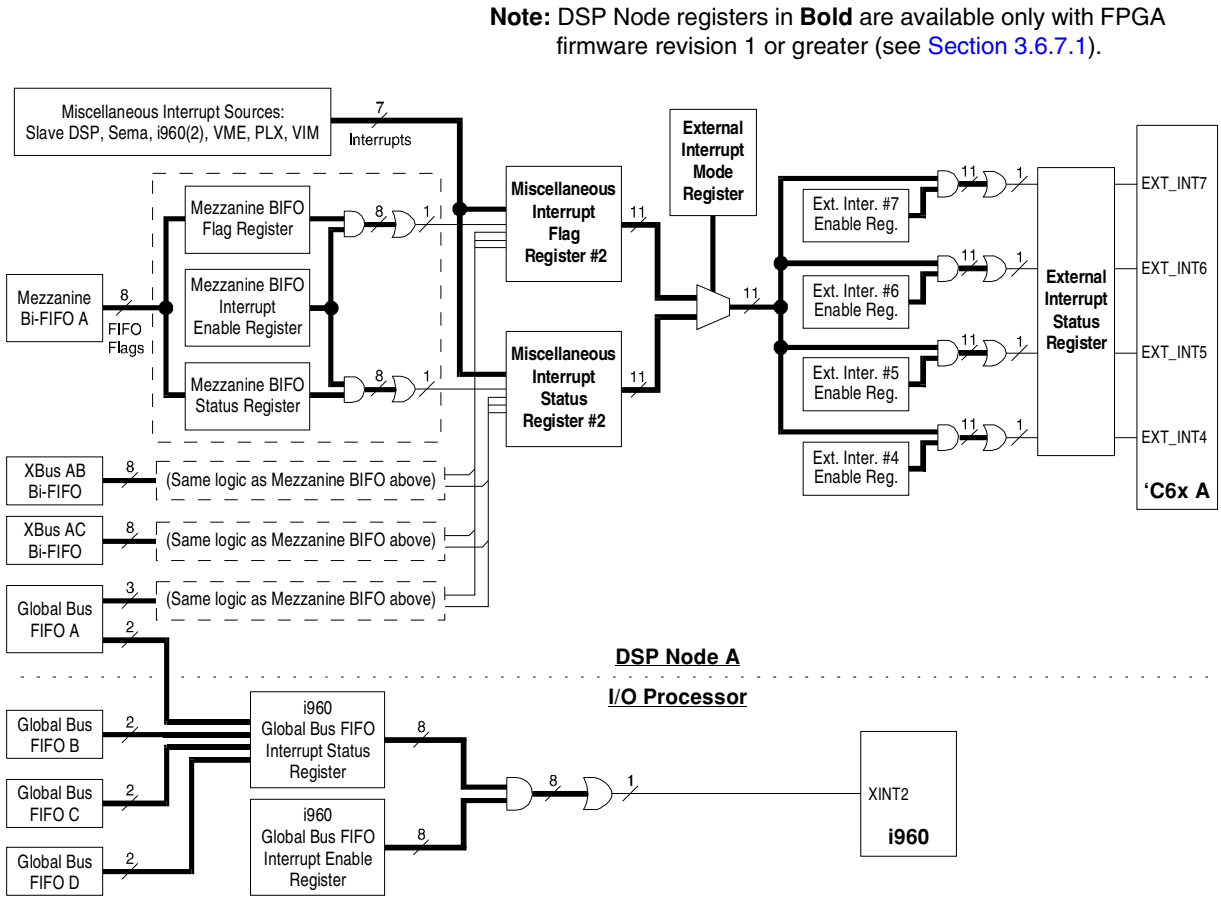


Figure 9–3: FIFO Interrupt Logic – Rev 1
(For boards with FPGA firmware revision 1 or greater)

For boards with FPGA firmware revision 0, see Figure 9–2 on the prior page.

9.5 FIFO Interrupts (continued)

The following figure details the FIFO interrupt logic for the Global Bus FIFO A for Model 4292/4293 boards with FPGA firmware version 1 or greater. See Figure 9–3 on the prior page for the overall FIFO interrupt logic.

Note: DSP Node registers in **Bold** are available only with FPGA firmware revision 1 or greater (see Section 3.6.7.1).

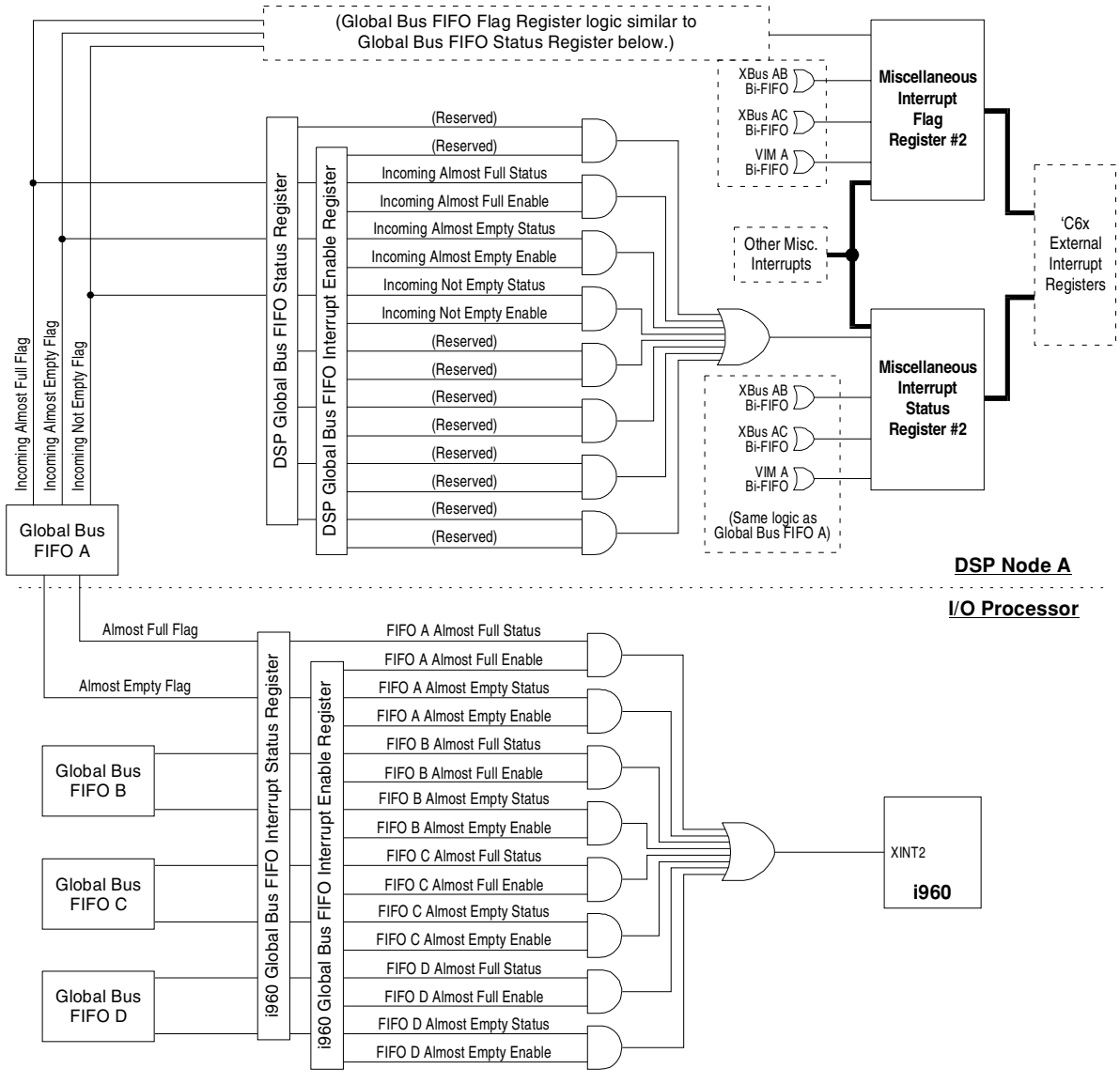


Figure 9–4: Global Bus FIFO Interrupt Logic – Rev 1
(For boards with FPGA firmware revision 1 or greater)

For boards with FPGA firmware revision 0, see Figure 9–2.

Chapter 10: FLASH Memory and Boot Firmware

10.1 Overview

The Model 4292/4293 provides separate FLASH memories for each 'C6x and i960 processor, which act as non−volatile storage and boot resource for the processors. When power is applied to the board, or when the board is reset, a boot sequence is initiated in each 'C6x processor and the i960 processor for initialization and self−test of the board. With the default board setup, the boot code stored in FLASH is loaded into the associated SDRAMs and executed to initialize the board.

The FLASH memories for each processor are shipped with default boot firmware, installed by Pentek. User boot code may be added to the FLASH for individual application requirements.

The following sections of this chapter describe the FLASH memories and default boot firmware operation:

- Section 10.2 describes the basic operation of each DSP Local FLASH memory by the associated 'C6x processor.
- Section 10.3 describes the basic operation of the i960 FLASH memory by the i960 processor.
- Section 10.4 describes the default boot operation of the 'C6x processors and the i960 processor, using the Pentek−supplied boot firmware, when the Model 4292/4293 is reset or power is applied to the board.
- Section 10.5 describes the Pentek−supplied firmware functions in each DSP FLASH memory. This firmware provides a variety of user functions that can be called by any device that has access to the Mailbox and Doorbell Registers of a PCI bus master PLX PCI 9054.
- Section 10.6 shows the firmware failure LED codes that may be illuminated if failures occur during boot firmware operation.
- Section 10.7 shows the procedures to add your user boot program to a DSP FLASH memory.

See also the reset logic diagram in Section 2.7 for illustration of the reset logic at power−up or reset of the board.

10.2 DSP Local FLASH Memory Operation

A 1–MByte local FLASH memory is provided for each 'C6x processor for non–volatile storage of initialization, self–test, and boot code. When the Model 4292/4293 is reset or power is applied to the board, each 'C6x copies the boot code from its local FLASH to its local SDRAM, and then executes the boot code from this SDRAM. The DSP FLASH memories are organized as separate boot, parameter, and user blocks, which can be write–protected using PCB jumpers (see Section 2.2.3, also see **CAUTION** below).

Table 10–1: DSP FLASH Memory Map

FLASH Memory Area	DSP Memory Address	Block Numbers
Boot Blocks (jumper write–protected)	**0x0100 0000 – 0x0100 3FFF**	0 – 1 (8 kBytes each)
Parameter Blocks (see **CAUTION**)	**0x0100 4000 – 0x0100 FFFF**	2 – 7 (8 kBytes each)
User Blocks (see **CAUTION**) *	**0x0101 0000 – 0x010F FFFF**	8 – 22 (64 kBytes each)

* FLASH memory address **0x0102 0000** (in Block 9), called 'Flash_Flag', is used by the default boot firmware to indicate the presence of user boot code, see Section 10.7.3

CAUTION

Each DSP FLASH is shipped with a default boot program and DSP firmware functions installed by Pentek. This Pentek code is stored in both boot blocks and parameter blocks. When you write–enable the user blocks with a PCB jumper (Section 2.2.3), you can still write–protect the boot blocks with a separate jumper but the Pentek boot code in the parameter blocks is **not** protected. **Users who tamper with the Pentek boot code do so at their own risk**.

10.3 i960 FLASH Memory Operation

The 4–MByte i960 FLASH memory provides non–volatile storage of programs for boot initialization. When the Model 4292/4293 is reset or power is applied to the board, the i960 copies the boot code from this FLASH to Global SDRAM, and then executes the boot code from Global SDRAM. The i960 FLASH is organized as separate boot and user blocks. The 64–kByte boot block is permanently protected using write–protect lock bits; the user blocks can be write–protected using a PCB jumper (see Section 2.2.5).

Table 10–2: i960 FLASH Memory Map

FLASH Memory Area	i960 Memory Address
User Blocks (jumper write–protected)	**0xFEC0 0000 – 0xFEFE FF2E**
Boot Block (permanent write–protected)	**0xFEFF 0000 – 0xFEFF FFFF**
Initialization Boot Record	**0xFEFF FF2F – 0xFEFF FF5F**

Note

The i960 FLASH is shipped with a default boot program installed by Pentek. This code is stored in the boot block and cannot be accessed by the user.

10.4 Firmware Boot Operation

When the Model 4292/4293 is reset or power is applied to the board, the 'C6x processors and the i960 processor each start individual boot sequences.

'C6x Boot Sections 10.4.3 and 10.4.4 describe the Pentek factory–installed 'C6x firmware boot operation. The boot program starts with Global Bus interrupts and cache memory disabled. Each 'C6x processor copies the boot code from its local FLASH to its local SDRAM, and then executes the boot code from SDRAM. If user boot code is stored in the FLASH (see Section 10.7), the boot sequence then loads and executes that code.

The first 64 kBytes of each SDRAM is reserved for use by the boot code. **Loading user DSP code into this area can cause unpredictable behavior**.

i960 Boot Section 10.4.5 describes the Pentek factory–installed i960 firmware boot operation. The i960 copies the boot code from its FLASH to Global SDRAM, and then executes the boot code from Global SDRAM. For a power–up of the board, the i960 boot sequence calls each 'C6x processor to run a Built–In Self Test (BIST)—for a board reset, the i960 boot sequence skips the 'C6x BIST function. Areas of Global SDRAM are reserved for i960 boot code use, as described in Section 10.4.1 below.

When complete, the i960 boot sequence calls the Monitor Program. Refer to Chapter 11 for description of this program.

10.4.1 Use of Global SDRAM During Boot

The boot sequences use fixed areas of Global SDRAM for controlling and storing results of boot initialization. The i960 boot sequence initializes this SDRAM to the data indicated in Table 10–3, on the next two pages. Many of the Global SDRAM memory locations are identified by symbolic names, such as 'Bypass_Flag', which are referenced in the subsequent flow diagrams and descriptions of the boot sequences.

NOTE: All memory addresses given for Global SDRAM in this chapter are relative to the PCI bus memory map (Section 6.3). If you access Global SDRAM from a VMEbus Master (using the Extended VMEbus Slave Memory Map, Section 7.3.2), the starting slave address of this memory is VME_slave + **0x41 0000**, which is the start of the 'User Data' area. (You cannot access the 'C6x and i960 processor data areas of Global SDRAM from the VMEbus.)

After boot initialization, Global SDRAM contains a copy of the Model 4292/4293 ID EEPROM code, and, if any VIM module(s) is (are) installed on the board, a copy of each VIM ID EEPROM code. Example contents of these ID EEPROMs are shown in Tables 10–5 and 10–4, respectively.

10.4 Firmware Boot Operation (continued)

10.4.1 Use of Global SDRAM During Boot (continued)

Table 10–3: Global SDRAM Boot Map

	PCI Bus Memory Address	Contents	
Master 'C6x Processors Section	0x1000 1000	0x0000 0000	User Boot Bypass_Flag, Processor A
	0x1000 1010	0x0000 0000	User Boot Bypass_Flag, Processor B
	0x1000 1020	0x0000 0000	User Boot Bypass_Flag, Processor C
	0x1000 1030	0x0000 0000	User Boot Bypass_Flag, Processor D
	0x1000 1040	Test_Count, Processor A (Section 10.4.3)	
	0x1000 1050	Test_Count, Processor B (Section 10.4.3)	
	0x1000 1060	Test_Count, Processor C (Section 10.4.3)	
	0x1000 1070	Test_Count, Processor D (Section 10.4.3)	
	0x1000 1080	0x1122 3344	Test Patterns, Processor A (Vec_Table)
	0x1000 1084	0xFFFF FFFF	
	0x1000 1088	0x5555 5555	
	0x1000 108C	0xAAAA AAAA	
	0x1000 1090	0x1234 5678	
	0x1000 1094	0x8765 4321	
	0x1000 1098	0x4292 0xxx	Boot code revision
	0x1000 109C	0x0000 0000	Reserved
	0x1000 10A0	0x1122 3344	Test Patterns, Processor B (Vec_Table)
	0x1000 10A4	0xFFFF FFFF	
	0x1000 10A8	0x5555 5555	
	0x1000 10AC	0xAAAA AAAA	
	0x1000 10B0	0x1234 5678	
	0x1000 10B4	0x8765 4321	
	0x1000 10B8	0x4292 1xxx	Boot code revision
	0x1000 10BC	0x0000 0000	Reserved
	0x1000 10C0	0x1122 3344	Test Patterns, Processor C (Vec_Table)
	0x1000 10C4	0xFFFF FFFF	
	0x1000 10C8	0x5555 5555	
	0x1000 10CC	0xAAAA AAAA	
	0x1000 10D0	0x1234 5678	
	0x1000 10D4	0x8765 4321	
	0x1000 10D8	0x4292 2xxx	Boot code revision
	0x1000 10DC	0x0000 0000	Reserved
	0x1000 10E0	0x1122 3344	Test Patterns, Processor D (Vec_Table)
	0x1000 10E4	0xFFFF FFFF	
	0x1000 10E8	0x5555 5555	
	0x1000 10EC	0xAAAA AAAA	
	0x1000 10F0	0x1234 5678	
	0x1000 10F4	0x8765 4321	
	0x1000 10F8	0x4292 3xxx	Boot code revision
	0x1000 10FC	0x0000 0000	Reserved

10.4 Firmware Boot Operation (continued)

10.4.1 Use of Global SDRAM During Boot (continued)

Table 10–3: Global SDRAM Boot Map (continued)

	PCI Bus Memory Address	Contents	
Master 'C6x Processors Section	0x1000 1100	VIM_Status, DSP Node A	
	0x1000 1104 – 0x1000 117C	VIM_Code, Node A VIM ID EEPROM (Table 10–4)	
	0x1000 1180	VIM_Status, DSP Node B	
	0x1000 1184 – 0x1000 11FC	VIM_Code, Node B VIM ID EEPROM (Table 10–4)	
	0x1000 1200	VIM_Status, DSP Node C	
	0x1000 1204 – 0x1000 127C	VIM_Code, Node C VIM ID EEPROM (Table 10–4)	
	0x1000 1280	VIM_Status, DSP Node D	
	0x1000 1284 – 0x1000 12FC	VIM_Code, Node D VIM ID EEPROM (Table 10–4)	
Slave 'C6x Processors Section (Model 4293 only)	0x1000 1300 – 0x1000 137F	Reserved	
	0x1000 1380 – 0x1000 1397	Test Patterns, Processor E (same as Proc. A Vec_Table)	
	0x1000 1398	0x4293 4xxx	Boot code revision
	0x1000 139C	0x0000 0000	Reserved
	0x1000 13A0 – 0x1000 13B7	Test Patterns, Processor F (same as Proc. B Vec_Table)	
	0x1000 13B8	0x4293 5xxx	Boot code revision
	0x1000 13BC	0x0000 0000	Reserved
	0x1000 13C0 – 0x1000 13D7	Test Patterns, Processor G (same as Proc. C Vec_Table)	
	0x1000 13D8	0x4293 6xxx	Boot code revision
	0x1000 13DC	0x0000 0000	Reserved
	0x1000 13E0 – 0x1000 13F7	Test Patterns, Processor H (same as Proc. D Vec_Table)	
	0x1000 13F8	0x4293 7xxx	Boot code revision
	0x1000 13FC	0x0000 0000	Reserved
i960 Processor Section	0x1000 1400	0xFEED 1960	i960_Status
	0x1000 1404	Temperature Sensor 1 Reading	
	0x1000 1408	Temperature Sensor 2 Reading	
	0x1000 140C	Temperature Sensor 3 Reading	
	0x1000 1410	Voltage Sensor 1 Reading	
	0x1000 1414	Voltage Sensor 2 Reading	
	0x1000 1418	Voltage Sensor 3 Reading	
	0x1000 141C	Voltage Sensor 4 Reading	
	0x1000 1420	Voltage Sensor 5 Reading	
	0x1000 1424 – 0x1000 143F	Reserved	
	0x1000 1440	0x00EE 1960	Board_Status
	0x1000 1444 – 0x1000 14BF	Board_Code, Board ID EEPROM (Table 10–5)	
	0x1000 14C0 – 0x1040 FFFF	Reserved for i960 Firmware	
User Data	0x1041 0000 – 0x10FF FFFF	User data, 32–bit data width	
	0x1041 0000 – 0x11FF FFFF	User data, 64–bit data width	

10.4 Firmware Boot Operation (continued)

10.4.1 Use of Global SDRAM During Boot (continued)

10.4.1.1 'C6x Processors Section

The boot sequences initialize separate sections of data for each Master 'C6x processor to the data shown in Table 10–3. This data includes the VIM_Status flag, and the contents of the VIM ID EEPROM, if a VIM is present, in the processor's VIM_Code area. On the Model 4293, the boot sequences also initialize separate sections of data for each Slave 'C6x processor.

A User Boot Bypass_Flag is provided for each Master 'C6x processor. This flag is cleared when Global SDRAM is initialized to allow the boot sequence to load and execute user boot code from the DSP's FLASH memory. The Bypass_Flag of each processor can be individually set to **0xC33C 3CC3** (for example, using SwiftNet's '**Boot Processors**' Target Add option) to bypass loading user boot code from its local FLASH (Section 10.7.4).

See Sections 10.4.3 and 10.4.4 for the Master and Slave 'C6x processor boot sequences.

10.4.1.2 i960 Processor Section

The i960 Processor Section of Global SDRAM contains data that is updated both at boot time and during normal board operation.

During boot, the i960 boot sequence sets the i960_Status flag and Board_Status flag, and copies the contents of the board's ID EEPROM into the Board_Code area, as shown in Table 10–3.

During normal operation, the i960 reads the board's voltages and temperatures at regular intervals. It stores the readings into the appropriate 'Temperature/Voltage Sensor N Readings' words of Global SDRAM (see Table 10–3) and then compares them with the last values read from the ID EEPROM. If any new reading is higher, the i960 stores the higher reading into the associated word in the board ID EEPROM (see Table 10–5 and Appendix B).

See Section 10.4.5 for the i960 processor boot sequence.

10.4.1.3 User Data

The Global SDRAM memory starting at PCI address **0x1041 0000** (VME_slave + **0x41 0000**, for access from the VMEbus using the Extended VMEbus Slave Memory Map, Section 7.3.2) is available for user applications or data.

10.4 Firmware Boot Operation (continued)

10.4.1 Use of Global SDRAM During Boot (continued)

Table 10–4: VIM ID EEPROM Example Code (Model 6230 shown)

EEPROM Word Location	GSDRAM Location*	Contents	Comments
00/01	+0x00	0x00EE C0DE	Valid data flag (VIM_Status word)
02/03	+0x04	0x6230 0000	Model, Model Extension (Extension currently unused)
04/05	+0x08	0x0354 2001	PCB number—this is taken from the VIM board
06/07	+0x0C	0x0001 FFFF	Assembly Rev, Option #1 Rev A = **0001**, Rev B = **0002**, etc.; Option #1 = option number, if no option field = **FFFF**
08/09	+0x10	0xFFFF FFFF	Option #2, Option #3
10/11	+0x14	0xFFFF FFFF	Option #4, Option #5
12/13	+0x18	0xFFFF FFFF	Option #6, Option #7
14/15	+0x1C	0xFFFF FFFF	Option #8, Option #9
16/17	+0x20	0xFFFF FFFF	Module–specific Code goes in locations 16 through 61
18/19	+0x24	0xFFFF FFFF	
20/21	+0x28	0xFFFF FFFF	
22/23	+0x2C	0xFFFF FFFF	
24/25	+0x30	0xFFFF FFFF	
26/27	+0x34	0xFFFF FFFF	
28/29	+0x38	0xFFFF FFFF	
30/31	+0x3C	0xFFFF FFFF	
32/33	+0x40	0xFFFF FFFF	
34/35	+0x44	0xFFFF FFFF	
36/37	+0x48	0xFFFF FFFF	
38/39	+0x4C	0xFFFF FFFF	
40/41	+0x50	0xFFFF FFFF	
42/43	+0x54	0xFFFF FFFF	
44/45	+0x58	0xFFFF FFFF	
46/47	+0x5C	0xFFFF FFFF	
48/49	+0x60	0xFFFF FFFF	
50/51	+0x64	0xFFFF FFFF	
52/53	+0x68	0xFFFF FFFF	
54/55	+0x6C	0xFFFF FFFF	
56/57	+0x70	0xFFFF FFFF	
58/59	+0x74	0xFFFF FFFF	
60/61	+0x78	0xFFFF FFFF	
62/63	+0x7C	0x0000 XXXX	Word 62 should always be cleared (**0000**). Word 63 contains the 16–bit checksum of words 0 – 61.

* Relative to the starting Memory Address for the applicable Node 'C6x Processor, Table 10–3

10.4 Firmware Boot Operation (continued)

10.4.1 Use of Global SDRAM During Boot (continued)

Table 10–5: Model 4292 ID EEPROM Example Code (Model 4292 shown)

EEPROM Word Location	G SDRAM Location	Contents	Comments
00/01	0x1000 1440	0x00EE 1960	Valid data flag (Board_Status word)
02/03	0x1000 1444	0x4292 0000	Model, Model Extension (Extension currently unused)
04/05	0x1000 1448	0x0354 2001	PCB number—this is taken from the board
06/07	0x1000 144C	0x0001 FFFF	Assembly Rev, Option #1 Rev A = **0001**, Rev B = **0002**, etc.; Option #1 = option number, if no option field = **FFFF**
08/09	0x1000 1450	0xFFFF FFFF	Option #2, Option #3
10/11	0x1000 1454	0xFFFF FFFF	Option #4, Option #5
12/13	0x1000 1458	0xFFFF FFFF	Option #6, Option #7
14/15	0x1000 145C	0xFFFF FFFF	Option #8, Option #9
16/17	0x1000 1460	0x0001 FFFF	PCB Rev, **FFFF** (Rev A = **0001**, Rev B = **0002**, etc.)
18/19	0x1000 1464	0x3042 921A	Firmware/FPGA Model Number and Revision for key components on the PCB—in each 32–bit word, the first seven hex digits are the Pentek Part Number, and the last hex digit is the Revision level of the part. For example, **0x3042 921A** refers to the Pentek part number 030.42921, revision A
20/21	0x1000 1468	0x3042 922A	
22/23	0x1000 146C	0x3042 923A	
24/25	0x1000 1470	0x3042 924A	
26/27	0x1000 1474	0x3042 925A	
28/29	0x1000 1478	0x3042 926A	
30/31	0x1000 147C	0x3042 927A	
32/33	0x1000 1480	0xXXXX XXXX	Temperature Sensor 1 Peak
34/35	0x1000 1484	0xXXXX XXXX	Temperature Sensor 2 Peak
36/37	0x1000 1488	0xXXXX XXXX	Temperature Sensor 3 Peak
38/39	0x1000 148C	0xXXXX XXXX	Voltage Sensor 1 Minimum
40/41	0x1000 1490	0xXXXX XXXX	Voltage Sensor 1 Maximum
42/43	0x1000 1494	0xXXXX XXXX	Voltage Sensor 2 Minimum
44/45	0x1000 1498	0xXXXX XXXX	Voltage Sensor 2 Maximum
46/47	0x1000 149C	0xXXXX XXXX	Voltage Sensor 3 Minimum
48/49	0x1000 14A0	0xXXXX XXXX	Voltage Sensor 3 Maximum
50/51	0x1000 14A4	0xXXXX XXXX	Voltage Sensor 4 Minimum
52/53	0x1000 14A8	0xXXXX XXXX	Voltage Sensor 4 Maximum
54/55	0x1000 14AC	0xXXXX XXXX	Voltage Sensor 5 Minimum
56/57	0x1000 14B0	0xXXXX XXXX	Voltage Sensor 5 Maximum
58/59	0x1000 14B4	0xFFFF FFFF	
60/61	0x1000 14B8	0xFFFF FFFF	
62/63	0x1000 14BC	0x0000 XXXX	Word 62 should always be cleared (**0000**). Word 63 contains the 16–bit checksum of words 0 – 61.

10.4 Firmware Boot Operation (continued)

10.4.2 Use of Universe II Mailbox Registers During Boot

During the 'C6x boot sequences, the Universe II Mailbox Registers (MBOXn) are used to select certain boot options for each Master 'C6x processor. The following table shows the address of the Mailbox Register that is used for each Master DSP. The addresses are shown as offsets relative to the base address of the Universe II Registers ('Universe_base') in the applicable memory map.

Table 10–6: Universe II Mailbox Registers Used During Boot

DSP	Universe II Register	Base Address Offset
DSP A	Mailbox 0 Register (MBOX0)	Universe_base + **0x0348**
DSP B	Mailbox 1 Register (MBOX1)	Universe_base + **0x034C**
DSP C	Mailbox 2 Register (MBOX2)	Universe_base + **0x0350**
DSP D	Mailbox 3 Register (MBOX3)	Universe_base + **0x0354**

See Section 5.4.3, 6.5.6, or 7.3.2.1 for the base (Universe_base) address from the i960, a PCI Bus device, or an external VMEbus Master, respectively.
See Section 7.3.1 for the base (Universe_base) address from an external VMEbus Master using the default VMEbus memory map.

To use these boot options, first store a code word (see below) into the Mailbox Register for the applicable DSP, and then re–boot the board.

The following table identifies the boot options that you can select for each Master 'C6x processor (see Section 10.4.3), and the code word you must store into the applicable Mailbox Register for that option. For the Model 4293, this boot option also applies to the associated Slave 'C6x processor.

Table 10–7: 'C6x User Boot Option Codes

Boot Option	Code Word
Run burn–in tests continuously	**0x00FA CD1A**
Bypass User boot code	**0xC33C 3CC3**
Erase User blocks in FLASH Memory	**0x0EEF DEAD**



You can only select one of these options at a time for the 'C6x boot sequence. Thus, if you need to use more than one option, you must re–boot multiple times, selecting a different option each time.

10.4 Firmware Boot Operation (continued)

10.4.3 Master 'C6x Processor Boot Sequence

The following describes the default firmware boot operation for each Master 'C6x processor. This boot sequence is illustrated in Figure 10–1 on page 146. Some of the steps refer to symbolic addresses in Global SDRAM, DSP FLASH memory, and Universe II Mailbox Registers.

1) The Master 'C6x processor boot program first clears its front panel processor LEDs (**0** and **1**).

2) The boot program then initializes resources on the 'C6x processor's local EMIF and Expansion Buses, including the PCI bus master. Memory regions are set aside for interrupt service routines, and all associated FIFOs are reset for normal operation (Section 9.2).

3) The Master 'C6x boot program then determines if a Slave 'C6x processor is present (only present on a Model 4293 board), and sets a flag indicating whether or not a Slave is present.

4) The 'C6x boot program checks its Universe II Mailbox Register (MBOXn) for a user boot option (Section 10.4.2).

   - If its Mailbox Register contains the code **0x00FA CD1A**, the 'C6x runs the processor Burn–in Tests. These tests run continually until a power–down or reset of the board.

   - If its Mailbox Register contains the code **0x0EEF DEAD**, the 'C6x erases the User blocks of its FLASH. (Note that write enable jumpers must be installed, Section 2.2.3.)

5) The 'C6x boot sequence checks the Flash_Flag location in its DSP FLASH memory.

   - If Flash_Flag contains the code **0xFEED CODE**, user boot code is available in its FLASH (Section 10.7.3). The boot program waits until the i960 boot sequence (Section 10.4.5) sets the i960_Status location in Global SDRAM to **0xFEED 1960**, and then continues to Step 6, on the next page.

   - If Flash_Flag does not contain the code **0xFEED CODE**, there is no user boot code in the FLASH and the boot program continues to Step 7, on the next page.

10.4 Firmware Boot Operation (continued)

10.4.3 Master 'C6x Processor Boot Sequence (continued)

6) The 'C6x boot sequence then checks both its boot Bypass_Flag in Global SDRAM and its Universe II Mailbox Register (MBOXn).

   - If either its Bypass_Flag or its Mailbox Register contains the code **0xC33C 3CC3**, the boot program bypasses loading the user code and continues to Step 7, below.
   - If the code **0xC33C 3CC3** is not set, the 'C6x loads the user boot code from its local FLASH into its local SDRAM and then executes this boot code, ending the default firmware boot sequence.

7) If user boot code is not available in FLASH or is bypassed, the Master 'C6x boot program checks if a Slave 'C6x processor is present. If a Slave is present (Model 4293 only), the boot program sets the Slave's Diagnostic code flag to the user boot option value from its Mailbox Register, and sets the Master Ready flag for the Slave boot sequence (Section 10.4.4).

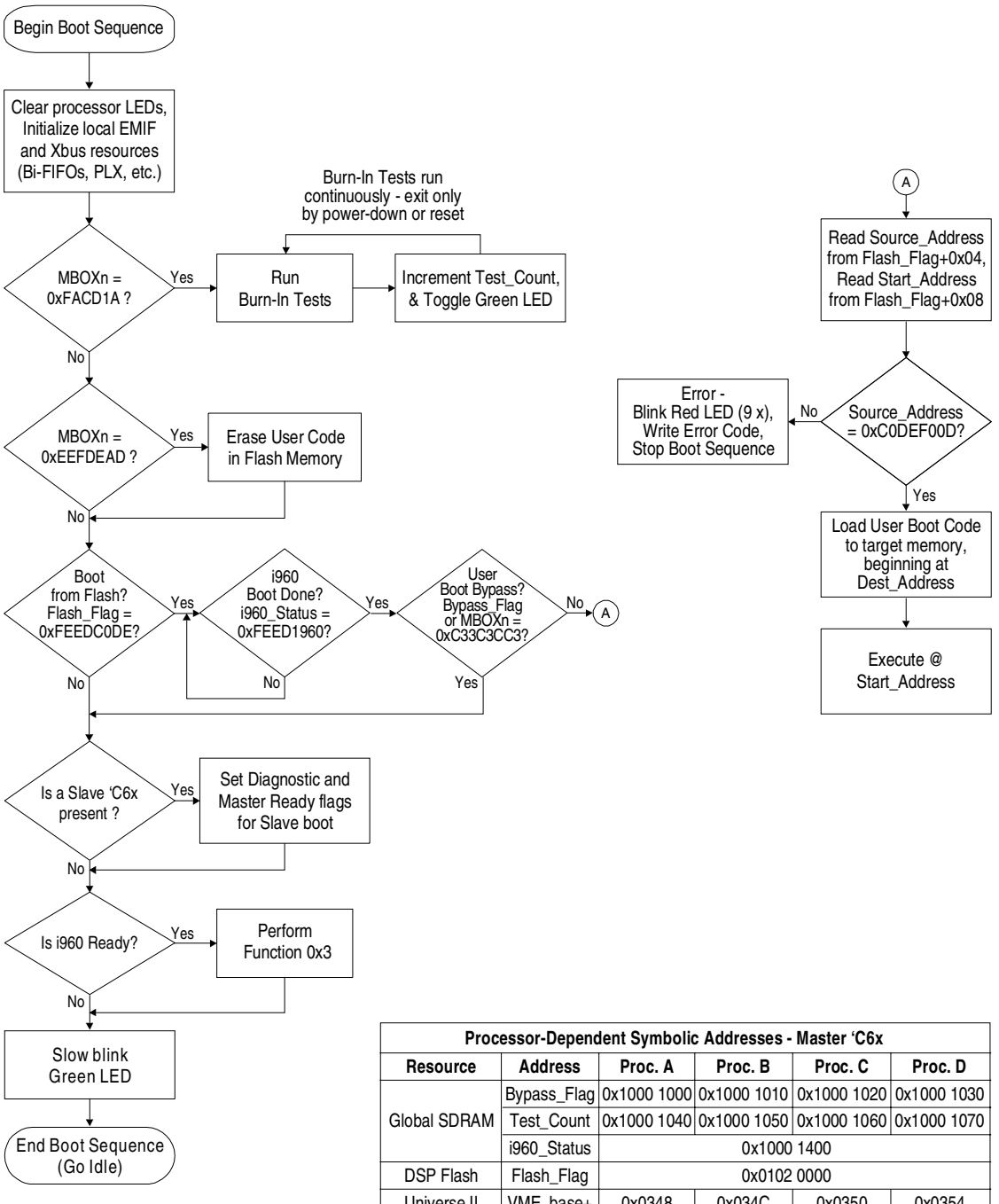
8) The 'C6x boot sequence checks if the i960 processor is ready. If the i960 is ready, the boot program initiates firmware function **0x3** to initialize SDRAM for this Master processor (see Figure 10–5).

9) The Master 'C6x boot sequence ends with slow blinking its front panel green **0** LED.

If a Master 'C6x processor detects certain errors during this boot sequence, that 'C6x repetitively blinks its front panel red **1** LED. See Section 10.6 for description of the firmware failure LED blink codes.

10.4 Firmware Boot Operation (continued)

10.4.3 Master 'C6x Processor Boot Sequence (continued)



Processor-Dependent Symbolic Addresses - Master 'C6x					
Resource	**Address**	**Proc. A**	**Proc. B**	**Proc. C**	**Proc. D**
Global SDRAM	Bypass_Flag	0x1000 1000	0x1000 1010	0x1000 1020	0x1000 1030
	Test_Count	0x1000 1040	0x1000 1050	0x1000 1060	0x1000 1070
	i960_Status	0x1000 1400			
DSP Flash	Flash_Flag	0x0102 0000			
Universe II Mailbox Reg.	VME_base+	0x0348 MBOX0	0x034C MBOX1	0x0350 MBOX2	0x0354 MBOX3

Figure 10–1: Master 'C6x Firmware Boot Sequence

10.4 Firmware Boot Operation (continued)

10.4.4 Slave 'C6x Processor Boot Sequence

NOTE: This Section is applicable only for the Model 4293.

The following describes the default firmware boot operation for each Slave 'C6x processor. This boot sequence is illustrated in Figure 10–2 on page 149. Some of the steps refer to a symbolic address in DSP FLASH memory, and to flags set by the Master 'C6x processor boot sequence (Section 10.4.3).

1) The 'C6x boot program first clears its front panel processor LEDs (**2** and **3**) and its Diagnostic and Ready flags from the Master 'C6x processor.

2) Next, the Slave 'C6x boot program copies its test patterns to its Local SDRAM, enables interrupts, and waits for the Master 'C6x processor boot sequence to set its Master Ready flag (Section 10.4.3, Step 7).

3) The Slave 'C6x boot program then checks its Diagnostic flag for one of the following two user boot options (set by the Master 'C6x processor boot sequence, Section 10.4.3, Step 7).

   - If the flag contains the code **0x00FA CD1A**, the 'C6x runs the processor Burn–in Tests. These tests run continually until a power–down or reset of the board.
   - If the flag contains the code **0xOEEF DEAD**, the 'C6x erases the User blocks of its FLASH. (Note that write enable jumpers must be installed, Section 2.2.3.)

4) The Slave 'C6x boot program checks the Flash_Flag location in its DSP FLASH memory.

   - If Flash_Flag contains the code **0xFEED CODE**, user boot code is available in its FLASH (Section 10.7.3). The boot program then continues to Step 5 on the next page.
   - If Flash_Flag does not contain the code **0xFEED CODE**, there is no user boot code in the FLASH, and the boot program continues to Step 6 on the next page.

10.4 Firmware Boot Operation (continued)

10.4.4 Slave 'C6x Processor Boot Sequence (continued)

5) The Slave 'C6x boot program then checks its Diagnostic flag (set by the Master 'C6x processor boot sequence, Section 10.4.3, Step 7).

   - If the flag contains the code **0xC33C 3CC3**, the boot sequence bypasses loading the user code and continues to Step 6 below.
   - If the code **0xC33C 3CC3** is not set, the 'C6x loads the user boot code from its local FLASH into its local SDRAM, and then executes this boot code, ending the default firmware boot sequence.

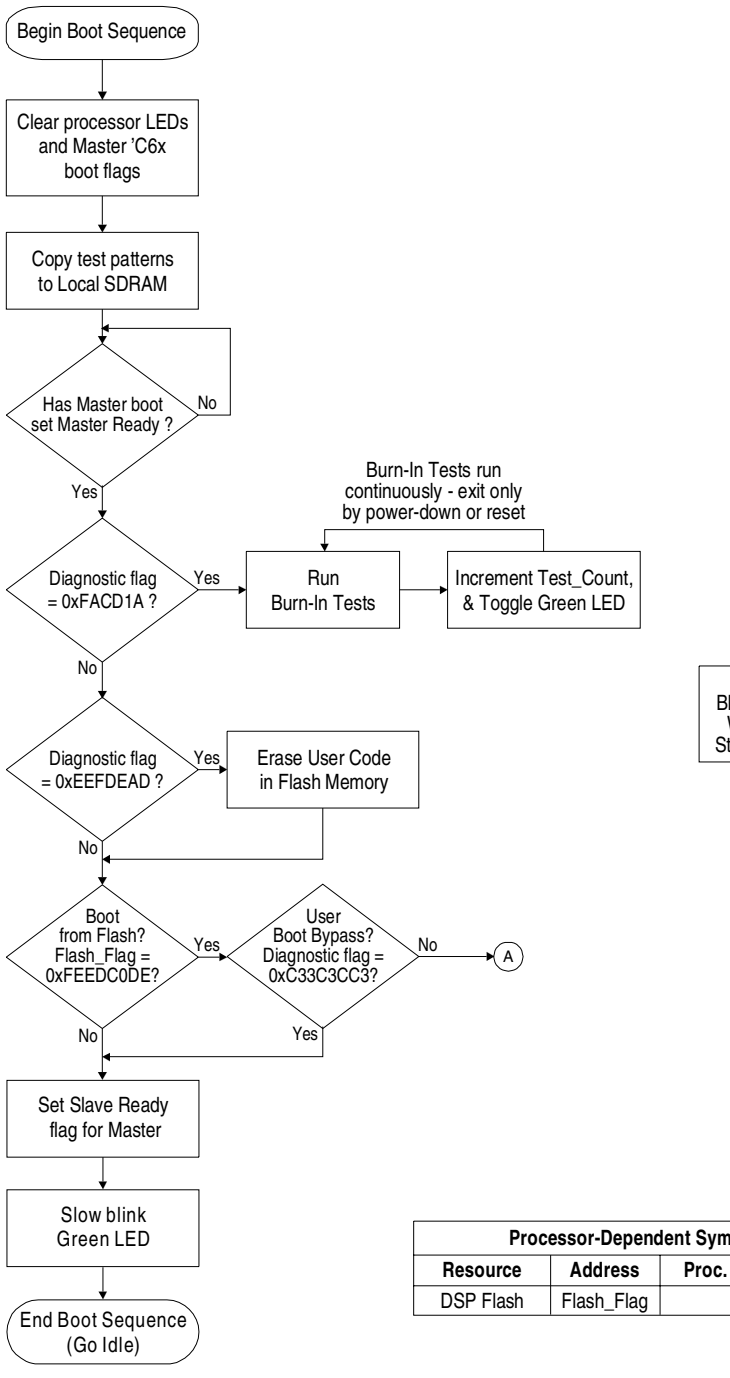
6) If user boot code is not available in FLASH or is bypassed, the Slave 'C6x boot program then sets the Slave Ready flag for the Master 'C6x processor boot sequence (Section 10.4.3).

7) The Slave 'C6x boot sequence ends with slow blinking its front panel green **0** LED.

If a Slave 'C6x processor detects certain errors during this boot sequence, that 'C6x repetitively blinks its front panel red **3** LED. See Section 10.6 for description of the firmware failure LED blink codes.

10.4 Firmware Boot Operation (continued)

10.4.4 Slave 'C6x Processor Boot Sequence (continued)



Processor-Dependent Symbolic Addresses - Slave 'C6x					
Resource	**Address**	**Proc. A**	**Proc. B**	**Proc. C**	**Proc. D**
DSP Flash	Flash_Flag	0x0102 0000			

Figure 10–2: Slave 'C6x Firmware Boot Sequence

10.4 Firmware Boot Operation (continued)

10.4.5 i960 Processor Boot Sequence

The following steps describe the default firmware boot operation for the i960 processor. This boot sequence is illustrated in Figure 10–3 on the next page. Refer to Table 10–3 for the Global SDRAM symbolic memory addresses.

1) At power–on or reset, the i960 first initiates a built–in self–test function (refer to the Intel i960® RM/RN I/O Processor Developer's Manual for details on this function). If this test fails, the i960 processor stops execution, and can only be restarted by a reset.

2) After i960 self–test passes, the i960 default firmware boot sequence starts. It determines the Model 4292/4293 configuration from the board ID EEPROM, initializes the PCI bus and local bus devices with a default memory map, and initializes Global SDRAM.

3) If the boot is initiated from a power–up 'cold start' of the board, the i960 interrupts each 'C6x processor to run its Built–In Self Test (BIST). Refer to Section 10.5.2 for description of the BIST sequence. If the boot is from a board reset, 'C6x BIST is bypassed.

4) The i960 boot sequence calls firmware function **0x3** in each Master 'C6x processor to copy the VIM ID EEPROM and 'C6x test patterns into Global SDRAM. See Figure 10–5 for illustration of this function.

5) The boot sequence copies the board's ID EEPROM (see Table 10–5) into the Board_Code area in Global SDRAM and verifies its checksum.

   - If the checksum is correct, the i960 writes the code **0x00EE 1960** into the Board_Status word in Global SDRAM.
   - If the checksum does not verify, the i960 writes **0x000B ADC5** into the Board_Status word in Global SDRAM.

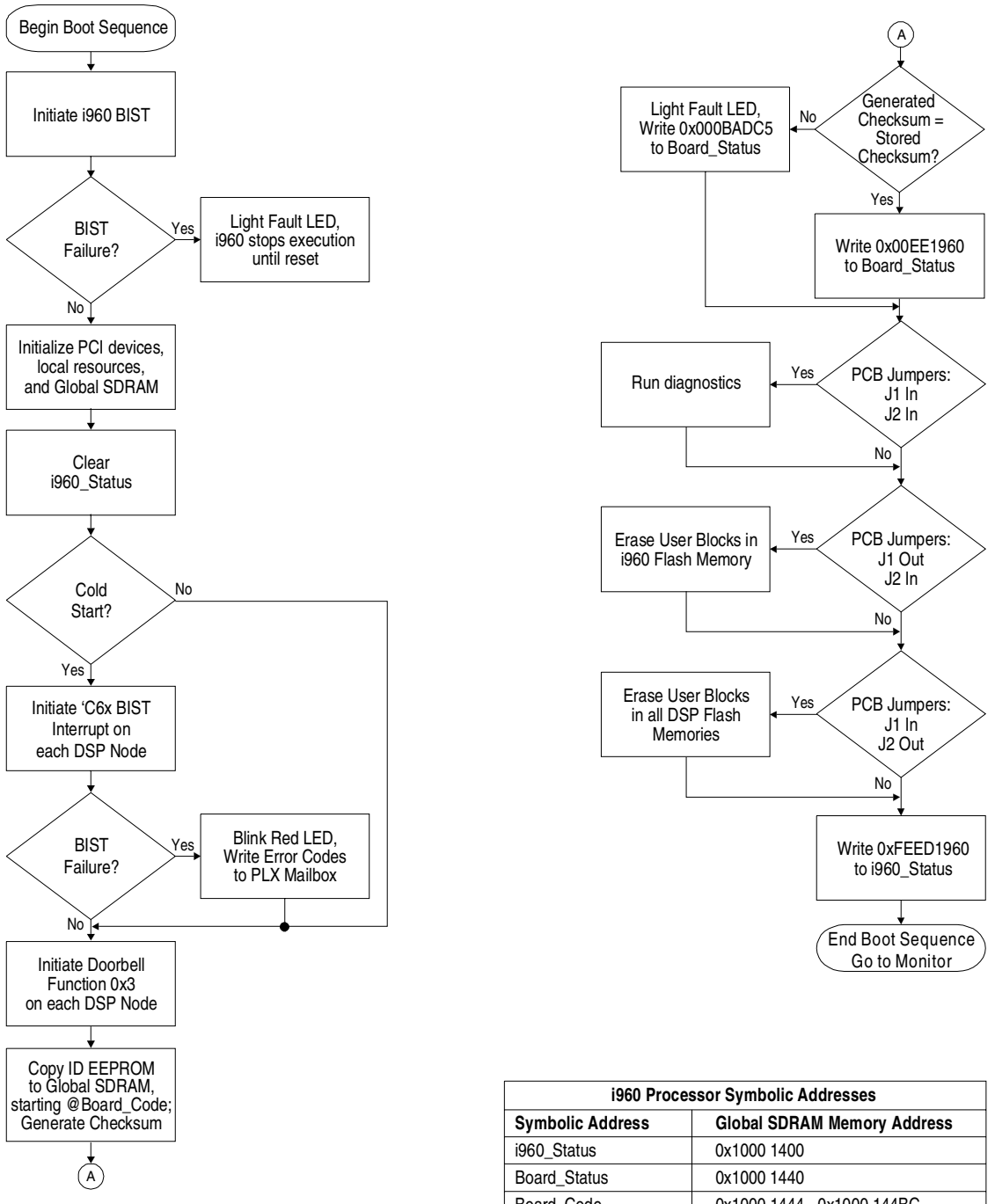
6) Depending on the setting of jumpers **J1** and **J2** (Section 2.2.1), the boot sequence runs the diagnostic self–tests, erases the user blocks of the i960 FLASH, or erases the user blocks of all DSP FLASH memories.

7) The i960 boot sequence writes the code **0xFEED 1960** into the i960_Status location in Global SDRAM, and then starts the Monitor program, described in Chapter 11.

If certain errors are detected during the boot sequences, the appropriate 'C6x processor repetitively blinks its front panel red **1** LED, and/or the i960 processor illuminates the front panel red **Fault** LED. See Section 10.6 for description of the firmware failure LED codes.

10.4 Firmware Boot Operation (continued)

10.4.5 i960 Processor Boot Sequence (continued)



i960 Processor Symbolic Addresses	
Symbolic Address	**Global SDRAM Memory Address**
i960_Status	0x1000 1400
Board_Status	0x1000 1440
Board_Code	0x1000 1444 - 0x1000 144BC

Figure 10–3: i960 Firmware Boot Sequence

10.5 DSP Firmware Functions

The default firmware supplied in each DSP FLASH memory provides several functions that can be called by any device that has access to the Mailbox and Doorbell Registers of each PCI bus master PLX PCI 9054. These firmware functions include 'C6x memory access and Master DSP FLASH memory operations. The following table shows these functions. Note that on the Model 4293 most of these firmware functions are available only in the Master 'C6x processors.

Table 10−8: DSP Firmware Function Summary

Function	Code	Arg. 1	Arg. 2	Arg. 3	Description
Read Data (Look)	**0x1**	'C6x Address	N/A	Data Read	Read data into argument 3 from the 'C6x address specified by argument 1
Write Data (Set)	**0x2**	'C6x Address	Data to Write	N/A	Write data from argument 2 into the 'C6x address specified by argument 1
Initialize 'C6x Section of Global SDRAM	**0x3** *	N/A	N/A	N/A	Check VIM module status, and initialize Global SDRAM to the 'C6x data shown in Table 10−3 (see Figure 10−5 for function flow)
Run Code (Go)	**0x4**	'C6x Address	N/A	N/A	Execute the code located at the 'C6x address specified by argument 1
Block Move from PCI Bus to Local	**0x7** *	Source Address	Destination Address	Move Length	Move the number of words specified by argument 3, from the PCI Bus address specified by argument 1, to the 'C6x address specified by argument 2
Boot Load	**0x8** *	Source Address	Code Execution Address	N/A	Load and execute user code from the 'C6x local SDRAM or DSP FLASH address specified by argument 1
FLASH Block Erase	**0x9** *	Block Number	N/A	N/A	Erase the 'C6x DSP FLASH user block identified by argument 1 (see Table 10−1 for block numbers – only blocks 8 to 22 can be erased)
Program FLASH	**0xa** *	Source Packet Address	N/A	N/A	Program the 'C6x DSP FLASH user blocks from the data packet specified by argument 1
Test Memory	**0xb** *	N/A	N/A	N/A	Initiate memory tests on the 'C6x processor

* On the Model 4293, these three firmware functions are available only in the Master 'C6x processors.

If an error is detected during the operation of firmware functions **0x3**, **0x9**, **0xa**, or **0xb**, the Master 'C6x processor repetitively blinks its front panel red **1** LED. See Section 10.6 for description of the firmware failure LED blink codes.

10.5 DSP Firmware Functions (continued)

10.5.1 Using Firmware Functions

The default DSP firmware functions listed in Table 10–8 can be called from the i960, PCI Bus, or VMEbus using the Mailbox and Doorbell Registers of each DSP's PLX PCI 9054. The register addresses are shown in the following table, as offsets to the base address of the PLX PCI 9054 Registers ('PLX_base') in the applicable memory map.

Table 10–9: PCI 9054 Mailbox and Doorbell Registers

PCI 9054 Register	Base Address Offset
Function Code (Mailbox Register 0)	PLX_base + **0x0040**
Argument 1 (MailBox Register 1)	PLX_base + **0x0044**
Argument 2 (MailBox Register 2)	PLX_base + **0x0048**
Argument 3 (MailBox Register 3)	PLX_base + **0x004C**
PCI–to–Local Doorbell Register	PLX_base + **0x0060**
Local–to–PCI Doorbell Register	PLX_base + **0x0064**

See Section 5.4.8, 6.5.1, or 7.3.2.2 for the base (PLX_base) address for the i960, a PCI Bus device, or an external VMEbus Master, respectively.

The following steps show how to call the default DSP firmware functions.

1) Write up to four parameters (a function code and up to three arguments, as specified in Table 10–8) to Mailbox Registers 0 to 3 of the desired node's PCI 9054 (for example, to call DSP B from the i960, the Mailbox Register 0 memory address is **0x8803 0040** [**0x8803 0000** + **0x0040**]).

2) Write **0x1** to the PCI–to–Local Doorbell Register to interrupt the Master 'C6x (on the Model 4293, write **0x2** to interrupt the Slave 'C6x).

3) The following actions occur in the 'C6x in response to the interrupt:

   a) The 'C6x reads the parameters from the Mailbox Registers, and then performs the requested function.

   b) When the function is complete, the 'C6x writes **0x80** (complete) or **0xFF** (error) to the Local–to–PCI Doorbell Register to interrupt the PCI bus.

4) Read the results from the Mailbox, if any, and then clear the interrupt by writing **0x80** (or **0xFF**, if error) to the Local–to–PCI Doorbell Register.

Refer to the PLX PCI 9054 Data Book for more information on using the Mail–box and Doorbell Registers.

10.5 DSP Firmware Functions (continued)

10.5.2 Using Built–In Self Test (BIST)

In addition to the firmware functions shown in Table 10–8, the default Master 'C6x firmware provides a Built–In Self Test (BIST) function. The following steps show how to call the 'C6x BIST from the i960, PCI Bus, or VMEbus.

1) Check that BIST is enabled, bit 7 of the PCI Built–In Self Test Register = 1. (The PCI BIST Register is at memory location **0x0Fh** in the 9054's PCI Configuration Registers. See PLX PCI 9054 Data Book.)

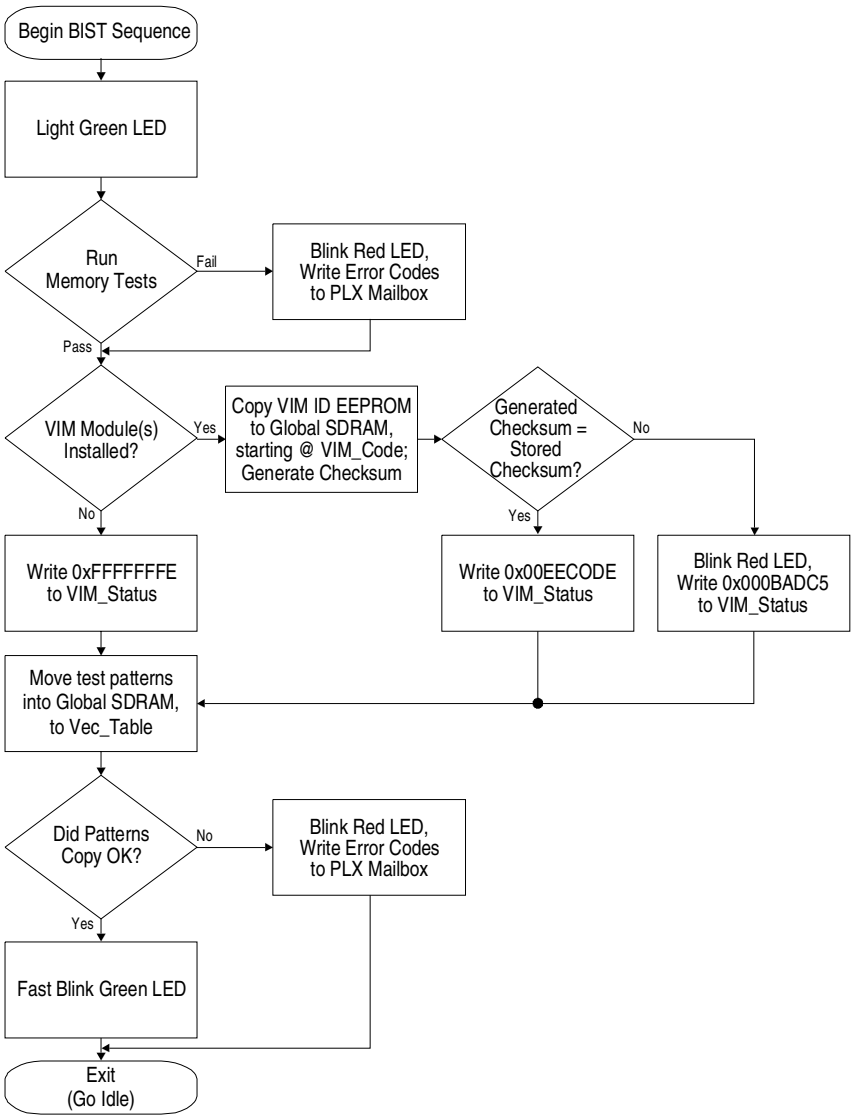


Figure 10–4: Master 'C6x BIST Interrupt Sequence

10.5 DSP Firmware Functions (continued)

10.5.2 Using Built–In Self Test (BIST) (continued)

2) Write bit 6 = 1 (**0x40**) to the PCI BIST Register, to interrupt the Master 'C6x.

3) The Master 'C6x performs the BIST sequence (Figure 10–4, prior page).

4) Wait until BIST finishes, as signaled by bit 6 of the PCI BIST Register = 0.

5) Read the BIST results from the PCI 9054 Mailbox Registers. The following table shows the Mailbox Registers used for these test results.

Table 10–10: BIST Failure Results

Results	PCI 9054 Register	Register Address
Error Code	Mailbox Register 0	PLX_base + **0x0040**
Failed Address	MailBox Register 1	PLX_base + **0x0044**
Expected Data	MailBox Register 2	PLX_base + **0x0048**
Actual Data	MailBox Register 3	PLX_base + **0x004C**

See Section 5.4.8, 6.5.1, or 7.3.2.2 for the base (PLX_base) address for the i960, a PCI Bus Master, or an external VMEbus Master, respectively.

Error Code — If an error is detected during BIST, the program stores an Error Code in bits D31 to D24 of Mailbox Register 0. The following error codes are possible:

0x00	No error
0xE2	Memory test 1 failure ('C6x local SDRAM)
0xE3	Memory test 2 failure ('C6x Internal Data memory)
0xE5	Global SDRAM test pattern move failure

Failed Address — For a BIST error, Mailbox Register 1 contains the memory address of the failure.

Expected Data — For a BIST error, Mailbox Register 2 contains the data that should have been stored at the failed address.

Actual Data — For a BIST error, Mailbox Register 3 contains the actual (erroneous) data from the failed address.

In addition, if an error is detected during BIST from either the 'C6x memory tests, VIM module checksum, or Global SDRAM 'C6x test pattern initialization, the red LED for that processor will blink (see Section 10.6).

10.5 DSP Firmware Functions (continued)

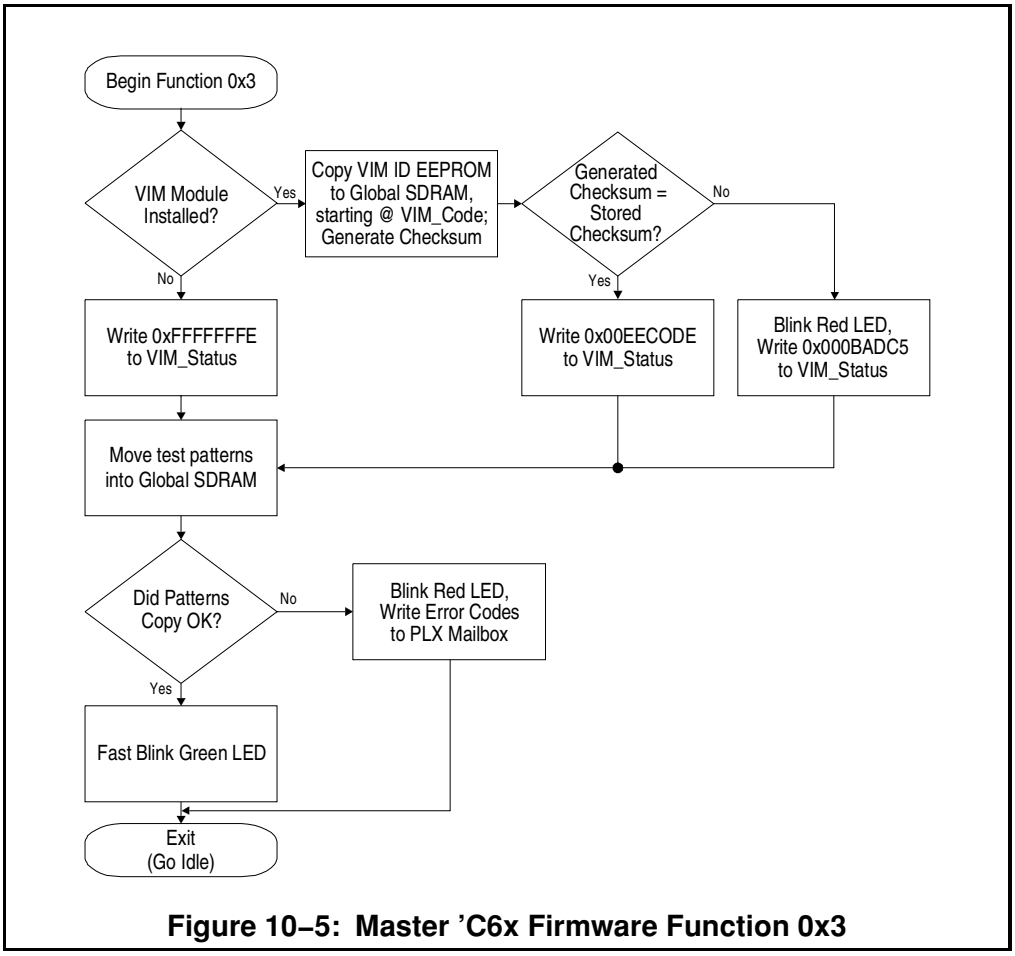


Figure 10–5: Master 'C6x Firmware Function 0x3

10.5.3 Checking for VIM Modules

When a Master 'C6x processor performs firmware function **0x3** (Figure 10–5) or BIST (Figure 10–4), it determines if a VIM module is installed using the VIM Module Present bit (bit D2, DSP Board Status Register, Section 3.6.7).

- If no VIM module is installed, the Master 'C6x writes the code **0xFFFF FFFE** to its VIM_Status word in Global SDRAM.
- If a VIM module is installed, the Master 'C6x copies the contents of the VIM ID EEPROM (see Table 10–4) into its VIM_Code area in Global SDRAM and verifies its checksum.
    - If the checksum is correct, the 'C6x writes **0x00EE CODE** to its VIM_Status word in Global SDRAM.
    - If the checksum does not verify, the 'C6x writes **0x000B ADC5** to its VIM_Status word in Global SDRAM.

10.6 Firmware Test Failure LED Codes

On the Model 4292/4293 PCB, to the left of the blank front panels, is a group of four LEDs associated with each DSP Node's processors (see figure at the right). (The LEDs are mounted on the solder side of the PCB.) The top group is for Node A, the next is Node B, the next is Node C, and the bottom group is Node D. The upper two LEDs of each group, labeled **0** (green) and **1** (red) are associated with the Master 'C6x processor; LEDs labeled **2** (green) and **3** (red) are for the Slave 'C6x processor.

If all firmware tests for a 'C6x processor pass, that processor's green LED (**0** or **2**) blinks at a rate of about five times per second. If a test fails, the processor's red LED (**1** or **3**) blinks repetitively, remains off for about a second, and repeats this cycle until you remove power or press the front panel **Reset** button. The red LED blink failure codes are shown in the following table.

Table 10–11: 'C6x Red LED Blink Failure Codes

Firmware Function	Number of Fast Blinks	'C6x Failure
BIST, or Memory Tests (**0xb**)	1	Memory Test 1 ('C6x SDRAM)
	2	Memory Test 2 ('C6x I.D. memory)
'C6x Boot Resource Initialization	3 *	PLX PCI 9054 register write/read
Block Erase (**0x9**)	5 *	'C6x FLASH block erase
	6 *	'C6x FLASH erase verify
Program FLASH (**0xa**)	7 *	'C6x FLASH write data
BIST or GI. SDRAM Initialization (**0x3**)	8 *	Global SDRAM 'C6x test pattern
'C6x User Boot	9	User Boot Code format failure
'C6x Boot Option	10	Erase User FLASH attempted without proper code in Universe II Mailbox
Slave 'C6x BIST	11	Slave 'C6x BIST failure
* Master 'C6x processor only		

On the front panel center plate, to the right of the RS–232/Ethernet connector, are three LEDs, one red and two green (see figure at the right). These are associated with the i960 and the RS–232/Ethernet communications. The i960 illuminates the red **Fault** LED when a fault is detected during boot initialization (or when the LED bit is set in the i960 Control Register, Section 5.5.6).

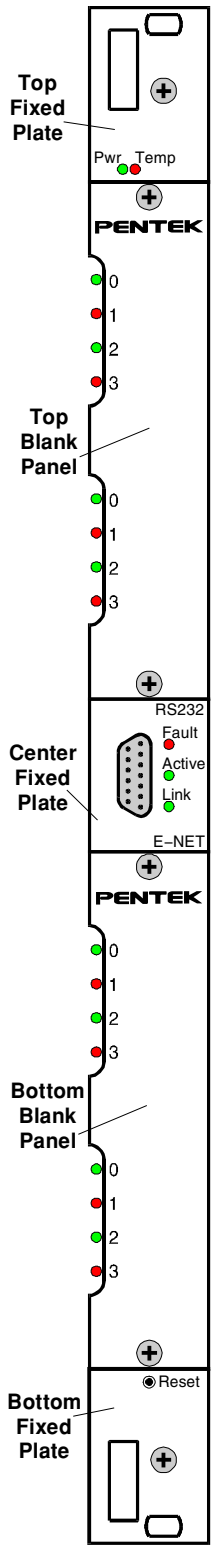


Figure 10–6: Model 4292/4293 Front Panel

10.7 User Boot Programs

The Model 4292/4293 can boot using user−created code. Custom boot code may be stored into any DSP's Local FLASH memory. During boot operation using the Pentek default boot firmware, each 'C6x boot sequence checks for user boot code at the end of the standard boot sequences, and then executes that code if it is present in the respec−tive FLASH (see Section 10.4.3). The following subsections describe how to prepare and install your boot code and set up the proper flags for executing that code.

10.7.1 Converting COFF Files into Intel HEX Format

Program files to be written to any DSP FLASH must be in Intel HEX format. A program for converting COFF (Common Object File Format) program files into this format, named **hex30**, is provided by Texas Instruments as a part of their Optimizing 'C' Compiler package.

You can convert COFF files to Intel HEX format by creating an appropriately named linker command file for TI's **hex30** program. The COFF file to be con−verted should have the extension **.out** or **.x92**, and the linker command file should have the extension **.cmd**. The following listing is an example of such a linker command file, called **boot92.cmd**.

```
/*********************************************/
/*         Boot build for C6x FLASH          */
/*********************************************/
dim.x92                 /* Input file         */
-o dim92.hex            /* HEX output file    */
-i                      /* intel format       */
-memwidth 8             /* 8-bit memory       */
-boot                   /* Convert all .sect  */
-bootorg     0h         /* Boot source addr   */
-cg   3def4000h         /* Global mem conf.   */
-cl   3def4210h         /* Local mem conf.    */
-e    04000c54h         /* PC after load addr */
-iack 4003fc00h         /* IACK dummy read    */
-ivtp 4003fc00h         /* IV pointer         */
-tvtp 4003fc00h         /* TV pointer         */
```

The command syntax to create an Intel HEX file **dim92.hex** from the COFF file **dim.x92**, using the above linker command file **boot92.cmd**, would be:

```
hex30 boot92.cmd<enter>
```

To modify the example linker command file shown above to work with your COFF files, substitute the name of your COFF file on the **/* Input file */** line, and the name for your HEX file after the **-o** on the **/* HEX output file */** line. Examine the program map file created when your COFF file was com−piled to find the program entry point, and substitute that address after the **-e** on the **/* PC after load addr */** line.

10.7 User Boot Programs (continued)

10.7.2 Erasing DSP FLASH Memories

Before a user HEX file can be loaded into a DSP FLASH memory, you should first erase any existing user code in FLASH.

The first 64 kBytes of each DSP FLASH (boot blocks and parameter blocks) are reserved for Pentek–supplied boot firmware. The Pentek boot code is known good code that will put the associated 'C6x processor into a known good state.

CAUTION

This Pentek boot code should not be erased nor modified. If this code is modified, Pentek technical support personnel cannot know the state of the processors after boot, and may not be able to predict how the board will behave. **Users who tamper with the Pentek boot code do so at their own risk**.

You can erase the user blocks of a DSP FLASH at boot time using the Pentek default boot firmware, as follows.

- To erase all user blocks on ALL DSP FLASH memories at boot, install PCB jumpers for 'Erase DSPs FLASH User Blocks' power–up mode before rebooting the board—see Section 2.2.1 for these jumper settings.

- To erase all user blocks on an individual node DSP FLASH memory at boot, store the code **0x0EEF DEAD** into the applicable DSP's Universe II Mailbox Register before rebooting the board—see Section 10.4.2 for the applicable Mailbox Register for each DSP.

You can also erase user blocks of each Master DSP FLASH memory individually, from block 8 to 22, using the Master 'C6x firmware function **Block Erase** (**0x9**). Refer to the procedure in Section 10.5.1 to erase a DSP FLASH with this function, and to Section 10.2 for identification of the block numbers for each region of DSP FLASH.

NOTE: To erase any user blocks in any DSP FLASH memory, you must first install PCB jumpers to enable write operations to the user blocks in FLASH—see Section 2.2.3 for these jumper settings.

CAUTION

When you write–enable the User blocks with a PCB jumper, you can still write–protect the Boot block with a separate jumper, but the Pentek code in the Parameter blocks is NOT protected (see Section 10.2).

10.7 User Boot Programs (continued)

10.7.3 Loading HEX Files into DSP FLASH

The procedure described in Section 10.7.1 creates a HEX file that is suitable for writing into any DSP FLASH memory. This HEX file can be downloaded to a DSP FLASH by the associated 'C6x processor (see Section 10.2).

You can write user HEX files directly to any Master DSP FLASH memory user blocks with the Master 'C6x firmware function **Program FLASH** (**0xa**), by any device that has access to that node's PLX PCI 9054 registers. Refer to the procedure in Section 10.5.1 to load your HEX files using this function. Argument 1 for the Program FLASH function is a pointer to a data packet for the source HEX file. The data packet is four words—the first word is the 'C6x address of the source (HEX) file, the second word is the FLASH address to store the code, the third word is the length minus 1 (in 32–bit words) of the boot code, and the last word is the execution starting address of the boot code.

NOTE: To write to any user block in the DSP FLASH, you must first install jumpers on the PCB to enable write operations to the user blocks in FLASH—see Section 2.2.3 for these jumper settings.

The 'C6x boot sequence checks the Flash_Flag word in its local FLASH memory to determine if user boot code is present (Section 10.4.3). To indicate that there is user code in the FLASH, you must store the code **0xFEED CODE** into the Flash_Flag location ('C6x memory address **0x0102 0000**).

10.7.4 Bypassing DSP User Boot Code

If you wish to bypass the user boot code in DSP FLASH, store the code word **0xC33C 3CC3** into the applicable Master DSP's Bypass_Flag word of Global SDRAM (Section 10.4.1) or Universe II Mailbox Register (Section 10.4.2) before you re–boot the 'C6x processor (for the Model 4293, this code also bypasses the associated Slave). The addresses of these resources are as follows:

Processor	Global SDRAM *	Universe II Mailbox Register
Processor A	**0x1000 1000**	MBOX0: VME_base + **0x0348**
Processor B	**0x1000 1010**	MBOX1: VME_base + **0x034C**
Processor C	**0x1000 1020**	MBOX2: VME_base + **0x0350**
Processor D	**0x1000 1030**	MBOX3: VME_base + **0x0354**

* PCI Bus Address

If you are using Pentek's SwiftNet Host Software with the Model 4292/4293, you can set the user boot Bypass_Flag in Global SDRAM to **0xC33C 3CC3** by setting SwiftNet's '**Boot Processors**' Target Add option to **disable** booting (refer to the applicable ***SwiftNet Host Software User's Guide***). This option sets the Bypass_Flags in Global SDRAM for all 'C6x processors.

Chapter 11: The Monitor Program

11.1 Overview

The easiest and most direct path for tailoring the Model 4292/4293's features to the requirements of your application is using the RedBoot™ Monitor, which is accessible from the board's RS–232/Ethernet serial port (see Section 2.6.6 for the RS–232/Ethernet connector). Pentek has modified Red Hat's Embedded Debug and Bootstrap program (RedBoot) to run on the Model 4292/4293 i960 processor.

The Monitor provides the following capabilities:

- Configuring VME Slave access
- Configuring VME Master access
- Configuring VME interface parameters such as Bus Request Level and Arbitration Modes
- Configuring Ethernet network interface
- Viewing Global SDRAM and i960 FLASH memory locations
- Loading and executing programs over RS–232 or Ethernet interfaces
- Loading and managing FLASH images

11.2 Connecting to the Serial Port

Connect the Model 4292/4293 RS–232/Ethernet serial port (see Section 2.6.6 for the serial port connector) either to an RS–232 terminal or to an RS–232 serial port on any PC or workstation capable of running terminal emulation software. The Monitor's factory default configuration is for RS–232 operation. You can enable Ethernet operation using the Monitor **fconfig** command defined later in this chapter (Section 11.5.20).

The default RS–232 parameters are:

```
Baud Rate: 9600
Parity: None
Data Bits: 8
Stop Bits: 1
```

Once connected, you can modify the baud rate using the Monitor **fconfig** command defined later in this chapter (Section 11.5.20).

11.3 Starting the Monitor's User Interface

After resetting the Model 4292/4293 board, RedBoot automatically loads a Command Processor application. The following listing is a sample of the startup dialog:

```
RedBoot(tm) bootstrap and debug environment - built 11:57:07, Jun 27 2001
Platform: PENTEK 4292 (I960 RN)
Copyright (C) 2000, 2001, Red Hat, Inc.

RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
FLASH: 0xfec00000 - 0xff000000, 32 blocks of 0x00020000 bytes each.
No PMC Module found
IP: 192.168.200.193, Default server: 192.168.200.68
== Executing boot script in 1.000 seconds - enter ^C to abort
RedBoot> fis load cmdproc
RedBoot> go

Pentek 4292 Firmware - built 12:44:00, Jun 22 2001
Platform: PENTEK 4292 (I960 RN)
RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
Pentek 4292>
```

The Command Processor application has a limited subset of the RedBoot Monitor command set. While running the Command Processor, you can easily return to the RedBoot command prompt by typing **reset** at the **Pentek 4292>** prompt, followed by entering <Ctrl C> before it executes the boot script. An example of this is shown below:

```
Pentek 4292> reset
... Resetting.
+
RedBoot(tm) bootstrap and debug environment - built 11:57:07, Jun 27 2001
Platform: PENTEK 4292 (I960 RN)
Copyright (C) 2000, 2001, Red Hat, Inc.

RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
FLASH: 0xfec00000 - 0xff000000, 32 blocks of 0x00020000 bytes each.
No PMC Module found
IP: 192.168.200.193, Default server: 192.168.200.68
== Executing boot script in 1.000 seconds - enter ^C to abort
^C
RedBoot>
```

11.4 Command Processor Commands

The Command Processor application provides a limited operator command set. All references to Global SDRAM and i960 FLASH memories in these commands are based on the i960 Processor Memory Map in Section 5.3.

The following Command Processor commands and their arguments are also available in the RedBoot Monitor command set, Section 11.5:

dump, **reset**, **version**, **fill**, **help**, **edit**, **pedit**, **pdump**

NOTE: After you reset or boot the Model 4292/4293, the Command Processor is also available to user applications running in the 'C6x processors or from the VME bus. See Chapter 12 for the procedures to use this capability.

11.4.1 dump

Syntax: **dump [-b <*location*>] [-l <*length*>]**

Description: Displays the current contents of a region in memory, in Hex format.

Arguments:

location Starting memory address to display, as viewed by the i960 processor. If this argument is omitted, the address following the last address displayed from a previous **dump** command is the starting address.

length Number of 32–bit words to display. If this argument is omitted, 32 words are displayed.

Example: Display memory range **0xA0300000** to **0xA0300010**.

```
Pentek 4292> dump -b 0xA0300000 -l 4

0xA0300000: 00000000 00000000 00000000 00000000
Pentek 4292>
```

11.4 Command Processor Commands (continued)

11.4.2 reset

Syntax: **reset**

Description: Resets the i960 processor.

Arguments: None

Example:

```
Pentek 4292> reset
... Resetting.
+
RedBoot(tm) bootstrap and debug environment - built 11:57:07, Jun 27 2001
Platform: PENTEK 4292 (I960 RN)
Copyright (C) 2000, 2001, Red Hat, Inc.

RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
FLASH: 0xfec00000 - 0xff000000, 32 blocks of 0x00020000 bytes each.
No PMC Module found
IP: 192.168.200.193, Default server: 192.168.200.68
== Executing boot script in 1.000 seconds - enter ^C to abort
RedBoot> fis load cmdproc
RedBoot> go

Pentek 4292 Firmware - built 12:44:00, Jun 22 2001
Platform: PENTEK 4292 (I960 RN)
RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
Pentek 4292>
```



After a reset, RedBoot automatically loads and executes the Command Processor application (the '**fis load cmdproc**' and '**go**' boot script commands, above). Alternately, you can return to the RedBoot command prompt by entering <Ctrl C> before the reset sequence executes this boot script (see Section 11.3).

11.4 Command Processor Commands (continued)

11.4.3 version

Syntax: `version`

Description: Displays the current revision of the Command Processor.

Arguments: None

Example: Display Command Processor version information.

```
Pentek 4292> version

Pentek 4292 Command Processor - built 12:44:00, Jun 22 2001
Platform: PENTEK 4292 (I960 RN)
RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
Pentek 4292>
```

11.4.4 fill

Syntax: `fill -b <location> -l <length> -v <value>`

Description: Fills the contents of memory.

Arguments:

location Starting memory address to fill, as viewed by the i960 processor.

length Number of 32–bit words to write.

value 32–bit data value to write into the specified words of memory.

Example: Initialize memory region **0xa0800000** – **0xa080007F**.

```
Pentek 4292> fill -b 0xa0800000 -l 0x20 -v 0xa5a5a5a5v
Pentek 4292>
```

11.4 Command Processor Commands (continued)

11.4.5 help

Syntax: `help`

Description: Displays a list of Command Processor commands and their arguments.

Arguments: None

Example:

```
Pentek 4292> help

Display Current Voltage and Temperature Values
   display -c -p
Display (hex dump) a range of memory
   dump -b <location> [-l <length>]
Edit the contents of memory
   edit -b <location> -v <value> [-s <size>]
Fill the contents of memory
   fill -b <location> -l <length> -v <value>
Help about help?
   help [<topic>]
Modify memory
   modify -b <location> -s <size>
Display (hex dump) PCI Configuration Space
   pdump -d <device> [-f <function>] [-l <length>]
Edit a register in PCI Configuration Space
   pedit -d <device> -f <function> -o <offset> -v <value>
Reset the system
   reset
Display firmware version information
   version
Pentek 4292>
```

11.4 Command Processor Commands (continued)

11.4.6 edit

Syntax: **`edit -b <location> -v <value> [-s <size>]`**

Description: Edits the contents of memory.

Arguments:

`location` Memory address to modify, as viewed by the i960 processor.

`value` Data value to write into the memory address.

`size` Number of bytes to write. This argument must be set to **`1`**, **`2`**, or **`4`** bytes. If this argument is omitted, a default of **`4`** bytes is used.

Example: Edit four bytes in Global SDRAM.

```
Pentek 4292> edit -b 0xa0800000 -v 0x12345678 -s 4
```

11.4.7 pedit (This command is available only in RedBoot version 1.6 or greater)

Syntax: **`pedit -d <device> -f <function> -o <offset> -v <value>`**

Description: Edits the contents of a register in PCI Configuration Space.

Arguments:

`device` Reference number of the PCI device to access:

`1` – Tundra Universe II VME Interface
`2` – PMC Site
`3` – Intel 82559 Ethernet Controller
`4` – Intel i960 PCI Bridge
`5` – Node A PLX PCI 9054
`6` – Node B PLX PCI 9054
`7` – Node C PLX PCI 9054
`8` – Node D PLX PCI 9054

`function` Reference number of the function to access in a multi–function device. Range is **`0`** to **`7`**.

`offset` Byte offset of the register to edit, from the start of the device's PCI configuration space.

`value` Data value to write into the register.

Example: Modify the PMC's command/status register at offset 4.

```
Pentek 4292> pedit -d 2 -f 0 -o 4 -v 0x7
```

11.4 Command Processor Commands (continued)

11.4.8 pdump (This command is available only in RedBoot version 1.6 or greater)

Syntax: **pdump -d <*device*> [-f <*function*>] [-l <*length*>]**

Description: Displays the contents of a device in PCI Configuration Space, in Hex format.

Arguments:

device Reference number of the PCI device to access. The PCI devices available are:

1 – Tundra Universe II VME Interface
2 – PMC Site
3 – Intel 82559 Ethernet Controller
4 – Intel i960 PCI Bridge
5 – Node A PLX PCI 9054
6 – Node B PLX PCI 9054
7 – Node C PLX PCI 9054
8 – Node D PLX PCI 9054

function Reference number of the function to access in a multi−function device. Range is **0** to **7**. When this argument is omitted, the last specified function is used. The default is **0**.

length Number of 32−bit words to display. When this argument is omitted, the last specified length is used. The default is 16 words.

Example: Dump the PMC's configuration space registers for the RACEway PMC (Option 005).

```
Pentek 4292> pdump -d 2

00000000: 00010034 04800007 06040002 00010000
00000010: f0000008 18600000 00000100 04800111
00000020: 20001870 fff0f000 00000000 00000000
00000030: 00000000 00000000 18700000 00000000
Pentek 4292>
```

11.4 Command Processor Commands (continued)

11.4.9 modify

(Note: This command is unique to the Command Processor, and is not provided in the RedBoot Monitor command set.)

Syntax: **`modify -b <`*`location`*`> [-s <`*`size`*`>]`**

Description: Interactive command for examining and changing memory locations.

Arguments:

`location` Starting memory address to change, as viewed by the i960 processor.

`size` Access size for addressing the memory. This argument must be set to **`1`**, **`2`**, or **`4`** bytes. If this argument is omitted, a default of **`4`** bytes is used.

While examining memory, the following characters have a special meaning when entered from the terminal:

`<CR>` Increments to the next location in memory without changing the contents in memory.

'**`.`**' Terminates **`modify`** command and returns to monitor.

'**`=`**' Reads the same location in memory without incrementing to the next address.

'**`-`**' Decrements the address to view the previous memory location.

When modifying a location in memory, the '.', '=', and '–' characters can be specified after the data being entered.

Example: This example demonstrates the various capabilities of the **`modify`** command.

```
Pentek 4292> modify -b 0xa0800000 -s 4
a0800000 00000000:
a0800004 00000000:-
a0800000 00000000:=
a0800000 00000000:=
a0800000 00000000:1234567=
a0800000 01234567:
a0800004 00000000:abcde-
a0800000 01234567:
a0800004 000abcde:
a0800008 00000000:a5a5a5a5.
Pentek 4292>
```

11.4 Command Processor Commands (continued)

11.4.10 display

(Note: This command is unique to the Command Processor, and is not provided in the RedBoot Monitor command set.)

Syntax: `display [-c] [-p]`

Description: Displays the current voltage and temperature sensor readings that are stored in the i960 Processor section of Global SDRAM (see Section 10.4.1 and Table 10–3).

Arguments:

`-c`	Continuous display of current temperature and voltage readings. The board must be reset to terminate this command.
`-p`	Displays the peak temperature and voltage readings.

Examples:

1) Display the current temperature and voltage readings.

```
Pentek 4292> display
+1.8V Value = 1.770
+2.5V Value = 2.451
+3.3V Value = 3.253
+5V Value = 5.058
+12V Value = 12.203
-12V Value = -12.179
Top Temperature Sensor Value =  32 °C
Center Temperature Sensor Value =  28 °C
Bottom Temperature Sensor Value =  29 °C
Pentek 4292>
```

2) Display the Max and Min temperature and voltage readings.

```
Pentek 4292> display -p
+1.8V Min = 1.770  Max = 1.780
+2.5V Min = 2.451  Max = 2.451
+3.3V Min = 3.270  Max = 3.270
+5V Min = 5.032  Max = 5.058
+12V Min = 12.203  Max = 12.265
-12V Min = -12.179  Max = -12.179
Top Temperature Sensor Peak =  40 °C
Center Temperature Sensor Peak =  34 °C
Bottom Temperature Sensor Peak =  31 °C
Pentek 4292>
```

11.5 RedBoot Monitor Commands

The RedBoot Monitor provides the following operator command set. All references to Global SDRAM and i960 FLASH memories in these commands are based on the i960 Processor Memory Map in Section 5.3.

The RedBoot Monitor includes most of the Command Processor command set. The following RedBoot Monitor commands are also available in the Command Processor command set, Section 11.4:

dump, **reset**, **version**, **fill**, **help**, **edit**, **pedit**, **pdump**

NOTE: Any board configuration parameters set by RedBoot Monitor commands (**fconfig** – Section 11.5.20, **vmeslaveconfig** – Section 11.5.24, **vmemasterconfig** – Section 11.5.25, and **appnetconfig** – Section 11.5.26) are saved in i960 FLASH and restored after a board reset.

11.5.1 dump

Syntax: **dump [-b <*location*>] [-l <*length*>]**

Description: Displays the current contents of a region in memory, in Hex format.

Arguments:

location Starting memory location to display, as viewed by the i960 processor. If this argument is omitted, the address following the last address displayed from a previous **dump** command is the starting address.

length Number of 32–bit words to display. If this argument is omitted, 32 words are displayed.

Example: Display memory range **0xA0300000** to **0xA0300010**.

```
RedBoot> dump -b 0xA0300000 -l 4

0xA0300000: 00000000 00000000 00000000 00000000
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.2 reset

Syntax: **reset**

Description: Resets the i960 processor.

Arguments: None

Example:

```
RedBoot> reset
... Resetting.
+
RedBoot(tm) bootstrap and debug environment - built 11:57:07, Jun 27 2001
Platform: PENTEK 4292 (I960 RN)
Copyright (C) 2000, 2001, Red Hat, Inc.

RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
FLASH: 0xfec00000 - 0xff000000, 32 blocks of 0x00020000 bytes each.
No PMC Module found
IP: 192.168.200.193, Default server: 192.168.200.68
== Executing boot script in 1.000 seconds - enter ^C to abort
RedBoot> fis load cmdproc
RedBoot> go

Pentek 4292 Firmware - built 12:44:00, Jun 22 2001
Platform: PENTEK 4292 (I960 RN)
RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
Pentek 4292>
```



After a reset, RedBoot automatically loads and executes the Command Processor application (the '**fis load cmdproc**' and '**go**' boot script commands, above). You can return to the RedBoot command prompt by entering <Ctrl C> before the reset sequence executes this boot script (see Section 11.3).

11.5 RedBoot Monitor Commands (continued)

11.5.3 version

Syntax: `version`

Description: Displays the current revision of the RedBoot Monitor.

Arguments: None

Example: Display RedBoot version information.

```
RedBoot> version

RedBoot(tm) bootstrap and debug environment - built 11:57:07, Jun 27 2001
Platform: PENTEK 4292 (I960 RN)
Copyright (C) 2000, 2001, Red Hat, Inc.

RAM: 0xa0001000-0xa1ffffff, 0xa0400000-0xa1ffffff available
RedBoot>
```

11.5.4 fill

Syntax: `fill -b <location> -l <length> -v <value>`

Description: Fills the contents of memory.

Arguments:

location	Starting memory address to fill, as viewed by the i960 processor.
length	Number of 32–bit words to write.
value	32–bit data value to write into the specified words of memory.

Example: Initialize memory region **0xa0800000** – **0xa080007F**.

```
RedBoot> fill -b 0xa0800000 -l 0x20 -v 0xa5a5a5a5
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.5 help

Syntax: `help`

Description: Displays a list of RedBoot commands and their arguments.

Arguments: None

Example:

```
RedBoot> help

Manage application's network parameters kept in FLASH memory
   appnetconfig [-l]
Compute a 32bit checksum [POSIX algorithm] for a range of memory
   cksum -b <location> -l <length>
Display (hex dump) a range of memory
   dump -b <location> [-l <length>]
Edit the contents of memory
   edit -b <location> -v <value> [-s <size>]
Fill the contents of memory
   fill -b <location> -l <length> -v <value>
Manage FLASH images
   fis {cmds}
Manage RedBoot configuration kept in FLASH memory
   fconfig [-i] [-l]
Execute code at a location
   go [-w <timeout>] [entry]
Help about help?
   help [<topic>]
Load a file
   load [-r] [-v] [-h <host>] [-m {TFTP | xyzMODEM}]
        [-b <base_address>] <file_name>
Display (hex dump) PCI Configuration Space
   pdump -d <device> [-f <function>] [-l <length>]
Edit a register in PCI Configuration Space
   pedit -d <device> -f <function> -o <offset> -v <value>
Network connectivity test
   ping [-v] [-n <count>] [-l <length>] [-t <timeout>] [-r <rate>]
        [-i <IP_addr>] -h <IP_addr>
Reset the system
   reset
Display RedBoot version information
   version
Manage VME Master Address parameters kept in FLASH memory
   vmemasterconfig [-l]
Manage VME Slave Address parameters kept in FLASH memory
   vmeslaveconfig [-l]
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.6 edit

Syntax: `edit -b <location> -v <value> [-s <size>]`

Description: Edits the contents of memory.

Arguments:

location Memory address to modify, as viewed by the i960 processor.

value Data value to write into the memory address.

size Number of bytes to write. This argument must be set to **1**, **2**, or **4** bytes. If this argument is omitted, a default of **4** bytes is used.

Example: Edit four bytes in Global SDRAM.

```
RedBoot> edit -b 0xa0800000 -v 0x12345678 -s 4
```

11.5.7 pedit (This command is available only in RedBoot version 1.6 or greater)

Syntax: `pedit -d <device> -f <function> -o <offset> -v <value>`

Description: Edits the contents of a register in PCI Configuration Space.

Arguments:

device Reference number of the PCI device to access:

1 – Tundra Universe II VME Interface
2 – PMC Site
3 – Intel 82559 Ethernet Controller
4 – Intel i960 PCI Bridge
5 – Node A PLX PCI 9054
6 – Node B PLX PCI 9054
7 – Node C PLX PCI 9054
8 – Node D PLX PCI 9054

function Reference number of the function to access in a multi–function device. Range is **0** to **7**.

offset Byte offset of the register to edit, from the start of the device's PCI configuration space.

value Data value to write into the register.

Example: Modify the PMC's command/status register at offset 4.

```
RedBoot> pedit -d 2 -f 0 -o 4 -v 0x7
```

11.5 RedBoot Monitor Commands (continued)

11.5.8 pdump (This command is available only in RedBoot version 1.6 or greater)

Syntax: **pdump -d <*device*> [-f <*function*>] [-l <*length*>]**

Description: Displays the contents of a device in PCI Configuration Space, in Hex format.

Arguments:

device Reference number of the PCI device to access. The PCI devices available are:

1 – Tundra Universe II VME Interface
2 – PMC Site
3 – Intel 82559 Ethernet Controller
4 – Intel i960 PCI Bridge
5 – Node A PLX PCI 9054
6 – Node B PLX PCI 9054
7 – Node C PLX PCI 9054
8 – Node D PLX PCI 9054

function Reference number of the function to access in a multi−function device. Range is **0** to **7**. When this argument is omitted, the last specified function is used. The default is **0**.

length Number of 32−bit words to display. When this argument is omitted, the last specified length is used. The default is 16 words.

Example: Dump the PMC's configuration space registers for the RACEway PMC (Option 005).

```
RedBoot> pdump -d 2

00000000: 00010034 04800007 06040002 00010000
00000010: f0000008 18600000 00000100 04800111
00000020: 20001870 fff0f000 00000000 00000000
00000030: 00000000 00000000 18700000 00000000
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

The following commands, Sections 11.5.9 through 11.5.26, are unique to the RedBoot Monitor, and are not provided in the Command Processor command set.

11.5.9 cksum

Syntax: `cksum -b <location> -l <length>`

Description: Calculates a POSIX checksum of a region of memory.

Arguments:

location	Starting address of a region in memory to checksum.
length	Number of bytes of memory to include in the checksum.

Example: Checksum a program stored in FLASH memory.

```
RedBoot> cksum -b 0xFEC00000 -l 0x40000

POSIX cksum = 0xDF546E72 (3746852466)
RedBoot>
```

11.5.10 fis init

Syntax: `fis init [-f]`

Description: Initializes the FLASH Image System (FIS). This command is called once, when RedBoot is first installed on the hardware, and should only be used by the factory.

Arguments:

`-f`	If this argument is specified, all blocks of FLASH memory are erased.

Example: Initialize the FLASH Image System.

```
RedBoot> fis init -f

About to initialize [format] FLASH image system - are you sure (y/n)? y
*** Initialize FLASH Image System
... Erase from 0xfec00000-0xfeec0000: ......................
... Unlock from 0xfeee0000-0xfef00000: .
... Erase from 0xfeee0000-0xfef00000: .
... Program from 0xa02e0000-0xa02e0500 at 0xfeee0000: .
... Lock from 0xfeee0000-0xfef00000: .
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.11 fis list

Syntax: **`fis list [-c] [-d]`**

Description: Lists the directory of all images saved in FLASH memory.

Arguments:

`-c` Displays the image checksum instead of the FLASH memory address (**`Mem addr`**).

`-d` Displays the image datalength instead of the amount of FLASH memory created (**`Length`**).

Example: Display directory of the FLASH Image System.

```
RedBoot> fis list

Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0xFEFC0000  0xFEFC0000  0x00040000  0x00000000
RedBoot[backup]   0xFEF80000  0xFEF80000  0x00040000  0x00000000
CmdProc           0xFEF00000  0xA00A0000  0x00080000  0xA00A0420
RedBoot config    0xFEEC0000  0xFEEC0000  0x00020000  0x00000000
FIS directory     0xFEEE0000  0xFEEE0000  0x00020000  0x00000000
RedBoot>
```

11.5.12 fis free

Syntax: **`fis free`**

Description: Displays the areas of FLASH memory not being used by the FLASH Image System.

Arguments: None

Example: Display all available unused FLASH memory.

```
RedBoot> fis free

0xFEC40000 .. 0xFED00000
0xFED20000 .. 0xFEEC0000
0xFEFA0000 .. 0xFEFC0000
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.13 fis create

Syntax: `fis create -b <location> -l <length> [-f <address>] [-e <strtaddr>] [-r <loadaddr>] [-n] <name>`

Description: Creates an image in the FIS Directory.

Arguments:

location	Address in Global SDRAM where the program image resides.
length	Length of the image, in bytes.
address	Address in FLASH to write the image. If this argument is omitted, the image is loaded to the first free block of FLASH that can fit the image.
strtaddr	Program's start address. If this argument is omitted, the starting address from a previous load command is used.
loadaddr	Address in SDRAM to place the image when an **`fis load`** command (Section 11.5.15) is issued. If this argument is omitted, the image is placed at the location specified by the **`-b`** argument (***location***).
`-n`	Update the FIS directory only. The image is not loaded into FLASH memory. This option fixes a corrupt FIS directory entry.
name	Name assigned to the image created.

Example: Load a program named **`check`** into FLASH—the **`check`** program was stored in Global SDRAM at **`0xA0500000`**.

```
RedBoot> fis create -b 0xa00a0000 -l 0x40000 check

... Erase from 0xfec00000-0xfec40000: ..
... Program from 0xa00a0000-0xa00e0000 at 0xfec00000: ..
... Unlock from 0xfeee0000-0xfef00000: .
... Erase from 0xfeee0000-0xfef00000: .
... Program from 0xa02e0000-0xa0300000 at 0xfeee0000: .
... Lock from 0xfeee0000-0xfef00000: .
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.14 fis delete

Syntax: `fis delete <name>`

Description: Deletes an existing program image from FLASH memory.

Arguments:

name	Name of the program image to delete.

Example: Remove the program named **check** from the FIS Directory.

```
RedBoot> fis delete check

Delete image 'check' - are you sure (y/n)? y
... Erase from 0xfec00000-0xfec40000: ..
... Unlock from 0xfeee0000-0xfef00000: .
... Erase from 0xfeee0000-0xfef00000: .
... Program from 0xa02e0000-0xa0300000 at 0xfeee0000: .
... Lock from 0xfeee0000-0xfef00000: .
RedBoot>
```

11.5.15 fis load

Syntax: `fis load [-b <location>] [-c] <name>`

Description: Copies an existing named image from FLASH memory to Global SDRAM.

Arguments:

location	Destination address in Global SDRAM. If this argument is omitted, the program is copied to the address (*loadaddr*) specified in the **fis create** command (Section 11.5.13).
-c	Calculate a checksum of the image after it has been copied and display it.
name	Name of the image to copy.

Example: Copy the **check** program to Global SDRAM and compute its checksum.

```
RedBoot> fis load -c check

Checksum: 0x1bdcec6f
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.16 fis lock

Syntax: `fis lock -f <address> -l <length>`

Description: Write−protects a section of FLASH memory. This prevents a program from being accidentally overwritten or erased.

Arguments: ***address*** Starting address in FLASH memory to lock.

length Length of FLASH memory to lock. The length should be a multiple of the block size (**0x20000**).

Example: Lock FLASH memory region **0xfec00000** to **0xfec80000**.

```
RedBoot> fis lock -f 0xfec00000 -l 0x80000

... Lock from 0xfec00000-0xfec80000: ....
RedBoot>
```

11.5.17 fis unlock

Syntax: `fis unlock -f <address> -l <length>`

Description: Write−enables a section of FLASH memory. This unlocks an area of memory that was locked with the **fis lock** command (Section 11.5.16, above).

Arguments: ***address*** Starting address in FLASH memory to unlock.

length Length of FLASH memory to unlock. The length should be a multiple of the block size (**0x20000**).

Example: Unlock FLASH memory region **0xfec00000** to **0xfec80000**.

```
RedBoot> fis unlock -f 0xfec00000 -l 0x80000

... Unlock from 0xfec00000-0xfec80000: ....
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.18 fis erase

Syntax: **`fis erase -f <`*`address`*`> -l <`*`length`*`>`**

Description: Erases a section of FLASH memory.

Arguments: ***`address`*** Starting address in FLASH memory to erase.

`length` Length of FLASH memory to erase, in bytes.

Example: Erase FLASH memory region **`0xfec00000`** to **`0xfec80000`**.

```
RedBoot> fis erase -f 0xfec00000 -l 0x80000

... Erase from 0xfec00000-0xfec80000: ....
RedBoot>
```

11.5.19 fis write

Syntax: **`fis write -b <`*`location`*`> -f <`*`address`*`> -l <`*`length`*`>`**

Description: Writes data from Global SDRAM to FLASH memory.

Arguments: ***`location`*** Address in Global SDRAM containing the source data to write.

`address` Starting address in FLASH memory to write.

`length` Length of data to write, in bytes.

Example: Copy an array of data, located at Global SDRAM address **`0xa0600000`** to **`0xa0610000`**, to FLASH memory address **`0xfed00000`**.

```
RedBoot> fis write -b 0xa0600000 -f 0xfed00000 -l 0x10000

* CAUTION * about to program FLASH at 0xfed00000..0xfed1ffff
    from 0xa0600000 - are you sure (y/n)? y

... Erase from 0xfed00000-0xfed20000: .
... Program from 0xa0600000-0xa0620000 at 0xfed00000: .
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.20 fconfig

Syntax: `fconfig [-l] [-i]`

Description: Manages the RedBoot configuration parameters stored in FLASH memory. The configurable parameters are:

`Run script at boot` – RedBoot has the capability of executing the RedBoot command set from a boot script. This parameter must be set to **`true`** (**`t`**) or **`false`** (**`f`**). The default is **`false`**. When this parameter is set to **`true`**, the user is prompted to enter a boot script. Specify each command followed by a <CR>. Complete the script by entering a <CR> on a blank line.

For example, at board reset the following boot script will load a program from FLASH memory and start executing it:

```
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis load cmdproc
>> go
>>
```

`Boot script timeout` – This parameter delays the execution of the boot script giving the user time to stop its execution. Specify timeout in seconds, to 1000ms resolution, with the format **`X.XXX`**. <Ctrl C> stops the boot script running.

`Enable Network` – This parameter enables the Ethernet network interface (see Section 2.6.6 for the Ethernet connector). Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable. The default is **`false`**.

Network settings used in this command are used only by RedBoot. The Command Processor and SwiftNet use the network settings provided in the **`appnetconfig`** command (Section 11.5.26).

`Use BOOTP for network configuration` – When the network is enabled, the board can obtain it's Ethernet IP address from a BOOTP server, or the IP address can be entered manually. Set to **`true`** (**`t`**) to use BOOTP or **`false`** (**`f`**) to enter manually with the following two IP address parameters. The default is **`false`**. (BOOTP configuration has not yet been tested.)

11.5 RedBoot Monitor Commands (continued)

11.5.20 fconfig (continued)

Description:
(continued)

`Local IP address` – When BOOTP is not being used, this parameter specifies the IP address for this board. IP address is specified with the following format: **`XXX.XXX.XXX.XXX`**.

`Default server IP address` – When BOOTP is not being used, this parameter specifies a default server IP Address. This is an optional parameter that is used by the **`load`** and **`ping`** commands as a default server. IP address is specified with the following format: **`XXX.XXX.XXX.XXX`**.

`Console Baud Rate` – This parameter selects the RS–232 baud rate for the console. This must be set to one of the following: **`9600`**, **`19200`**, **`38400`**, **`57600`**, or **`115200`**. The default is **`9600`**.

`GDB connection port` – When the network is enabled, this parameter selects the TCP port to be used by the GDB debugger for debugging over the network interface. This parameter is also used to access the RedBoot Monitor over the network using telnet (i.e., **`telnet XXXX 9000`**, where **`XXXX`** is the hostname for the Model 4292/4293). The default is **`9000`**.

Arguments:

`-l` List current configuration settings. If this argument is omitted, you can modify each parameter as it is listed. To modify a value, type in the new value. To keep the displayed value, just use the **Enter** key.

`-i` Restore default configuration parameters. This includes restoring all the parameters in:
`appnetconfig` (Section 11.5.26),
`vmeslaveconfig` (Section 11.5.24), and
`vmemasterconfig` (Section 11.5.25).

Example: Display the current configuration parameters.

```
RedBoot> fconfig -l

Run script at boot: false
Enable Network: false
Console baud rate: 9600
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.21 load

Syntax: **`load [-r] [-b <address>] [-v] [-h <host>]`**
`[-m {tftp|xmodem|ymodem}] <name>`

Description: Loads a file from the serial port using XModem or YModem protocol, or from the network using TFTP. Files being loaded must be raw data or S–Record format.

Arguments:

`-r` Specifies that the file contains raw data. If this argument is specified, the **`-b address`** argument (below) must also be provided.

`address` Address in Global SDRAM to place the file. This is required when using the **`-r`** argument or when a program cannot be loaded to the same address that it executes from (i.e., FLASH).

`-v` Displays a spinner indicator on the RS–232 console while the file is being loaded.

`host` Host's IP address when loading a program via TFTP. If the host is the same as the default server IP address specified in **`fconfig`**, then this does not need to be specified. The format is **`XXX.XXX.XXX.XXX`**.

`-m` Communication protocol used to load the file. Set to **`xmodem`** or **`ymodem`** for serial port downloads, or to **`tftp`** for network downloads. If this argument is omitted, the default is **`tftp`**.

`name` The name of the file to load.

Examples: Load the **`check`** program using TFTP—the program should be loaded to the same place that it will execute from.

```
RedBoot> load -v check.srec
Entry point: 0xa00a0420, address range: 0xa00a0000-0xa00ddd68
RedBoot>
```

Load the **`check`** program using the serial port—the program should be loaded to the same place that it will execute from.

```
RedBoot> load -m ymodem check.srec
xyzModem - CRC mode, 2(SOH)/380(STX)/0(CAN) packets, 5 retries
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.22 go

Syntax: **`go [-w <`*`timeout`*`>] [<`*`entry`*`>]`**

Description: Starts the execution of a program that was loaded by either the **`load`** command (Section 11.5.21) or the **`fis load`** command (Section 11.5.15).

Arguments:

`timeout` Number of seconds to delay before starting the program's execution. If this argument is omitted, the default timeout is **`0`** seconds (no delay).

`entry` Starting address of the program. If this argument is omitted, the entry point from the previous **`load`** command is used.

Example: Load and execute the **`main`** program—the program is executed from the same address that it was loaded to.

```
RedBoot> load main.hex
Entry point: 0xa00a0420, address range: 0xa00a0000-0xa00ddd68
RedBoot> go
```

11.5 RedBoot Monitor Commands (continued)

11.5.23 ping

Syntax: **`ping [-v] [-n <`*`count`*`>] [-t <`*`timeout`*`>] [-r <`*`rate`*`>] [-i <`*`local`*`>] -h <`*`host`*`>`**

Description: Verifies the connection between the Model 4292/4293 and a host on the local network. For this command, the network must be enabled in **`fconfig`** (Section 11.5.20).

Arguments:

`-v`	Selects verbose output. Information about each packet sent is displayed.
`count`	Number of packets to send. If this argument is omitted, the default count is **`10`**.
`timeout`	Timeout on receiving packets, in seconds. If this argument is omitted, the default timeout is **`1`** second.
`rate`	Interval between sending packets, in seconds. If this argument is omitted, the default interval is **`1`** second. Specifying **`0`** results in packets being sent without delay.
`local`	Local IP Address. This argument is necessary only if you wish to override the setting specified in **`fconfig`**. The IP address format is **`XXX.XXX.XXX.XXX`**.
`host`	Host's IP Address. The host IP address format is **`XXX.XXX.XXX.XXX`**.

Example: Check connection to another host on the network.

```
RedBoot> ping -h 192.168.200.1
Network PING - from 192.168.200.193 to 192.168.200.1
PING - received 10 of 10 expected
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.24 vmeslaveconfig

Syntax: `vmeslaveconfig [-l]`

Description: Manages the VME Slave parameters stored in FLASH memory. The configurable parameters are:

`Enable VME Slave Extended Memory Map` – This parameter enables the VME Slave Extended Memory Map for access to various resources on the board from an external VMEbus Master. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable VME Slave access. The default is **`false`**. The Model 4292/4293 as shipped from the factory only has access to the Tundra Universe II's registers in A16 or A24 Address Modes.

When **`VME Slave Extended Memory Map`** is enabled (above), the following parameters can be configured (the default values provide the Slave memory map shown in Section 7.3.2).

`Enable A24 VME Slave Address Space` – This parameter selects either the A24 or A32 VME Slave Extended Memory Map. Set to **`true`** (**`t`**) to enable A24 access or **`false`** (**`f`**) to enable A32 access. The default is **`false`**.

`VME Slave Base Address` – This parameter selects the VME Slave Extended Memory Map's Base Address. When A24 access is enabled, this parameter should be set to either **`0x0`** or **`0x800000`**. When A32 access is enabled (default), this parameter must be in the range **`0x0`** to **`0xFE000000`**, in multiples of **`0x02000000`**. The default is **`0x0`**.

`Enable VME Slave Global Bifo Space` – This parameter enables VME Slave access to the Global FIFO's. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable VME access. The default is **`true`**.

`Enable VME Slave Message Buffer Space` – This parameter enables VME Slave access to the i960 Messaging Unit buffers. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable access. The default is **`true`**.

`Enable VME Slave DSP Node Space` – This parameter enables VME Slave access to the DSP Nodes (PLX PCI 9054 registers and 'C6x Host Port registers). Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable VME access. The default is **`true`**.

11.5 RedBoot Monitor Commands (continued)

11.5.24 vmeslaveconfig (continued)

Description:
(continued)

`Enable VME Slave PCI I/O Space` – This parameter enables VME Slave access to the PCI Bus I/O Space. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable VME access. The default is **`true`**.

`Enable VME Slave PMC Space` – This parameter enables VME Slave access to the PMC's memory space. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable VME access. The default is **`true`**.

`PMC Space Base Address Offset` – When VME Slave access to PMC memory space is enabled, this parameter specifies the offset from the VME Slave Base Address where PMC access begins. The default setting is **`0x200000`**.

`PMC Access Space Size` – When VME Slave access to PMC memory space is enabled, this parameter specifies the size, in bytes, of the VME access window. The default size is **`0x200000`**.

`Enable VME Slave Global SDRAM Space` – This parameter enables VME Access to the Global SDRAM. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable VME access. The default is **`true`**.

`Global SDRAM Space Base Address Offset` – When VME Slave access to Global SDRAM is enabled, this parameter specifies the offset from the VME Slave Base Address where Global SDRAM access begins. The default setting is **`0x400000`**.

`Global SDRAM Access Space Size` – When VME Slave access to Global SDRAM is enabled, this parameter specifies the size, in bytes, of the VME access window. The default size is **`0x1C00000`** in A32 Mode or **`0x400000`** in A24 Mode.

`Enable VME Slave Image 7 User Configurable` – This parameter enables Universe II's VME Slave Image 7. This slave image can be set up as needed depending on the user's application. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable this image. The default is **`false`**. Refer to the Tundra Universe II manual.

`VME Slave Image 7 Base Address` – When VME Slave Image 7 is enabled, this parameter specifies the complete VME address for this image.

11.5 RedBoot Monitor Commands (continued)

11.5.24 vmeslaveconfig (continued)

Description:
(continued) **`VME Slave Image 7 Destination Address`** – When VME Slave Image 7 is enabled, this parameter specifies the starting PCI Destination Address.

`VME Slave Image 7 Size` – When VME Slave Image 7 is enabled, this parameter specifies the size of this image window.

`VME Slave Image 7 Control Register Setting` – When VME Slave Image 7 is enabled, this parameter specifies the VME Slave Image 7 control register setting. Refer to the Tundra Universe II manual.

Arguments: **`-l`** List current configuration settings. If this argument is omitted, you can modify each parameter as it is listed. To modify a value, type in the new value. To keep the displayed value, just use the **Enter** key.

Example: Display the current VME Slave configuration settings.

```
RedBoot> vmeslaveconfig -l
Enable VME Slave Extended Memory Map: true
Enable A24 VME Slave Address Space(true=A24,false=A32): false
VME Slave Base Address: 0x0
Enable VME Slave Global Bifo Space: true
Enable VME Slave Message Buffer Space: true
Enable VME Slave DSP Node Space: true
Enable VME Slave PCI IO Space: true
Enable VME Slave PMC Space: true
PMC Space Base Address Offset: 0x200000
PMC Access Space Size: 0x200000
Enable VME Slave Global SDRAM Space: true
Global SDRAM Space Base Address Offset: 0x400000
Global SDRAM Access Space Size: 0x1c00000
Enable VME Slave Image 7 - User Configurable: false
RedBoot>
```

11.5 RedBoot Monitor Commands (continued)

11.5.25 vmemasterconfig

Syntax: `vmemasterconfig [-l]`

Description: Manages the VME Master parameters stored in FLASH memory. The configurable parameters are as follows (the default values provide the Master memory map shown in Section 7.2).

`Enable VME Master A32/D32 Space` – This parameter enables VME Master A32/D32 Address Space. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable. The default is **`true`**.

`VME Master A32/D32 PCI Address` – When A32/D32 Space is enabled, this parameter contains the PCI address that provides access to this region. The default PCI address is **`0x90000000`**.

`VME Master A32/D32 VME Address` – When A32/D32 Space is enabled, this parameter contains the starting VME address that is generated when the PCI address defined above is accessed. The default VME address is **`0x0`**.

`VME Master A32/D32 Space Size` – When A32/D32 Space is enabled, this parameter contains the size, in bytes, of the A32/D32 window. The default size is **`0x60000000`**.

`Enable VME Master A32/D64 Space` – This parameter enables VME Master A32/D64 Address Space. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable. The default is **`true`**.

`VME Master A32/D64 PCI Address` – When A32/D64 Space is enabled, this parameter contains the PCI address that provides access to this region. The default PCI address is **`0x84000000`**.

`VME Master A32/D64 VME Address` – When A32/D64 Space is enabled, this parameter contains the starting VME address that is generated when the PCI address defined above is accessed. The default VME address is **`0x04000000`**.

`VME Master A32/D64 Space Size` – When A32/D64 Space is enabled, this parameter contains the size, in bytes, of the A32/D64 window. The default size is **`0x0C000000`**.

11.5 RedBoot Monitor Commands (continued)

11.5.25 vmemasterconfig (continued)

Description:
(continued)

`Enable VME Master Image 6 User Configurable` – This parameter enables Universe II's PCI Target Image 6. This master image can be set up as needed depending on the user's application. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable this image. The default is **`false`**. Refer to the Tundra Universe II manual.

`VME Master Image 6 PCI Address` – When VME Master Image 6 is enabled, this parameter specifies the PCI address that provides access for this image.

`VME Master Image 6 VME Address` – When VME Master Image is enabled, this parameter contains the starting VME address that is generated when the PCI address defined above is accessed.

`VME Master Image 6 Size` – When VME Master Image 6 is enabled, this parameter contains the size, in bytes, of this window.

`VME Master Image 6 Control Register Setting` – When VME Master Image 6 is enabled, this parameter specifies the PCI Target Image 6 control register setting. Refer to the Tundra Universe II manual.

`VME Master Image 6 User AM` – When VME Master Image 6 is enabled, this optional parameter specifies a custom AM code. This parameter is needed if the ones selectable in the PCI Target Image control register aren't applicable. A setting of **`0x0`** disables this parameter.

`Enable VME Master Image 7 User Configurable` – This parameter enables Universe II's PCI Target Image 7. This master image can be set up as needed depending on the user's application. Set to **`true`** (**`t`**) to enable or **`false`** (**`f`**) to disable this image. The default is **`false`**. Refer to the Tundra Universe II manual.

`VME Master Image 7 PCI Address` – When VME Master Image 7 is enabled, this parameter specifies the PCI address that provides access for this image.

11.5 RedBoot Monitor Commands (continued)

11.5.25 vmemasterconfig (continued)

Description:
(continued)

`VME Master Image 7 VME Address` – When VME Master Image 7 is enabled, this parameter contains the starting VME address that is generated when the PCI address defined above is accessed.

`VME Master Image 7 Size` – When VME Master Image 7 is enabled, this parameter contains the size, in bytes, of this window.

`VME Master Image 7 Control Register Setting` – When VME Master Image 7 is enabled, this parameter specifies the PCI Target Image 7 control register setting. Refer to the Tundra Universe II manual.

`VME Master Image 7 User AM` – When VME Master Image 7 is enabled, this optional parameter specifies a custom AM code. This parameter is needed if the ones selectable in the PCI Target Image control register are not applicable. A setting of **`0x0`** disables this parameter.

`VMEBus Release Mode` – This parameter selects the VMEbus Release Mode. Set to **`0`** for release when done or **`1`** for release on request. The default is **`0`**.

`VMEBus Request level` – This parameter selects the VMEbus Request Level. Set to **`0`**, **`1`**, **`2`**, or **`3`**. The default is **`3`**.

`VMEBus Request Mode` – This parameter selects the VMEbus Request Mode. Set to **`0`** for request on demand or **`1`** for fair requester. The default is **`0`**.

Arguments:

`-l`	List current configuration settings. If this argument is omitted, you can modify each parameter as it is listed. To modify a value, type in the new value. To keep the displayed value, just use the **Enter** key.

11.5 RedBoot Monitor Commands (continued)

11.5.25 vmemasterconfig (continued)

Example: Display the current VME Master configuration settings.

```
RedBoot> vmemasterconfig -l
Enable VME Master A32/D32 Space: true
VME Master A32/D32 PCI Address: 0x90000000
VME Master A32/D32 VME Address: 0x0
VME Master A32/D32 Space Size: 0x60000000
Enable VME Master A32/D64 Space: true
VME Master A32/D64 PCI Address: 0x84000000
VME Master A32/D64 VME Address: 0x4000000
VME Master A32/D64 Space Size: 0xc000000
Enable VME Master Image 6 - User Configurable: false
Enable VME Master Image 7 - User Configurable: false
VMEBus Release Mode(0=RWD,1=ROR): 0
VMEBus Request Level(0-3): 3
VMEBus Request Mode(0=demand,1=fair): 0
RedBoot>
```

11.5.26 appnetconfig

Syntax: **`appnetconfig [-l]`**

Description: Manages application's network parameters stored in FLASH memory (used by the Command Processor and SwiftNet applications). The configurable parameters are:

`Enable Network` – This parameter enables the Ethernet network interface (see Section 2.6.6 for the Ethernet connector). Set to **`true(t)`** to enable or **`false(f)`** to disable. The default is **`false`**.

`Use BOOTP for network configuration` – When the network is enabled, the Model 4292/4293 can obtain it's Ethernet IP Address from a BOOTP server, or the IP Address can be entered manually. Set to **`true(t)`** to use BOOTP, or **`false(f)`** to enter manually with the following parameters. The default is **`false`**. (BOOTP configuration has not yet been tested.)

`Host IP Address` – This parameter specifies the IP Address for the host running a SwiftNet application. The IP address is specified with the following format: **`XXX.XXX.XXX.XXX`**. Refer to SwiftNet documentation for usage information.

`Host Name` – This parameter specifies the name of the host associated with the host IP address.

11.5 RedBoot Monitor Commands (continued)

11.5.26 appnetconfig (continued)

Description:
(continued)

`Target IP Address` – This parameter specifies the IP Address for this board. The IP address is specified with the following format: **`XXX.XXX.XXX.XXX`**.

`Target Name` – This parameter specifies the name associated with the target IP address.

`Broadcast Address` – This parameter specifies the IP Address for broadcasting packets to all systems on a network. This parameter is not used by the Command Processor or SwiftNet. The broadcast address is specified with the following format: **`XXX.XXX.XXX.XXX`**.

`Gateway Address` – This parameter specifies the IP address of the machine used as a gateway to another network. The gateway address is specified in the following format: **`XXX.XXX.XXX.XXX`**. To access machines that are only on a local network, specify a gateway address of **`0.0.0.0`**. The default is **`0.0.0.0`**.

`Network Mask` – This parameter specifies the network mask. The network mask is specified with the following format: **`XXX.XXX.XXX.XXX`**. The default network mask is **`255.255.255.0`**.

`Boot Flags` – This parameter is reserved for future use by SwiftNet. The default value is **`0x0`**.

Arguments:

`-l`	List current configuration settings. If this argument is omitted, you can modify each parameter as it is listed. To modify a value, type in the new value. To keep the displayed value, just use the **Enter** key.

Example: Display the current configuration parameters.

```
RedBoot> appnetconfig -l
Enable Network: true
Use BOOTP for network configuration: false
Host IP Address: 192.168.200.68
Host Name: ese-68
Target IP address: 192.168.200.190
Target Name: ese-190
Broadcast Address: 192.168.200.255
Gateway Address: 0.0.0.0
Network Mask: 255.255.255.0
Boot Flags: 0x0
RedBoot>
```

This page is intentionally blank

Chapter 12: The Command Processor

12.1 Overview

The i960 I/O processor loads and executes a Command Processor application when the Model 4292/4293 is powered on or following a board reset. The Command Processor application provides the following capabilities:

- Transfers data between any two resources on the board.
- Synchronizes data transfers with external events.
- Provides debugging of 'C6x processors via Ethernet when used with Pentek's SwiftNet and TI's Code Composer Studio.
- Generation and Handling of VME Interrupts.
- Provides TCP/IP networking to the 'C6x processors.

After the board completes power on or board reset boot operations, the Command Processor is available to user applications running in the 'C6x processors or from the VMEbus. Pentek's ReadyFlow Board Support Libraries provide routines for these Command Processor functions (see Section 1.6). Refer to the ReadyFlow Board Support Libraries manual (Pentek part #801.42920) for description of these capabilities.

The Command Processor application also has a limited operator command set available at boot time using the board's RS–232/Ethernet serial port (see Section 11.4).

12.2 Theory of Operation

The Command Processor application uses the i960 processor's Messaging Unit (MU) to receive messages from the 'C6x processors, VME, and PMC. Each message contains a message header followed by one of more commands. A message can be executed for a specified number of times or run continuously. When a message or a command completes, several options are provided for notifying when a message has completed. The Command Processor provides 64 message frames and can execute up to 16 different messages at any one time.

12.2.1 Getting a Message Frame

The Command Processor has 64 message frames allocated in Global SDRAM. Each message frame is 512 Bytes in length. User applications can acquire message frames at system startup, and either use the frames repeatedly for one or more messages or allocate and free the frames dynamically.

12.2 Theory of Operation (continued)

12.2.1 Getting a Message Frame (continued)

The user application can obtain a message frame by accessing the i960 MU's Inbound Queue Port at the address indicated in the following table.

Table 12–1: Inbound Queue Port Addresses

PCI Address	VMEbus A24/A32 Address*
0x1000 0040	VME_slave + **0x03 0040**
* Only available when the Extended VMEbus Slave Memory Map is enabled as described in Section 7.3.2	

When the user application reads this port, the i960 MU returns an address to the next available message frame, as viewed by the PCI Bus. If no message frames are available, then **0xFFFF FFFF** is returned. This should only occur if more than 64 messages are allocated and never freed. The Inbound Queue Port must be accessed with 32–bit data accesses ONLY.

NOTE: The address is provided in little endian format and may need to be byte–swapped when accessed by a big endian processor from the VMEbus. This is not a problem when accessed by a DSP.

12.2.2 Executing a Message

The user application can execute a message by writing the address of the message frame to the i960 MU's Inbound Queue Port (see Table 12–1 above). The Command Processor will receive an interrupt indicating that a message frame address has been written to the Inbound Queue Port. Upon receiving the interrupt, one of 16 command processor tasks are awakened to process the message. The address written to the Inbound Queue Port must be in the same format as when it was originally read. Messages must be written into the message frame in little endian format.

12.2.3 Freeing a Message Frame

The user application can free a message frame using the Command Processor FREE_MSG_FRAME command (see Section 12.4.40). When the command is executed, the message frame is placed on the free message queue. Another method for releasing the message frame is to specify the Return Message Frame flag in the message header (see Section 12.3.1.10).

12.3 Message Format

Each message contains a Message Header followed by one or more Commands. The complete message format is shown below (the maximum message size is 512 Bytes).

Message Frame + 0	**Message Header**	Section 12.3.1
Message Frame + 28 Bytes	**Command**	Section 12.3.2
...		
Message Frame + N Bytes	Next **Command**	Section 12.3.2

12.3.1 Message Header Format

Table 12–2 shows the seven–word message header format. The message originator (user application) must set most of the fields for each message. The Command Processor sets four read–only fields, Command Status, Command Id, Message Status, and Number of Times Message Executed, when processing the command. The description of each field is provided in the subsections following this table.

Table 12–2: Message Header Format

	D31 – D24	D23 – D16	D15 – D9	D8	D7 – D0
Word 1	Command Status*	Command Id*	Message Status*		Message Id
Word 2	Message Size		Message Count		
Word 3	Number of Times Message Executed*		Processor Id		Nr of Commands
Word 4	Reserved (Set field to 0)				
Word 5	Reserved (Set field to 0)			(1)	Notification Type
Word 6	Notification Parameter 1				
Word 7	Notification Parameter 2				

* These fields are read only. (1) Return Message Frame

12.3.1.1 Command Status

This read–only field is set by the Command Processor to indicate the status of the command specified in the Command Id field (the current command being processed or the last command that has been processed). The possible status values are:

0x1 - Command Complete
0x2 - Command in Progress
0x4 - Command Invalid
0x8 - Command Parameter Invalid
0x10 - Command Failed

12.3 Message Format (continued)

12.3.1 Message Header Format (continued)

12.3.1.2 Command Id

This read−only field identifies the current command being pro−cessed or the last command that has been processed by the Com−mand Processor. A list of Command Id's handled by the Command Processor is provided in Section 12.4.

12.3.1.3 Message Status

This read−only field is set by the Command Processor to indicate the overall message status. The possible status values are:

0x1 - Message Complete
0x2 - Message in Progress
0x4 - Message Error

When a Message Error occurs, check the Command Status field (Section 12.3.2.1) to determine the reason for the error.

12.3.1.4 Message Id

The message originator can use this field to identify a message reply. For this use, set the field within the range **0** to **0xFF**.

The i960 firmware uses this field to notify the DSP when the mes−sage has completed. When the message Notification Type is DSP_MEMORY_WRITE or DSP_MESSAGE_WRITE, the i960 will write the value '1', left−shifted by the number in Message Id, to the doorbell register in the PLX PCI 9054, which generates an interrupt to the DSP. Upon receiving the interrupt, the DSP can query the doorbell register to determine which message has completed. For this use, set the field within the range **0** to **31**.

12.3.1.5 Message Size

This field specifies the size of the message in bytes. This field must be set within the range **28** to **512**.

12.3.1.6 Message Count

This field specifies the number of times a message should be exe−cuted. This field must be set within the range **0** to **65535**. To exe−cute a message continuously, set the message count field to **0**.

12.3 Message Format (continued)

12.3.1 Message Header Format (continued)

12.3.1.7 Nr of Commands

This field specifies the number of commands contained in the message. This field must be set within the range **1** to **9**.

12.3.1.8 Number of Times Message Executed

This read−only field is set by the Command Processor to indicate the number of times this message has been executed.

12.3.1.9 Processor Id

This field specifies the Processor Id of the message originator. The Command Processor uses this field to keep track of message frames and data buffers allocated for a given Processor Id. When a FREE_PROC_RESOURCES command (Section 12.4.41) is issued, the Command Processor frees up all message frames and data buffers previously allocated for the specified Processor Id. The field must be set based on the following:

1	DSP A
2	DSP B
3	DSP C
4	DSP D
5 – 15	Reserved
16 – 255	Other Processors

12.3.1.10 Return Message Frame

This single−bit parameter frees a message frame after the message completes. This flag should only be used for Notification Types MESSAGE_WRITE and DSP_MESSAGE_WRITE. Set to **1** to free a message frame.

12.3.1.11 Notification Type and Parameters

These three fields specify the method used to notify the message originator when a message or command completes. The possible Notification Type values are:

0x0 - NO_REPLY
No notification occurs when the message completes. Message originator can poll the message header to determine when the command has completed. No Notification Parameters are specified.

12.3 Message Format (continued)

12.3.1 Message Header Format (continued)

12.3.1.11 Notification Types and Parameters (continued)

0x1 - VME_INT

A VME interrupt is generated when a command and/or message completes. Upon receiving the interrupt, the originator can read the message frame to check status and retrieve any return data. It can also be used to signal to a destination VME Processor that a frame of data has been transferred from the 4292/4293 to its memory.

Two Notification Parameters must be specified with this type:

Notification Parameter 1 – This field contains the VME Interrupt Level, within the range **1** to **7**.

Notification Parameter 2 – This field contains the VME Interrupt Vector, which must be an even number in the range **0** to **254**.

0x2 - MEMORY_WRITE

The first word in the message header is written to a specified memory location within the i960's memory map. This saves the message originator from polling for message completion across multiple data buses.

One Notification Parameter must be specified with this type:

Notification Parameter 1 – This field contains a memory location within the i960's memory map.

0x3 - DSP_MEMORY_WRITE

The first word in the message header is written to a specified memory location within the DSP's memory map. The i960 also writes the value '1', left–shifted by the number in Message Id, to the doorbell register in the PLX PCI 9054, which generates an interrupt to the DSP. Upon receiving the interrupt, the DSP can query the doorbell register to determine which message has completed. This type saves the message originator from polling for message completion across multiple data buses.

12.3 Message Format (continued)

12.3.1 Message Header Format (continued)

12.3.1.11 Notification Types and Parameters (continued)

0x3 - DSP_MEMORY_WRITE (continued)

Two Notification Parameters must be specified with this type:

Notification Parameter 1 – This field identifies the DSP to be notified. This field must be set to:

0 - DSP A
1 - DSP B
2 - DSP C
3 - DSP D

Notification Parameter 2 – This field contains a memory location within the DSP's memory map.

0x4 - MAILBOX_WRITE

The first word in the message header is written to a specified mailbox location within the i960 memory map. This saves the message originator from polling for message completion over multiple data buses. This type is identical to the MEMORY_WRITE type (**0x2**), except that it uses a mailbox.

One Notification Parameter must be specified with this type:

Notification Parameter 1 – This field contains a mailbox location within the i960's memory map.

0x5 - DOORBELL_WRITE

A value is written to a doorbell register within the i960's memory map. This type can be used to signal another processor that has a doorbell register.

Two Notification Parameters must be specified with this type:

Notification Parameter 1 – This field contains a doorbell location within the i960's memory map.

Notification Parameter 2 – This field contains the value to be written to the doorbell register.

12.3 Message Format (continued)

12.3.1 Message Header Format (continued)

12.3.1.11 Notification Types and Parameters (continued)

0x6 - MESSAGE_WRITE

The complete message is written to a specified memory location in the i960's memory map. This type saves the message originator from polling for message completion and retrieving any return data from the command across multiple buses.

One Notification Parameter must be specified with this type:

Notification Parameter 1 – This field contains a memory location within the i960's memory map.

0x7 - DSP_MESSAGE_WRITE

The complete message is written to a specified memory location in the DSP's memory map. The i960 also writes the value '1', left–shifted by the number in Message Id, to the doorbell register in the PLX PCI 9054, which generates an interrupt to the DSP. Upon receiving the interrupt, the DSP can query the doorbell register to determine which message has completed. This type saves the message originator from polling for message completion and retrieving any return data from the command across multiple buses.

Two Notification Parameters must be specified with this type:

Notification Parameter 1 – This field identifies the DSP to be notified. This field must be set to:

0 - DSP A
1 - DSP B
2 - DSP C
3 - DSP D

Notification Parameter 2 – This field contains a memory location within the DSP's memory map.

12.3 Message Format (continued)

12.3.2 Command Format

Each command in a message contains a Command Header, followed by a list of parameters specific to the command. Descriptions of each command and its parameters are provided in Section 12.4.

Table 12–3 shows the five–word command header format. The message originator (user application) must set most of the fields for each command. The Command Processor sets the one read–only field, Command Status, when processing the command. The description of each field is provided in the subsections following this table.

Table 12–3: Command Header Format

	D31 – D24	D23 – D16	D15 – D8	D7 – D0
Word 1	Command Status*	Command Id	Reserved (Set field to 0)	
Word 2	Reserved (Set field to 0)			
Word 3	Notification Type			
Word 4	Notification Parameter 1			
Word 5	Notification Parameter 2			
* This field is read only.				

12.3.2.1 Command Status

This read–only field is set by the Command Processor to indicate the status of the command specified in the Command Id field. The possible Command Status values are:

0x1 - Command Complete
0x2 - Command in Progress
0x4 - Command Invalid
0x8 - Command Parameter Invalid
0x10 - Command Failed

12.3.2.2 Command Id

This field identifies the command. A list of Command Id's han–dled by the Command Processor is provided in Section 12.4. Any parameters for this command are included following the header.

12.3.2.3 Notification Type and Parameters

The Notification Types and Parameters are identical to those in the message header section (see Section 12.3.1.11).

12.4 Commands

The following table lists the commands handled by the Command Processor. Descrip–tions of these commands are provided in the referenced subsections following this table. Each command starts with a five–word Command Header (see Section 12.3.2), followed by an optional set of parameters.

Table 12–4: Command Processor Command List

Command Name	Command Id	Description	Information
READ_I960	0x1	Reads a location in the i960 Memory Map	Section 12.4.1
WRITE_I960	0x2	Writes to a location in the i960 Memory Map	Section 12.4.2
READ_DSP	0x3	Reads a location in the DSP Memory Map	Section 12.4.3
WRITE_DSP	0x4	Writes to a location in the DSP Memory Map	Section 12.4.4
EXECUTE_PROGRAM	0x5	Executes an i960 program in Global SDRAM (This command is currently not implemented)	Section 12.4.5
RESET_DSP_NODE	0x6	Resets one or more DSP processors	Section 12.4.6
RELEASE_DSP_NODE	0x7	Releases reset of one or more DSP processors	Section 12.4.7
EXECUTE_DSP_PROGRAM	0x8	Executes a DSP program	Section 12.4.8
READ_PCI_CONFIG	0x9	Reads a location in PCI Configuration Space	Section 12.4.9
WRITE_PCI_CONFIG	0xA	Writes to a location in PCI Configuration Space	Section 12.4.10
Reserved	0xB – 0xF		–
XFER_SDRAM_TO_VME	0x10	Transfers a block of data from Global SDRAM to VMEbus	Section 12.4.11
XFER_VME_TO_SDRAM	0x11	Transfers a block of data from VMEbus to Global SDRAM	Section 12.4.12
Reserved	0x12 – 0x15		–
XFER_SDRAM_TO_PMC	0x16	Transfers a block of data from Global SDRAM to PMC site	Section 12.4.13
XFER_PMC_TO_SDRAM	0x17	Transfers a block of data from PMC site to Global SDRAM	Section 12.4.14
Reserved	0x18		–
Reserved	0x19		–
XFER_SDRAM_TO_DSP	0x1A	Transfers a block of data from Global SDRAM to DSP Memory	Section 12.4.15
XFER_DSP_TO_SDRAM	0x1B	Transfers a block of data from DSP Memory to Global SDRAM	Section 12.4.16
I960_XFER_SDRAM_TO_DSP	0x1C	Transfers a block of data from Global SDRAM to DSP Memory, using i960 DMA	Section 12.4.17
Reserved	0x1D – 0x1F		–

12.4 Commands (continued)

Table 12–4: Command Processor Command List (continued)

Command Name	Command Id	Description	Information
TCP_SOCKET	**0x20**	Creates an unbound socket in a communications domain	Section 12.4.18
TCP_BIND	**0x21**	Assigns an address to an unnamed socket	Section 12.4.19
TCP_LISTEN	**0x22**	Listens for a connection on a socket	Section 12.4.20
TCP_ACCEPT	**0x23**	Accepts a connection to a socket	Section 12.4.21
TCP_CONNECT	**0x25**	Initiates a connection on a socket	Section 12.4.22
TCP_READ	**0x26**	Reads a block of data from a socket	Section 12.4.23
TCP_WRITE	**0x27**	Writes a block of data to a socket	Section 12.4.24
TCP_CLOSE	**0x28**	Closes a socket connection	Section 12.4.25
TCP_SELECT	**0x29**	Waits for a number of socket file descriptors to change status	Section 12.4.26
TCP_SENDTO	**0x2A**	Transmits a message to another socket	Section 12.4.27
TCP_RECVFROM	**0x2B**	Receives a message from another socket.	Section 12.4.28
TCP_SETSOCKOPT	**0x2C**	Sets the specified socket option	Section 12.4.29
TCP_GETSOCKOPT	**0x2D**	Gets the current setting of the specified option	Section 12.4.30
IOCTL	**0x2E**	Controls a socket's operating characteristics	Section 12.4.31
GENERATE_VME_INT	**0x40**	Generates a VMEbus Interrupt	Section 12.4.32
WAIT_ON_VME_INT	**0x41**	Waits for a VMEbus Interrupt to occur	Section 12.4.33
GENERATE_SYSRESET	**0x42**	Generates a VMEbus SYSRESET	Section 12.4.34
WAIT_ON_INT	**0x48**	Waits for an interrupt to occur	Section 12.4.35
GET_DATA_BUFFER	**0x50**	Allocates a data buffer in Global SDRAM	Section 12.4.36
FREE_DATA_BUFFER	**0x51**	Frees a data buffer in Global SDRAM	Section 12.4.37
GET_SEMAPHORE	**0x54**	Acquires a semaphore for exclusive access to a resource	Section 12.4.38
FREE_SEMAPHORE	**0x55**	Frees a previously acquired semaphore	Section 12.4.39
FREE_MSG_FRAME	**0x58**	Frees a message frame	Section 12.4.40
FREE_PROC_RESOURCES	**0x59**	Frees message frames and data buffers allocated for a specific processor	Section 12.4.41

Each command described in the following subsections starts with a five–word Command Header, followed by an optional set of parameters. The Command Header format is the same for any command, as described in Section 12.3.2.

12.4 Commands (continued)

12.4.1 READ_I960

Command Id: `0x1`

Description: Reads a memory location in the i960's memory map.

Command Format (32–bit words):

Command Header (five 32–bit words)
I960 Memory Location
Return Value

Parameters: **I960 Memory Location** – Address of the memory location to read, as viewed by the i960 processor.

Return Value – Contents of the i960 memory location, returned by the Command Processor.

12.4.2 WRITE_I960

Command Id: `0x2`

Description: Write to a memory location in the i960's memory map.

Command Format (32–bit words):

Command Header (five 32–bit words)
I960 Memory Location
Value
Mask

Parameters: **I960 Memory Location** – Address of the memory location to write, as viewed by the i960 processor.

Value – Data value to write to the i960 memory location.

Mask – Data mask specifying the bits to modify. When the mask is set to `0xFFFFFFFF`, the entire memory location is written with the data value. Otherwise, only the bits in the mask that are set to a **1** are modified in the memory location.

12.4 Commands (continued)

12.4.3 READ_DSP

Command Id: 0x3

Description: Reads a memory location in the DSP's memory map.

Command Format (32–bit words):

Command Header (five 32–bit words)		
B	Reserved (Set field to 0)	DSP
DSP Memory Location		
Return Value		

Parameters: **B (Byte Swap)** – This single–bit parameter specifies whether the data read should be endian byte–swapped during the transfer from DSP memory. Set to **1** for byte swapping.

DSP – This 8–bit parameter specifies the DSP to access. Set to one of the following values:

0 - DSP A
1 - DSP B
2 - DSP C
3 - DSP D

DSP Memory Location – Address of the memory location to read, as viewed by the DSP.

Return Value – Contents of the DSP memory location, returned by the Command Processor.

12.4 Commands (continued)

12.4.4 WRITE_DSP

Command Id: `0x4`

Description: Writes a memory location in the DSP's memory map.

Command Format (32–bit words):

Command Header (five 32–bit words)		
B	Reserved (Set field to 0)	DSP
DSP Memory Location		
Value		
Mask		

Parameters: **B (Byte Swap)** – This single–bit parameter specifies whether the data written should be endian byte–swapped during the transfer to DSP memory. Set to **1** for byte swapping.

DSP – This 8–bit parameter specifies the DSP to access. Set to one of the following values:

0 - DSP A
1 - DSP B
2 - DSP C
3 - DSP D

DSP Memory Location – Address of the memory location to write, as viewed by the DSP.

Value – Data value to write to the DSP memory location.

Mask – Data mask specifying the bits to modify. When the mask is set to **0xFFFFFFFF**, the entire memory location is written with the data value. Otherwise, only the bits in the mask that are set to a **1** are modified in the memory location.

12.4 Commands (continued)

12.4.5 EXECUTE_PROGRAM

Command Id: `0x5`

Description: Executes an i960 program located in Global SDRAM. (**NOTE**: This command is currently not implemented.)

Command Format (32–bit words):

Command Header (five 32–bit words)
i960 Program Start Address

Parameters: **i960 Program Start Address** – Program start address, as viewed by the i960 processor.

12.4.6 RESET_DSP_NODE

Command Id: `0x6`

Description: Resets one or more DSP processors.

Command Format (32–bit words):

Command Header (five 32–bit words)
DSP Mask
Hold in Reset

Parameters: **DSP Mask** – Bit mask indicating the processors to reset. Combine one or more of the following constants:

0x1 - DSP A
0x2 - DSP B
0x4 - DSP C
0x8 - DSP D

Hold In Reset – This parameter specifies whether the reset is released immediately or by the RELEASE_DSP_NODE command (Section 12.4.7). Set to one of the following values:

0 - Release DSP Node Immediately
1 - Hold in Reset

12.4 Commands (continued)

12.4.7 RELEASE_DSP_NODE

Command Id: `0x7`

Description: Releases one or more DSP Processors from reset.

Command Format (32–bit words):

Command Header (five 32–bit words)
DSP Mask

Parameters: **DSP Mask** – Bit mask indicating the processors to release. Combine one or more of the following constants:

`0x1` - DSP A
`0x2` - DSP B
`0x4` - DSP C
`0x8` - DSP D

12.4.8 EXECUTE_DSP_PROGRAM

Command Id: `0x8`

Description: Executes a program located in DSP Memory.

Command Format (32–bit words):

Command Header (five 32–bit words)
DSP
DSP Program Start Address

Parameters: **DSP** – DSP to access. Set to one of the following values:

`0` - DSP A
`1` - DSP B
`2` - DSP C
`3` - DSP D

DSP Program Start Address – Program start address, as viewed by the DSP.

12.4 Commands (continued)

12.4.9 READ_PCI_CONFIG

Command Id: **0x9**

Description: Reads from a location in PCI Configuration Space.

Command Format (32–bit words):

Command Header (five 32–bit words)
Device
Function
Register Offset
Return Value

Parameters: **Device** – Reference number of the PCI Device to read. Set to one of the following values:

- **1** – Tundra Universe II VME Interface
- **2** – PMC Site
- **3** – Intel 82559 Ethernet Controller
- **4** – Intel i960 PCI Bridge
- **5** – Node A PLX PCI 9054
- **6** – Node B PLX PCI 9054
- **7** – Node C PLX PCI 9054
- **8** – Node D PLX PCI 9054

Function – Reference number of the function to access in a multi–function device. Range is **0** to **7**.

Register Offset – Byte offset to one of the device's registers in PCI Configuration Space.

Return Value – Contents of the device's register, returned by the Command Processor.

12.4 Commands (continued)

12.4.10 WRITE_PCI_CONFIG

Command Id: `0xA`

Description: Writes to a location in PCI Configuration Space.

Command Format (32–bit words):

Command Header (five 32–bit words)
Device
Function
Register Offset
Register Value
Register Mask

Parameters: **Device** – Reference number of the PCI Device to write to. Set to one of the following values:

- **1** – Tundra Universe II VME Interface
- **2** – PMC Site
- **3** – Intel 82559 Ethernet Controller
- **4** – Intel i960 PCI Bridge
- **5** – Node A PLX PCI 9054
- **6** – Node B PLX PCI 9054
- **7** – Node C PLX PCI 9054
- **8** – Node D PLX PCI 9054

Function – Reference number of the function to access in a multi–function device. Range is **0** to **7**.

Register Offset – Byte offset to one of the device's registers in PCI Configuration Space.

Register Value – Data value to write into the specified device's register.

Register Mask – Data mask specifying the bits in the device's registers to modify. When the mask is set to **0xFFFFFFFF**, the entire register is written with the data value. Otherwise, only the bits in the mask that are set to a **1** are modified in the register.

12.4 Commands (continued)

12.4.11 XFER_SDRAM_TO_VME

Command Id: `0x10`

Description: Transfers a block of data from Global SDRAM to VME. This command uses the DMA controller in the Universe II VME chip to perform the transfer. When the DMA controller is being used by another command, this command will wait for it to be available before proceeding with the transfer.

Command Format (32–bit words):

Command Header (five 32–bit words)
Global SDRAM Address
VME Address
Transfer Size
DMA Descriptor
DMA Return Status
Return Value

Parameters: **Global SDRAM Address** – Source Global SDRAM address of the data to transfer, as viewed from PCI. This address must be in the range `0x10000000` to `0x11FFFFFF` (`0x10000000` to `0x10FFFFFF` for 32–bit memory mode, set by jumper **J10**, Section 2.2.4).

VME Address – Destination VME address of the data to transfer. The range depends on the VME Address Mode:

```
A16 - 0 to 0xFFFF
A24 - 0 to 0xFFFFFF
A32 - 0 to 0xFFFFFFFF
```

Transfer Size – Number of bytes to transfer. The transfer size must be in the range `1` to `0xFFFFFF`.

12.4 Commands (continued)

12.4.11 XFER_SDRAM_TO_VME (continued)

Parameters: (continued)

DMA Descriptor – This parameter defines the transfer's data width, address mode, and transfer type. Combine the following constants to select the attribute for the transfer:

Data Width:

0x000000 – 8–bit Data
0x400000 – 16–bit Data
0x800000 – 32–bit Data
0xC00000 – 64–bit Data

Address Mode:

0x000000 – A16 Address Space
0x010000 – A24 Address Space
0x020000 – A32 Address Space

Address Modifier Codes:

0x000000 – Data AM Code
0x004000 – Program AM Code
0x000000 – Non–Privileged AM Code
0x001000 – Supervisor AM Code

Transfer Type:

0x000000 – Single Cycle Transfer
0x000100 – Block Transfer

DMA Return Status – This return parameter contains the status of the DMA. The return status may contain one or more of the following constants:

0x0100 - DMA Programming Protocol Error – set when the lower 3 address bits in the Global SDRAM and VME Address differ
0x0200 - VME Bus Error
0x0400 - PCI Bus Error
0x0800 - DMA is done
0x2000 - DMA is halted
0x4000 - DMA is stopped
0x8000 - DMA is active

Return Value – Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.12 XFER_VME_TO_SDRAM

Command Id: `0x11`

Description: Transfers a block of data from VME to Global SDRAM. This command uses the DMA controller in the Universe II VME chip to perform the transfer. When the DMA controller is being used by another command, this command will wait for it to be available before proceeding with the transfer.

Command Format (32–bit words):

Command Header (five 32–bit words)
Global SDRAM Address
VME Address
Transfer Size
DMA Descriptor
DMA Return Status
Return Value

Parameters: **Global SDRAM Address** – Destination Global SDRAM address for storing the data, as viewed from PCI. This address must be in the range `0x10000000` to `0x11FFFFFF` (`0x10000000` to `0x10FFFFFF` for 32–bit memory mode, set by jumper **J10**, Section 2.2.4).

VME Address – Source VME address of the data to transfer. The range depends on the VME Address Mode:

A16 - `0 to 0xFFFF`
A24 - `0 to 0xFFFFFF`
A32 - `0 to 0xFFFFFFFF`

Transfer Size – Number of bytes to transfer. The transfer size must be in the range `1` to `0xFFFFFF`.

12.4 Commands (continued)

12.4.12 XFER_VME_TO_SDRAM (continued)

Parameters: (continued)

DMA Descriptor – This parameter defines the transfer's data width, address mode, and transfer type. Combine the following constants to select the attribute for the transfer:

Data Width:

0x000000 – 8–bit Data
0x400000 – 16–bit Data
0x800000 – 32–bit Data
0xC00000 – 64–bit Data

Address Mode:

0x000000 – A16 Address Space
0x010000 – A24 Address Space
0x020000 – A32 Address Space

Address Modifier Codes:

0x000000 – Data AM Code
0x004000 – Program AM Code
0x000000 – Non–Privileged AM Code
0x001000 – Supervisor AM Code

Transfer Type:

0x000000 – Single Cycle Transfer
0x000100 – Block Transfer

DMA Return Status – This return parameter contains the status of the DMA. The return status may contain one or more of the following constants:

0x0100 - DMA Programming Protocol Error – set when the lower 3 address bits in the Global SDRAM and VME Address differ
0x0200 - VME Bus Error
0x0400 - PCI Bus Error
0x0800 - DMA is done
0x2000 - DMA is halted
0x4000 - DMA is stopped
0x8000 - DMA is active

Return Value – Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.13 XFER_SDRAM_TO_PMC

Command Id: `0x16`

Description: Transfers a block of data from Global SDRAM to the PMC Site. This command uses one of the two primary bus DMA controllers in the i960 to perform the transfer. When both DMA controllers are being used by other commands, this command will wait for one of them to be available before proceeding with the transfer.

Command Format (32–bit words):

Command Header (five 32–bit words)
Global SDRAM Address
PMC Address
Transfer Size
Return Value

Parameters: **Global SDRAM Address** – Source Global SDRAM address of the data to transfer, as viewed from PCI. This address must be in the range `0x10000000` to `0x11FFFFFF` (`0x10000000` to `0x10FFFFFF` for 32–bit memory mode, set by jumper **J10**, Section 2.2.4).

PMC Address – Destination PMC address of the data to transfer, as viewed by PCI. This address must be in the range `0xF0000000` to `0xFFFFFFFF`.

Transfer Size – Number of bytes to transfer. The transfer size must be in the range `1` to `0xFFFFFF`.

Return Value – Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.14 XFER_PMC_TO_SDRAM

Command Id: `0x17`

Description: Transfers a block of data from the PMC Site to Global SDRAM. This command uses one of the two primary bus DMA controllers in the i960 to perform the transfer. When both DMA controllers are being used by other commands, this command will wait for one of them to be available before proceeding with the transfer.

Command Format (32−bit words):

Command Header (five 32−bit words)
Global SDRAM Address
PMC Address
Transfer Size
Return Value

Parameters: **Global SDRAM Address** − Destination Global SDRAM address for storing the data, as viewed from PCI. This address must be in the range **`0x10000000`** to **`0x11FFFFFF`** (**`0x10000000`** to **`0x10FFFFFF`** for 32−bit memory mode, set by jumper **J10**, Section 2.2.4).

PMC Address − Source PMC address of the data to transfer, as viewed by PCI. This address must be in the range **`0xF0000000`** to **`0xFFFFFFFF`**.

Transfer Size − Number of bytes to transfer. The transfer size must be in the range **`1`** to **`0xFFFFFF`**.

Return Value − Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.15 XFER_SDRAM_TO_DSP

Command Id: `0x1A`

Description: Transfers a block of data from Global SDRAM to DSP Memory. This command uses one of the DMA Controllers in the PLX PCI 9054 to perform the transfer. When the DMA controller is being used by other commands, this command will wait for it to be available before proceeding with the transfer.

Command Format (32–bit words):

Command Header (five 32–bit words)		
Global SDRAM Address		
B	Reserved (Set field to 0)	DSP
DSP Address		
Transfer Size		
Return Value		

Parameters: **Global SDRAM Address** – Source Global SDRAM address of the data to transfer, as viewed from PCI. This address must be in the range **`0x10000000`** to **`0x11FFFFFF`** (**`0x10000000`** to **`0x10FFFFFF`** for 32–bit memory mode, set by jumper **J10**, Section 2.2.4).

B (Byte Swap) – This single–bit parameter specifies whether the data written should be endian byte–swapped (by the PLX PCI 9054) during the transfer. Set to **`1`** for byte swapping.

DSP – This 8–bit parameter specifies the DSP to access. Set to one of the following values:

`0` - DSP A **`1`** - DSP B
`2` - DSP C **`3`** - DSP D

DSP Address – Destination DSP address of the data to transfer, as viewed by the DSP.

Transfer Size – Number of bytes to transfer. The transfer size must be a multiple of **`4`**, from **`4`** to **`0x4000`**.

Return Value – Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.16 XFER_DSP_TO_SDRAM

Command Id: `0x1B`

Description: Transfers a block of data from DSP Memory to Global SDRAM. This command uses one of the DMA Controllers in the PLX PCI 9054 to perform the transfer. When the DMA controller is being used by other commands, this command will wait for it to be available before proceeding with the transfer.

Command Format (32–bit words):

Command Header (five 32–bit words)		
Global SDRAM Address		
B	Reserved (Set field to 0)	DSP
DSP Address		
Transfer Size		
Return Value		

Parameters: **Global SDRAM Address** – Destination Global SDRAM address for storing the data, as viewed from PCI. This address must be in the range `0x10000000` to `0x11FFFFFF` (`0x10000000` to `0x10FFFFFF` for 32–bit memory mode, set by jumper **J10**, Section 2.2.4).

B (Byte Swap) – This single–bit parameter specifies whether the data written should be endian byte–swapped (by the PLX PCI 9054) during the transfer. Set to `1` for byte swapping.

DSP – This 8–bit parameter specifies the DSP to access. Set to one of the following values:

`0` - DSP A `1` - DSP B
`2` - DSP C `3` - DSP D

DSP Address – Source DSP address of the data to transfer, as viewed by the DSP.

Transfer Size – Number of bytes to transfer. The transfer size must be a multiple of `4`, from `4` to `0x4000`.

Return Value – Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.17 I960_XFER_SDRAM_TO_DSP

Command Id: `0x1C`

Description: Transfers a block of data from Global SDRAM to DSP Memory. This command uses the i960's Secondary Bus DMA Controller to perform the transfer. When the DMA controller is being used by other commands, this command will wait for it to be available before proceeding with the transfer.

Command Format (32–bit words):

Command Header (five 32–bit words)		
Global SDRAM Address		
B	Reserved (set to 0)	DSP
DSP Address		
Transfer Size		
Return Value		

Parameters: **Global SDRAM Address** – Source Global SDRAM address of the data to transfer, as viewed from PCI. This address must be in the range **`0x10000000`** to **`0x11FFFFFF`** (**`0x10000000`** to **`0x10FFFFFF`** for 32–bit memory mode, set by jumper **J10**, Section 2.2.4).

B (Byte Swap) – This single–bit parameter specifies whether the data written should be endian byte–swapped (by the PLX PCI 9054) during the transfer. Set to **`1`** for byte swapping.

DSP – This 8–bit parameter specifies the DSP to access. Set to one of the following values:

`0` - DSP A **`1`** - DSP B
`2` - DSP C **`3`** - DSP D

DSP Address – Destination DSP address of the data to transfer, as viewed by the DSP.

Transfer Size – Number of bytes to transfer. The transfer size must be a multiple of **`4`**, from **`4`** to **`0x4000`**.

Return Value – Actual number of bytes transferred, returned by the Command Processor on command completion.

12.4 Commands (continued)

12.4.18 TCP_SOCKET

Command Id: `0x20`

Description: Creates an unbound socket in a communications domain, and returns a **Socket File Descriptor** that can be used in functions that operate on sockets.

Command Format (32−bit words):

Command Header (five 32−bit words)
Domain
Type
Protocol
Returned Socket File Descriptor
Returned Error Status

Parameters: **Domain** − Address family used in the communication domain within which communication will take place. Currently, the only address family supported is AF_INET (Internetwork: UDP, TCP, ...), and its decimal code is **2**.

Type − Code for the socket type to create. Currently, the following decimal codes are supported:

1 SOCK_STREAM − Provides sequenced, reliable, two−way connection−based byte streams; it may provide a transmission mechanism for out−of−band data.

2 SOCK_DGRAM − Supports datagrams (connectionless, unreliable messages of a fixed maximum length).

3 SOCK_RAW − Provides raw network protocol access.

5 SOCK_SEQPACKET − Provides a sequenced, reliable, two−way connection−based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each read system call.

12.4 Commands (continued)

12.4.18 TCP_SOCKET (continued)

Parameters: (continued)

Protocol – Code for the protocol to use with this socket. Specifying a protocol of **0** causes the socket to use a default protocol appropriate for the requested socket type. Currently, the following decimal codes are supported:

0 IPPROTO_IP – Internet Protocol

1 IPPROTO_ICMP – Internet Control Message Protocol

4 IPPROTO_IPV4 – Internet Protocol inside Internet Protocol

6 IPPROTO_TCP – Transmission Control Protocol

17 IPPROTO_UCP – User Datagram Protocol

Returned Socket File Descriptor – This return parameter contains a file descriptor that can be used in later function calls that operate on sockets.

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.19 TCP_BIND

Command Id: `0x21`

Description: Assigns an address to an unnamed socket. Initially, Sockets are created unnamed. Bind gives the socket the local address.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Local Address
Local Address Size
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Local Address – This parameter is a data structure that contains the address to bind to the socket. The size and format vary depending on the address family of the socket.

Local Address Size – Data structure size, in bytes, of the **Local Address** data structure above.

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.20 TCP_LISTEN

Command Id: **0x22**

Description: Listens for a connection on a socket. This command applies to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Queue Limit
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Queue Limit – Maximum length the queue of pending connections may grow to. Typically, set this parameter to decimal **128**, SOMAXCONN.

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.21 TCP_ACCEPT

Command Id: `0x23`

Description: Accepts a connection on a socket. This command extracts the first connection request on the queue of pending connections, creates a new socket with the same properties as the specified **Socket File Descriptor**, and allocates a new file descriptor for this socket. This command applies to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Returned Source Address
Returned Source Address Size
Returned Socket File Descriptor
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Returned Source Address – This return parameter is a data structure that contains the address of the connecting entity. The size and format vary depending on the address family of the socket.

Returned Source Address Size – This return parameter contains the size in bytes of the **Returned Source Address** data structure above.

Returned Socket File Descriptor – This return parameter contains a file descriptor that can be used in subsequent read/write commands that operate on sockets (Section 12.4.23 or 12.4.24).

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.22 TCP_CONNECT

Command Id: `0x25`

Description: This command initiates a connection on a socket.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Server Address
Server Address Size
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Server Address – This parameter is a data structure that contains the server's address. The size and format vary depending on the address family of the socket.

Server Address Size – Data structure size, in bytes, of the **Server Address** data structure above.

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.23 TCP_READ

Command Id: `0x26`

Description: Reads a block of data from a socket. This command will not complete until some or all of the data is available. To override this behavior, make the socket non−blocking by sending the IOCTL command and setting FIONBIO to a **1** (Section 12.4.31).

Command Format (32−bit words):

Command Header (five 32−bit words)
Socket File Descriptor
Buffer Address
Buffer Size
Flags
Returned Value
Returned Error Status

Parameters: **Socket File Descriptor** − Socket's file descriptor that was returned in the TCP_SOCKET or TCP_ACCEPT command (Section 12.4.18 or 12.4.21).

Buffer Address − Destination address of the buffer used to store the data, as viewed from the PCI bus.

Buffer Size − Destination data buffer size, in bytes.

Flags − This bit−mask parameter may include one or more of the following constants:

0x00000002 WAIT_FOR_ALL_DATA − If set, wait for the buffer to be completely filled with data. If not set, wait for some data to be available from socket. To override this behavior, make the socket non−blocking using the IOCTL command (Section 12.4.31).

0x80000000 ENDIAN_SWAP (BYTE_SWAP) − If set, swap data on transfer from Global SDRAM to DSP Memory. If not set, data is not swapped during transfer.

12.4 Commands (continued)

12.4.23 TCP_READ (continued)

Parameters (continued)**:**

Returned Value – This return parameter contains the number of bytes actually transferred.

Returned Error Status – This parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4.24 TCP_WRITE

Command Id: **0x27**

Description: Writes a block of data to a socket.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Buffer Address
Buffer Size
Returned Value
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET or TCP_ACCEPT command (Section 12.4.18 or 12.4.21).

Buffer Address – Source address of the buffer containing the data to send, as viewed from the PCI bus.

Buffer Size – Source data buffer size, in bytes.

Returned Value – This return parameter contains the number of bytes actually transferred.

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.25 TCP_CLOSE

Command Id: `0x28`

Description: Closes a socket connection.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET or TCP_ACCEPT command (Section 12.4.18 or 12.4.21).

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.26 TCP_SELECT

Command Id: `0x29`

Description: Waits for a number of socket file descriptors to change status. Three independent sets of descriptors are watched to see if characters are available for reading, to see if it is all right to write to them, and for exceptions.

Command Format (32–bit words):

Command Header (five 32–bit words)
Descriptor
Read Socket File Descriptors
Write Socket File Descriptors
Exception Socket File Descriptors
Timeout
Returned Value
Returned Error Status

Parameters: **Descriptor** – Highest–numbered socket file descriptor in any of the three sets, +1.

Read SFDs – Address to a list of socket file descriptors that will be watched to see if characters are available for reading.

Write SFDs – Address to a list of socket file descriptors that will be watched to see if it is all right to immediately write to them.

Exception SFDs – Address to a list of socket file descriptors that will be watched for exceptions.

Timeout – Maximum amount of time elapsed before Select returns. If set to **0**, Select returns immediately. If set to **-1**, the Select command can block indefinitely.

Returned Value – This return parameter contains the number of socket file descriptors that have changed status.

Returned Error Status – Upon command completion, this parameter contains a **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.27 TCP_SENDTO

Command Id: `0x2A`

Description: Transmits a message to another socket.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Message Address
Message Size
Flags
To Address
To Address Size
Returned Value
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Message Address – Source address of the data buffer containing the message, as viewed from the PCI bus.

Message Size – Message size, in bytes.

12.4 Commands (continued)

12.4.27 TCP_SENDTO (continued)

Parameters: (continued)

Flags – This bit–mask parameter may include one or more of the following constants:

0x1 MSG_OOB – Sends out–of–band data on sockets that support this notion (such as SOCK_STREAM); the underlying protocol must also support out–of–band data.

0x4 MSG_DONTROUTE – Bypasses the usual routing table lookup and sends the packet directly to the interface described by the destination address. This is usually used only by diagnostic or routing programs.

0x80 MSG_DONTWAIT – Enables non–blocking operation; if the operation would block, EAGAIN is returned.

To Address – This parameter is a data structure that contains the address of the receiver. The size and format vary depending on the address family of the socket.

To Address Size – Data structure size, in bytes, of the **To Address** data structure above.

Returned Value – This return parameter contains the number of bytes actually transferred.

Returned Error Status – Upon command completion, this parameter contains **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.28 TCP_RECVFROM

Command Id: `0x2B`

Description: Receives a message from another socket.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Buffer Address
Buffer Size
Flags
Returned From Address
Returned From Address Size
Returned Value
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Buffer Address – Destination address of the data buffer used to store the incoming message, as viewed from the PCI bus.

Buffer Size – Destination data buffer size, in bytes.

12.4 Commands (continued)

12.4.28 TCP_RECVFROM (continued)

Parameters: (continued)

Flags – This bit–mask parameter may include one or more of the following constants:

0x1 MSG_OOB – Requests receipt of out–of–band data that would not be received in the normal data stream. Some protocols place expedited data at the head of the normal data queue, and thus this flag cannot be used with such protocols.

0x2 MSG_PEEK – Causes the receive operation to return data from the beginning of the receive queue without removing that data from the queue. Thus, a subsequent receive call will return the same data.

0x40 MSG_WAITALL – Requests that the operation block until the full request is satisfied. However, the call may still return less data than requested if a signal is caught, an error or disconnect occurs, or the next data to be received is of a different type than that returned.

Returned From Address – Upon command completion, this return parameter is a data structure that contains the address of the sender. The size and format vary depending on the address family of the socket.

Returned From Address Size – This return parameter contains the size in bytes of the **Returned From Address** data structure above.

Returned Value – This return parameter contains the number of bytes actually transferred.

Returned Error Status – Upon command completion, this parameter contains **0** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.29 TCP_SETSOCKOPT

Command Id: `0x2C`

Description: Sets the specified option for a socket.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Level
Option Name
Option Value
Option Value Size
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Level – Protocol level for the socket. Typically, set this parameter to value **`0xFFFF`**, SOL_SOCKET, to manipulate options at the socket level.

Option Name – Option name of the option to set. The header file **`socket.h`** provided with Pentek's ReadyFlow libraries contains a list of available options.

Option Value – Address containing the option value. For most options, the option value is a single 32–bit integer value.

Option Value Size – Data size, in bytes, of the **Option Value** parameter above.

Returned Error Status – Upon command completion, this parameter contains **`0`** when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.30 TCP_GETSOCKOPT

Command Id: `0x2D`

Description: Gets the current setting of the specified option.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Level
Option Name
Returned Option Value
Returned Option Value Size
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Level – Protocol level of the socket. Typically, set this parameter to value `0xFFFF`, SOL_SOCKET, to manipulate options at the socket level.

Option Name – Option name of the option to view. The header file `socket.h` provided with Pentek's ReadyFlow contains a list of available options.

Returned Option Value – Upon command completion, this parameter contains the returned option value. For most options, the option value is a single 32–bit integer value. The SO_LINGER option returns a structure containing two 32–bit integers, therefore, space has been reserved to support it.

Returned Option Value Size – This return parameter contains the size, in bytes, of the **Returned Option Value** parameter above.

Returned Error Status – Upon command completion, this parameter contains `0` when successful or an **errno** error code when a failure occurs.

12.4 Commands (continued)

12.4.31 IOCTL

Command Id: `0x2E`

Description: Controls a TCP socket's operating characteristics.

Command Format (32–bit words):

Command Header (five 32–bit words)
Socket File Descriptor
Command
Command Value
Returned Error Status

Parameters: **Socket File Descriptor** – Socket's file descriptor that was returned in the TCP_SOCKET command (Section 12.4.18).

Command – Specifies the IOCTL command. The IOCTL commands supported are:

0x80047350 FIONBIO – Setting the **Command Value** to a **1** makes the socket non–blocking for the TCP_READ command (Section 12.4.23).

0x40047352 FIONREAD – This command will return the number of available characters in the socket.

Command Value – This read or write parameter contains the value of the IOCTL command. The value is a single 32–bit unsigned integer value.

For FIONBIO, set this parameter to a **1** to make the socket non–blocking.

For FIONREAD, this return parameter will contain the number of available characters in the socket.

Returned Error Status – Upon command completion, this parameter contains **0** when successful or an **errno** error code when a failure occurs.

NOTE: This command is available only in Command Processor version 1.9 or later.

12.4 Commands (continued)

12.4.32 GENERATE_VME_INT

Command Id: `0x40`

Description: Generates a VMEbus Interrupt.

Command Format (32–bit words):

Command Header (five 32–bit words)
Interrupt Level
Interrupt Vector

Parameters: **Interrupt Level** – VME Interrupt level to generate. This parameter must be within the range `1` to `7`.

Interrupt Vector – VME Interrupt Vector for this interrupt. This parameter must be an even number in the range `0` to `254`.

12.4.33 WAIT_ON_VME_INT

Command Id: `0x41`

Description: Waits for a VMEbus Interrupt to occur. This command enables the i960 handling of the specified VME Interrupt. When the interrupt occurs, it will not be re–enabled until this command is issued again.

Command Format (32–bit words):

Command Header (five 32–bit words)
Interrupt Level
Returned Interrupt Vector

Parameters: **Interrupt Level** – VME Interrupt level being waited on. This parameter must be within the range `1` to `7`.

Returned Interrupt Vector – This return parameter contains the VME Interrupt Vector created by the device that generated the VME interrupt.

12.4 Commands (continued)

12.4.34 GENERATE_SYSRESET

Command Id: `0x42`

Description: Generates a SYSRESET on the VMEbus. This command also resets the Model 4292/4293 board if jumper **J21** is removed from the PCB (see Section 2.2.7).

Command Format (32–bit words):

Command Header (five 32–bit words)

12.4.35 WAIT_ON_INT

Command Id: `0x48`

Description: Waits for an interrupt to occur. This command can wait on PLX PCI 9054 Doorbell Interrupts, DSP i960 interrupts, or a PMC Interrupt. When SwiftNet's API or SNIO is being used, the DSP i960 interrupts are reserved for SwiftNet.

Command Format (32–bit words):

Command Header (five 32–bit words)
Interrupt Source
Doorbell Mask

Parameters: **Interrupt Source** – Interrupt condition being waited on. Set to only one of the following values:

0x0001 - Node A PLX PCI 9054 Doorbell Interrupt
0x0002 - Node B PLX PCI 9054 Doorbell Interrupt
0x0004 - Node C PLX PCI 9054 Doorbell Interrupt
0x0008 - Node D PLX PCI 9054 Doorbell Interrupt
0x0010 - DSP A i960 Interrupt
0x0020 - DSP B i960 Interrupt
0x0040 - DSP C i960 Interrupt
0x0080 - DSP D i960 Interrupt
0x1000 - PMC Interrupt

Doorbell Mask – When the interrupt source is one of the PLX PCI 9054 Doorbell Interrupts, this parameter specifies the bits in the Doorbell Register that must be set for this command to complete. Multiple messages can wait on a single Doorbell Register provided that they use different doorbell masks. Set to any value between **1** and **0xFFFFFFFF**.

12.4 Commands (continued)

12.4.36 GET_DATA_BUFFER

Command Id: **0x50**

Description: Allocates a data buffer in Global SDRAM.

Command Format (32–bit words):

Command Header (five 32–bit words)
Buffer Size
Return Buffer Address

Parameters: **Buffer Size** – Buffer size, in bytes, of the requested data buffer.

Return Buffer Address – Allocated data buffer address returned by Command Processor, as viewed on the PCI Bus. This parameter should be in the range **0x10410000** to **0x11FFFFFF** (**0x10410000** to **0x10FFFFFF** if set for 32–bit memory mode by jumper **J10**, Section 2.2.4).

12.4.37 FREE_DATA_BUFFER

Command Id: **0x51**

Description: Frees a data buffer in Global SDRAM previously allocated by the GET_DATA_BUFFER command (Section 12.4.36).

Command Format (32–bit words):

Command Header (five 32–bit words)
Buffer Address

Parameters: **Buffer Address** – PCI Bus address of the allocated data buffer to free. This parameter must be in the range **0x10410000** to **0x11FFFFFF** (**0x10410000** to **0x10FFFFFF** if set for 32–bit memory mode by jumper **J10**, Section 2.2.4).

12.4 Commands (continued)

12.4.38 GET_SEMAPHORE

Command Id: `0x54`

Description: Acquires semaphores for one or more Model 4292/4293 resources. Various resources on the board can only be accessed by one processor at a time. This command will wait for a resource to be available. Once this command completes, all other messages and commands are held off from accessing the resource until the semaphore is freed using the FREE_SEMAPHORE command (Section 12.4.39).

Command Format (32–bit words):

Command Header (five 32–bit words)
Semaphore Mask

Parameters: **Semaphore Mask** – Bit mask identifying the semaphores to acquire. Combine one or more of the following constants for each applicable resource:

0x00000001 - Node A PLX PCI 9054 Registers
0x00000002 - Node B PLX PCI 9054 Registers
0x00000004 - Node C PLX PCI 9054 Registers
0x00000008 - Node D PLX PCI 9054 Registers
0x00000010 - DSP A Expansion Host Port
0x00000020 - DSP B Expansion Host Port
0x00000040 - DSP C Expansion Host Port
0x00000080 - DSP D Expansion Host Port
0x00001000 - PMC Registers
0x00010000 - VME Interrupt Level 1 Handler
0x00020000 - VME Interrupt Level 2 Handler
0x00040000 - VME Interrupt Level 3 Handler
0x00080000 - VME Interrupt Level 4 Handler
0x00100000 - VME Interrupt Level 5 Handler
0x00200000 - VME Interrupt Level 6 Handler
0x00400000 - VME Interrupt Level 7 Handler
0x00800000 - Universe II DMA Controller
0x01000000 - General Purpose Semaphore 1
0x02000000 - General Purpose Semaphore 2
0x04000000 - General Purpose Semaphore 3
0x08000000 - General Purpose Semaphore 4

12.4 Commands (continued)

12.4.39 FREE_SEMAPHORE

Command Id: `0x55`

Description: Frees one or more semaphores previously acquired using the GET_SEMAPHORE command (Section 12.4.38).

Command Format (32–bit words):

Command Header (five 32–bit words)
Semaphore Mask

Parameters: **Semaphore Mask** – Bit mask identifying the semaphores to free. Combine one or more of the following constants for each applicable resource:

0x00000001 - Node A PLX PCI 9054 Registers
0x00000002 - Node B PLX PCI 9054 Registers
0x00000004 - Node C PLX PCI 9054 Registers
0x00000008 - Node D PLX PCI 9054 Registers
0x00000010 - DSP A Expansion Host Port
0x00000020 - DSP B Expansion Host Port
0x00000040 - DSP C Expansion Host Port
0x00000080 - DSP D Expansion Host Port
0x00001000 - PMC Registers
0x00010000 - VME Interrupt Level 1 Handler
0x00020000 - VME Interrupt Level 2 Handler
0x00040000 - VME Interrupt Level 3 Handler
0x00080000 - VME Interrupt Level 4 Handler
0x00100000 - VME Interrupt Level 5 Handler
0x00200000 - VME Interrupt Level 6 Handler
0x00400000 - VME Interrupt Level 7 Handler
0x00800000 - Universe II DMA Controller
0x01000000 - General Purpose Semaphore 1
0x02000000 - General Purpose Semaphore 2
0x04000000 - General Purpose Semaphore 3
0x08000000 - General Purpose Semaphore 4

12.4 Commands (continued)

12.4.40 FREE_MSG_FRAME

Command Id: `0x58`

Description Frees a previously allocated message frame (see Section 12.2.1 for obtaining a message frame).

Command Format (32–bit words):

Command Header (five 32–bit words)
Message Frame Address

Parameters: **Message Frame Address** – PCI Bus address of the message frame to free. This parameter should be in the range `0x10400000` to `0x1040FFFF`.

12.4.41 FREE_PROC_RESOURCES

Command Id: `0x59`

Description: Frees any data buffers or message frames that were allocated for a specified Processor Id. This command searches all the message frames to find those with a Processor Id that match the one contained in the message header (Section 12.3.1.9). Each message frame that matches is freed.

Command Format (32–bit words):

Command Header (five 32–bit words)

Parameters: None

Appendix A: Model 4292/4293 Registers Summary

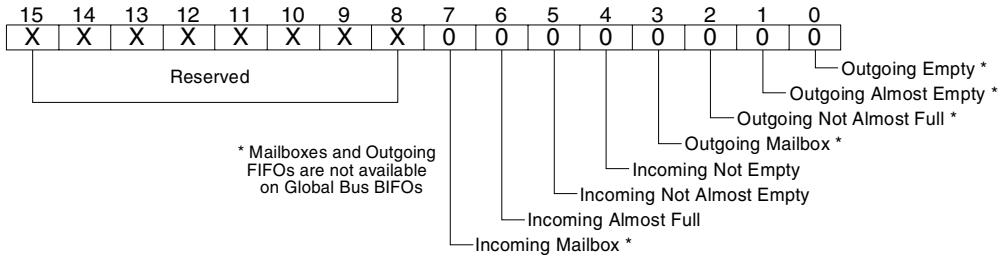
A.1 Introduction

This appendix provides a quick reference summary of the programmable registers on the Model 4292/4293.

A.2 DSP Local Control Registers

Each Master 'C6x processor has access to several local control registers that allow it to process interrupts and control the operation of its resources. These registers are described in detail in Section 3.6 of the manual.

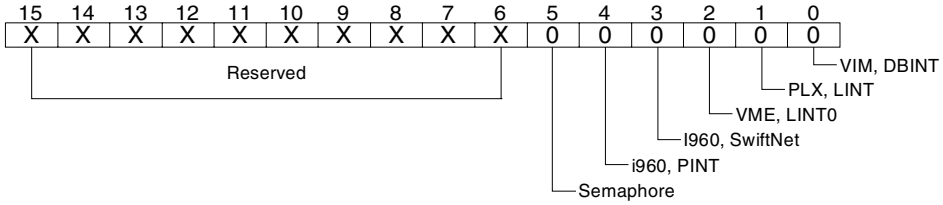
Global Bus FIFO Interrupt Enable Register (BIER0):	R/W @ 0x0220 0008
XBus Inter–Processor XX BIFO Interrupt Enable Register (BIER1):	R/W @ 0x0220 000C
XBus Inter–Processor YY BIFO Interrupt Enable Register (BIER2):	R/W @ 0x0220 0010
Mezzanine BIFO Interrupt Enable Register (BIER3):	R/W @ 0x0220 0014
Global Bus FIFO Flag Register (BIFR0):	R/Clr @ 0x0220 0020
XBus Inter–Processor XX BIFO Flag Register (BIFR1):	R/Clr @ 0x0220 0024
XBus Inter–Processor YY BIFO Flag Register (BIFR2):	R/Clr @ 0x0220 0028
Mezzanine BIFO Flag Register (BIFR3):	R/Clr @ 0x0220 002C
Global Bus FIFO Status Register (BISR0):	R.O. @ 0x0220 0038
XBus Inter–Processor XX BIFO Status Register (BISR1):	R.O. @ 0x0220 003C
XBus Inter–Processor YY BIFO Status Register (BISR2):	R.O. @ 0x0220 0040
Mezzanine BIFO Status Register (BISR3):	R.O. @ 0x0220 0044



A.2 DSP Local Control Registers (continued)

Miscellaneous Interrupt Flag Register (MIFR): R/Clr @ 0x0220 0048

Miscellaneous Interrupt Status Register (MISR): R.O. @ 0x0220 004C



'C6x External Interrupt #4 Enable Register (EIER4): R/W @ 0x0220 0050

'C6x External Interrupt #5 Enable Register (EIER5): R/W @ 0x0220 0054

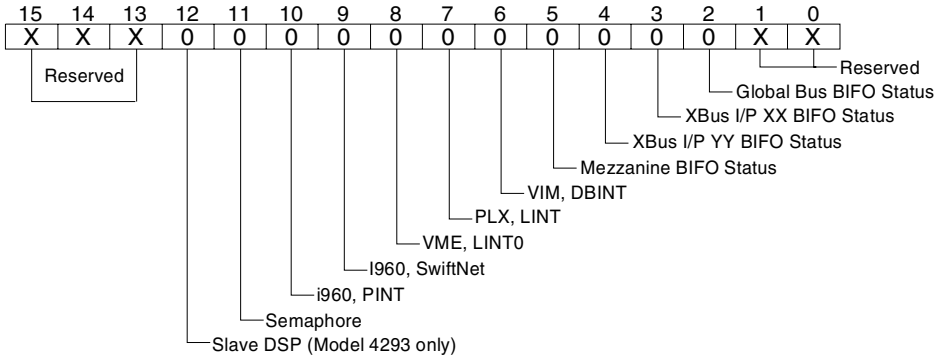
'C6x External Interrupt #6 Enable Register (EIER6): R/W @ 0x0220 0058

'C6x External Interrupt #7 Enable Register (EIER7): R/W @ 0x0220 005C

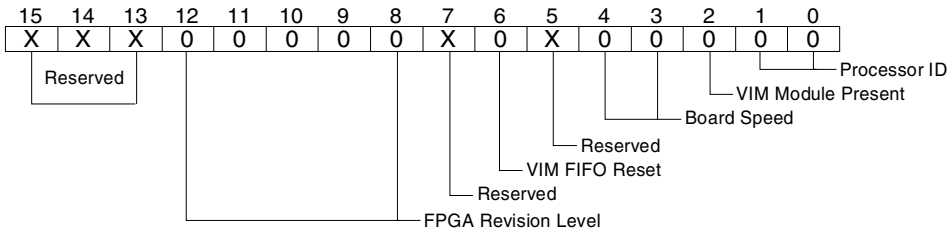
'C6x External Interrupt Mode Register (EIMR): R/W @ 0x0220 006C

Miscellaneous Interrupt Flag Register #2 (MIFR2): R/W @ 0x0220 0074

Miscellaneous Interrupt Status Register #2 (MISR2): R/W @ 0x0220 0078

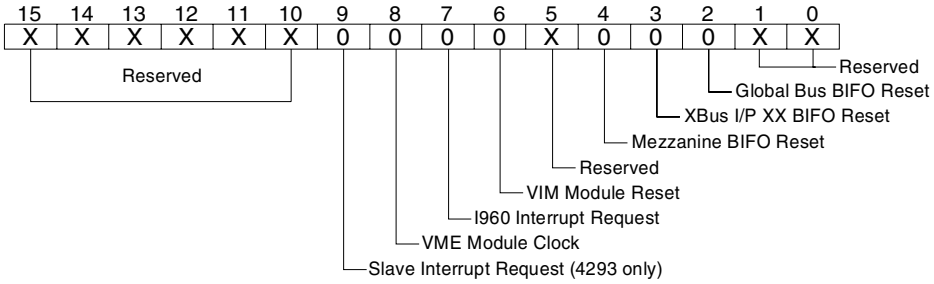


Board Status Register (BSR): R.O. @ 0x0220 0060

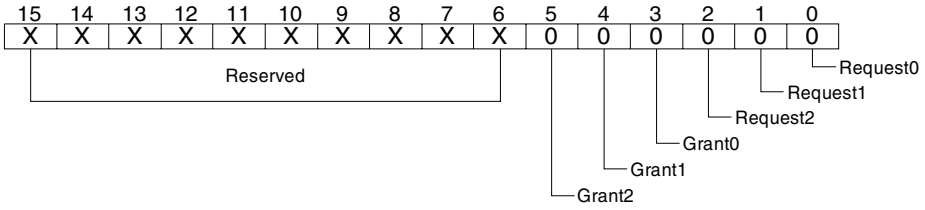


A.2 DSP Local Control Registers (continued)

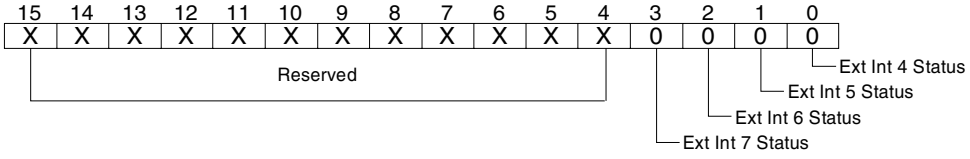
Miscellaneous Control Register (MCR): R/W @ 0x0220 0064



General Purpose Semaphore Register (GPSR): R/W @ 0x0220 0068



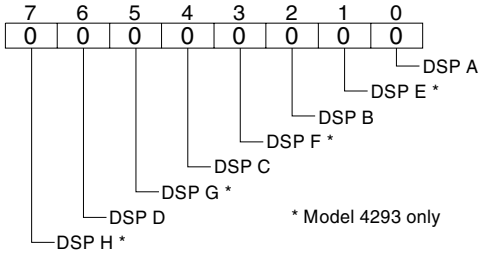
'C6x External Interrupt Status Register (EISR) (Model 4293 only): R/W @ 0x0220 0070



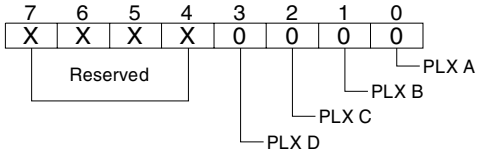
A.3 i960 Control Registers

The i960 I/O processor has access to local control registers that allow it to process interrupts and control the operation of resources available to it. These registers are described in detail in Section 5.5 of the manual.

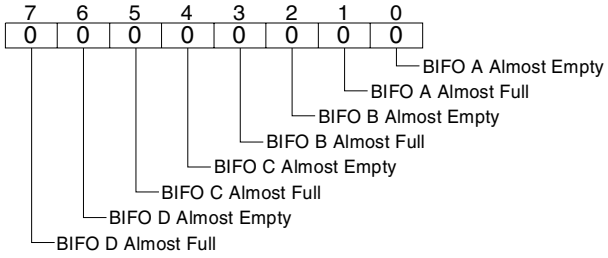
DSP Interrupt Request Register: R/W @ 0xFE00 0000
DSP Interrupt Enable Register: R/W @ 0xFE00 0002
DSP Interrupt Flag Register: R.O. @ 0xFE00 0009



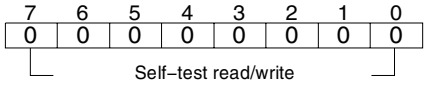
PLX Interrupt Enable Register: R/W @ 0xFE00 0001
PLX Interrupt Status Register: R.O. @ 0xFE00 0008



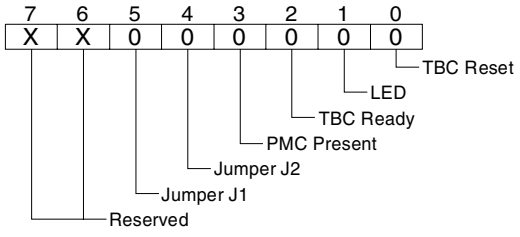
Global Bus BIFO Interrupt Enable Register: R/W @ 0xFE00 0003
Global Bus BIFO Interrupt Status Register: R.O. @ 0xFE00 000A



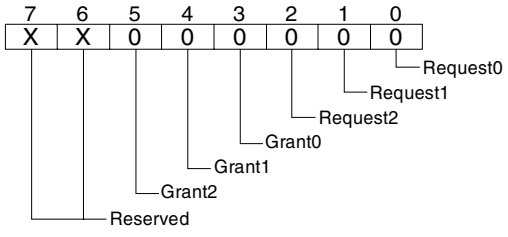
Built–In Self Test #0 Register: R/W @ 0xFE00 0004
Built–In Self Test #1 Register: R/W @ 0xFE00 0005



Control Register: R/W @ 0xFE00 0006



General Purpose Semaphore Register: R/W @ 0xFE00 0007



Appendix B: Model 4292/4293 ID EEPROM

B.1 Introduction

The Model 4292/4293 contains an Identification (ID) EEPROM, with a checksum stored in the last word. When the board is booted, the i960 processor reads this ID EEPROM and generates a 16–bit checksum from the contents read. If the generated checksum does not match the stored value, the processor illuminates its red LED. (Even though the processor illuminates the red LED, the processor is fully functional.)

The contents of the ID EEPROM, plus the calculated checksum, are stored in Global SDRAM (see Section 10.4.5 and Table 10–3). The first word stored is a check flag based on the checksum: valid checksum (**0x00EE CODE**), bad checksum (**0x000B ADC5**), or no VIM module installed (**0xFFFF FFFE**).

You can read the serial ID EEPROM on the i960's I^2C bus at binary address **1010xxxb**, where **xxx** selects one of eight 256–word blocks (see Section 5.4.10).

NOTE: The ID EEPROM is shipped with the board identification information preset by Pentek, and should **NOT** be modified by the user.

B.2 ID EEPROM Format and Contents

Table B–1 on the next page shows the contents of the ID EEPROM, and an brief explanation of the contents of each pair of 16–bit words. The SDRAM location listed in the table is the address where each pair of words is stored, relative to the start address of the Global SDRAM.

The Temperature and Voltage words in the ID EEPROM are used to record the operating temperatures and voltages from sensors on the Model 4292/4293 PCB. The temperature values are stored as degrees C and the voltages are stored as mVolts. (These are stored as binary values, represented by the hex codes **0xXXXX XXXX** in Table B–1.)

During normal operation, the i960 processor reads the board's voltages and temperatures at regular intervals (about 1 minute). It stores the new values in the 'Readings' area of the Global SDRAM (see Table 10–3) and compares them with the latest values from the ID EEPROM. If the new readings are higher, the i960 stores these new readings into the appropriate words of this ID EEPROM. When new voltage/temperature values are stored in this ID EEPROM, a new checksum is generated and stored in the last word of the EEPROM. (The Global SDRAM copy of the EEPROM is also updated when new values and checksum are stored into the ID EEPROM.)

B.2 EEPROM Format Example (continued)

Table B–1: EEPROM Example Code (Model 4292 illustrated)

EEPROM Word Location	SDRAM Location	Contents	Comments
00/01	+0x00	0x00EE 1960	Valid data flag
02/03	+0x04	0x4292 0000	Model, Model Extension (Extension currently unused)
04/05	+0x08	0x0354 2001	PCB number — this is taken from the board
06/07	+0x0C	0x0001 FFFF	Assembly Rev (Rev A = **0001**, B = **0002**, etc.), Option #1 — up to 9 installed options may be specified, when there are no options, '**F**' appears in the entire field
08/09	+0x10	0xFFFF FFFF	Option #2, Option #3
10/11	+0x14	0xFFFF FFFF	Option #4, Option #5
12/13	+0x18	0xFFFF FFFF	Option #6, Option #7
14/15	+0x1C	0xFFFF FFFF	Option #8, Option #9
16/17	+0x20	0x0001 FFFF	PCB Rev (Rev A = **0001**, B = **0002**, etc.), **0xFFFF**
18/19	+0x24	0x3042 921A	Firmware/FPGA Model Number and Rev for key components on the PCB — the first seven hex digits are the Pentek Part Number, and the last hex digit is the Rev level of the part. For example **0x3042 921A** refers to the part number 030.42921, revision A
20/21	+0x28	0x3042 922A	
22/23	+0x2C	0x3042 923A	
24/25	+0x30	0x3042 924A	
26/27	+0x34	0x3042 925A	
28/29	+0x38	0x3042 926A	
30/31	+0x3C	0x3042 927A	
32/33	+0x40	0xXXXX XXXX	Temperature Sensor 1 Peak
34/35	+0x44	0xXXXX XXXX	Temperature Sensor 2 Peak
36/37	+0x48	0xXXXX XXXX	Temperature Sensor 3 Peak
38/39	+0x4C	0xXXXX XXXX	Voltage Sensor 1 Minimum
40/41	+0x50	0xXXXX XXXX	Voltage Sensor 1 Maximum
42/43	+0x54	0xXXXX XXXX	Voltage Sensor 2 Minimum
44/45	+0x58	0xXXXX XXXX	Voltage Sensor 2 Maximum
46/47	+0x5C	0xXXXX XXXX	Voltage Sensor 3 Minimum
48/49	+0x60	0xXXXX XXXX	Voltage Sensor 3 Maximum
50/51	+0x64	0xXXXX XXXX	Voltage Sensor 4 Minimum
52/53	+0x68	0xXXXX XXXX	Voltage Sensor 4 Maximum
54/55	+0x6C	0xXXXX XXXX	Voltage Sensor 5 Minimum
56/57	+0x70	0xXXXX XXXX	Voltage Sensor 5 Maximum
58/59	+0x74	0xFFFF FFFF	
60/61	+0x78	0xFFFF FFFF	
62/63	+0x7C	0x0000 XXXX	Word 62 should always be cleared (**0000**). Word 63 contains the 16–bit checksum of words 0 – 61.

Appendix C: Data Transfer Benchmarks

C.1 Introduction

This appendix presents the data transfer rates between any two blocks of memory on the Model 4292/4293. These transfer rates were acquired using test programs on the Master 'C6x processors and the i960 processor. All transfers were performed using DMA operation between the memory blocks. All results are derived from the measured time for a transfer of 1000 words. The transfer rates are shown in terms of clock cycles.

C.2 Master 'C6x Processor Data Transfer Timing

The following table shows the number of clock cycles it takes for a Master 'C6x processor to move one 32–bit word from one memory resource to another, with DMA enabled.

Table C–1: Master 'C6x Processor Data Transfer Timing

From	To	Clock Cycles
'C6x Internal Data Memory	DSP Local SDRAM	2
DSP Local SDRAM	'C6x Internal Data Memory	3
'C6x Internal Data Memory	Global Bus FIFO	4
Global Bus FIFO	'C6x Internal Data Memory	4
'C6x Internal Data Memory	XBus I/P XX Bi–FIFO	4
XBus I/P XX Bi–FIFO	'C6x Internal Data Memory	4
'C6x Internal Data Memory	XBus I/P YY Bi–FIFO	4
XBus I/P YY Bi–FIFO	'C6x Internal Data Memory	4
'C6x Internal Data Memory	Mezzanine Bi–FIFO	4
Mezzanine Bi–FIFO	'C6x Internal Data Memory	4
'C6x Internal Data Memory	VIM Random Access I/O	81
VIM Random Access I/O	'C6x Internal Data Memory	81
DSP Local SDRAM	VIM Random Access I/O	88
VIM Random Access I/O	DSP Local SDRAM	94
'C6x Internal Data Memory	Global SDRAM	9
Global SDRAM	'C6x Internal Data Memory	19 *
Global SDRAM	'C6x Internal Data Memory	9 **
DSP Local SDRAM	Global SDRAM	17
Global SDRAM	DSP Local SDRAM	19 *
Global SDRAM	DSP Local SDRAM	13 **
'C6x Internal Data Memory	PMC module I/O	9

For transfers over the PCI bus, DMA read access was tested using two different DMA Master Command Codes (assigned using the PLX PCI 9054 internal register CNTRL):
* Memory Read (default)
** Memory Read Multiple

C.3 i960 Processor Data Transfer Timing

The following table shows the number of clock cycles it takes for the i960 processor to move one 32–bit word from one memory resource to another, with DMA enabled.

Table C–2: i960 Processor Data Transfer Timing

From	To	PCI Command *	Clock Cycles
Global SDRAM	Global SDRAM	N/A	33
Node A Global Bus FIFO	Global SDRAM		28
Global SDRAM	Node A Global Bus FIFO		28
PMC module I/O	Global SDRAM	7	30
PMC module I/O	Global SDRAM	6	33
PMC module I/O	Global SDRAM	C	33
PMC module I/O	Global SDRAM	E	33
PMC module I/O	Node A Global Bus FIFO	7	30
PMC module I/O	Node B Global Bus FIFO	7	29
PMC module I/O	Node C Global Bus FIFO	7	30
PMC module I/O	Node D Global Bus FIFO	7	30
Node A Host Port Registers, Prefetch	Global SDRAM	6	67
Node A Host Port Registers, Prefetch	Global SDRAM	C	67
Node A Host Port Registers, Prefetch	Global SDRAM	E	67
Node A Host Port Registers, Non-Prefetch	Global SDRAM	6	815
Node A Host Port Registers, Non-Prefetch	Global SDRAM	C	815
Node A Host Port Registers, Non-Prefetch	Global SDRAM	E	823
Node A Global Bus FIFO	Global SDRAM	7	98

* For PCI bridge transfers, DMA access was tested using the following PCI Command codes (PCI Command is assigned in the i960 DMA Chain Descriptor):
7 – Memory Write
6 – Memory Read
C – Memory Read Multiple
E – Memory Read Line

Refer to Appendix D for the data transfer timing for RACEway operations.

Appendix D: Option 005 RACEway Operation

D.1 Introduction

Option 005 for the Model 4292/4293 is a PMC (PCI Mezzanine Card) module incorporating a complete RACEway RACE++® interface to the PCI bus. This device uses the PXB++ PCI Bridge chip from Mercury Computer Systems. When Option 005 is installed, the Model 4292/4293 supports both outgoing (Master) and incoming (Slave) RACEway operations with other RACEway–equipped VME boards.

- The electrical characteristics of the PCI bus, commonly used in desktop personal computers, define the 32–bit parallel synchronous bus operating at peak rates of 132 MBytes/second.

- RACE++ is a high–speed synchronous backplane bus capable of delivering 32–bit words between VME boards at rates up to 267 MBytes/second. Interlink Modules (ILKs or ILKPs) connect the RACE++ (RACEway) busses of two or more VME boards. The switches on the ILKs are called RACEway Crossbar switches. ILKs can support multiple 267–MB/sec RACE++ transfers simultaneously, depending on the number of slots and crossbar switches utilized.

This manual assumes that the reader is familiar with the RACEway interface and its protocols. The complete RACEway Interlink standard, ANSI/VITA 5.1–1999, is available for purchase from the VMEbus International Trade Association at their web site, **http://www.vita.com**. A Raceway Interlink Functional Specification is available from Mercury Computer Systems' web site, **http://www.mc.com**.

D.2 General Operation

The Option 005 PMC supports both outgoing (Master) and incoming (Slave) RACEway operations on the Model 4292/4293 using the PCI bus. This module provides translation of PCI addresses to or from RACEway Route and Address Words for the PCI address space of the 4292/4293. Any Model 4292/4293 device that has access to the PMC site can use this RACEway interface.

NOTE: The Option 005 PMC Raceway can generate Broadcast/Multicast writes, but cannot receive them.

Pentek's ReadyFlow Board Support Package (BSP) for the Model 4292/4293 provides routines and examples for control and operation of all RACEway functions. Refer to the ReadyFlow Software Support Documentation (Pentek part #801.42920) for description of these capabilities.

D.3 RACEway Transfer Rates

The transfer rates between a Model 4292/4293 Node 'C6x Processor and RACEway are as follows:

Writes to RACEway:

- 128 MBytes/sec when data being written is in the Processor's Internal Data RAM
- 59 MBytes/sec when the data being written is in the Processor's local SDRAM

Reads from RACEway:

- 6.7–29 MBytes/sec from another Model 4292's Global SDRAM when RACEway is clocked at 40 MHz
- 6.0–7.3 MBytes/sec from another Model 4292's Global SDRAM when RACEway is clocked at 66 MHz
- 65 MBytes/sec from a Micro Memory 6490 Board to the Processor's SDRAM when RACEway is clocked at 40 MHz
- 77 MBytes/sec from a Micro Memory 6490 Board to the Processor's IDRAM when RACEway is clocked at 66 MHz

Appendix E: Tundra® Universe II™ Registers

The following table lists the Tundra Universe II Control and Status Registers, their functions, and their memory address offset. The address offset is from the Universe II Registers base address in the applicable memory map (see Section 3.5.2 for the base address for each 'C6x processor, Section 5.4.3 for the i960 processor, Section 6.5.6 for a PCI Bus Master, and Section 7.3.1 for an external VMEbus Master). For description of each register, please refer to the Tundra Universe II User Manual.

Table E–1: Universe II Control and Status Registers

Address Offset	Register Function	Name
0x000	PCI Configuration Space ID	PCI_ID
0x004	PCI Configuration Space Control and Status	PCI_CSR
0x008	PCI Configuration Class	PCI_CLASS
0x00C	PCI Configuration Miscellaneous 0	PCI_MISC0
0x010	PCI Configuration Base Address 0	PCI_BS0
0x014	PCI Configuration Base Address 1	PCI_BS1
0x018 – 0x024	PCI Unimplemented	
0x028 – 0x02C	PCI Reserved	
0x030	PCI Unimplemented	
0x034 – 0x038	PCI Reserved	
0x03C	PCI Configuration Miscellaneous 1	PCI_MISC1
0x040 – 0x0FF	PCI Unimplemented	
0x100	PCI Target Image 0 Control	LSI0_CTL
0x104	PCI Target Image 0 Base Address	LSI0_BS
0x108	PCI Target Image 0 Bound Address	LSI0_BD
0x10C	PCI Target Image 0 Translation Offset	LSI0_TO
0x110	Reserved	
0x114	PCI Target Image 1 Control	LSI1_CTL
0x118	PCI Target Image 1 Base Address	LSI1_BS
0x11C	PCI Target Image 1 Bound Address	LSI1_BD
0x120	PCI Target Image 1 Translation Offset	LSI1_TO
0x124	Reserved	
0x128	PCI Target Image 2 Control	LSI2_CTL
0x12C	PCI Target Image 2 Base Address	LSI2_BS
0x130	PCI Target Image 2 Bound Address	LSI2_BD
0x134	PCI Target Image 2 Translation Offset	LSI2_TO
0x138	Reserved	
0x13C	PCI Target Image 3 Control	LSI3_CTL
0x140	PCI Target Image 3 Base Address	LSI3_BS
0x144	PCI Target Image 3 Bound Address	LSI3_BD
0x148	PCI Target Image 3 Translation Offset	LSI3_TO

Table E–1: Universe II Control and Status Registers		
0x14C – 0x16C	Reserved	
0x170	Special Cycle Control	SCYC_CTL
0x174	Special Cycle PCI Bus Address	SCYC_ADDR
0x178	Special Cycle Swap/Compare Enable	SCYC_EN
0x17C	Special Cycle Compare Data	SCYC_CMP
0x180	Special Cycle Swap Data	SCYC_SWP
0x184	PCI Miscellaneous	LMISC
0x188	Special PCI Target Image	SLSI
0x18C	PCI Command Error Log	L_CMDERR
0x190	PCI Address Error Log	LAERR
0x194 – 0x19C	Reserved	
0x1A0	PCI Target Image 4 Control	LSI4_CTL
0x1A4	PCI Target Image 4 Base Address	LSI4_BS
0x1A8	PCI Target Image 4 Bound Address	LSI4_BD
0x1AC	PCI Target Image 4 Translation Offset	LSI4_TO
0x1B0	Reserved	
0x1B4	PCI Target Image 5 Control	LSI5_CTL
0x1B8	PCI Target Image 5 Base Address	LSI5_BS
0x1BC	PCI Target Image 5 Bound Address	LSI5_BD
0x1C0	PCI Target Image 5 Translation Offset	LSI5_TO
0x1C4	Reserved	
0x1C8	PCI Target Image 6 Control	LSI6_CTL
0x1CC	PCI Target Image 6 Base Address	LSI6_BS
0x1D0	PCI Target Image 6 Bound Address	LSI6_BD
0x1D4	PCI Target Image 6 Translation Offset	LSI6_TO
0x1D8	Reserved	
0x1DC	PCI Target Image 7 Control	LSI7_CTL
0x1E0	PCI Target Image 7 Base Address	LSI7_BS
0x1E4	PCI Target Image 7 Bound Address	LSI7_BD
0x1E8	PCI Target Image 7 Translation Offset	LSI7_TO
0x1EC – 0x1FC	Reserved	
0x200	DMA Transfer Control	DCTL
0x204	DMA Transfer Byte Count	DTBC
0x208	DMA PCI Bus Address	DLA
0x20C	Reserved	
0x210	DMA VMEbus Address	DVA
0x214	Reserved	
0x218	DMA Command Packet Pointer	DCPP
0x21C	Reserved	
0x220	DMA General Control and Status	DGCS
0x224	DMA Linked List Update Enable	D_LLUE
0x228 – 0x2FC	Reserved	

Table E–1: Universe II Control and Status Registers		
0x300	PCI Interrupt Enable	LINT_EN
0x304	PCI Interrupt Status	LINT_STAT
0x308	PCI Interrupt Map 0	LINT_MAP0
0x30C	PCI Interrupt Map 1	LINT_MAP1
0x310	VMEbus Interrupt Enable	VINT_EN
0x314	VMEbus Interrupt Status	VINT_STAT
0x318	VMEbus Interrupt Map 0	VINT_MAP0
0x31C	VMEbus Interrupt Map 1	VINT_MAP1
0x320	Interrupt Status/ID Out	STATID
0x324	VIRQ1 Status/ID Out	V1_STATID
0x328	VIRQ2 Status/ID Out	V2_STATID
0x32C	VIRQ3 Status/ID Out	V3_STATID
0x330	VIRQ4 Status/ID Out	V4_STATID
0x334	VIRQ5 Status/ID Out	V5_STATID
0x338	VIRQ6 Status/ID Out	V6_STATID
0x33C	VIRQ7 Status/ID Out	V7_STATID
0x340	PCI Interrupt Map 2	LINT_MAP2
0x344	VMEbus Interrupt Map 2	VINT_MAP2
0x348	Mailbox 0	MBOX0
0x34C	Mailbox 1	MBOX1
0x350	Mailbox 2	MBOX2
0x354	Mailbox 3	MBOX3
0x358	Semaphore 0	SEMA0
0x35C	Semaphore 1	SEMA1
0x360 – 0x3FC	Reserved	
0x400	Master Control	MAST_CTL
0x404	Miscellaneous Control	MISC_CTL
0x408	Miscellaneous Status	MISC_STAT
0x40C	User AM Codes	USER_AM
0x410 – 0xEFC	Reserved	
0xF00	VMEbus Slave Image 0 Control	VSI0_CTL
0xF04	VMEbus Slave Image 0 Base Address	VSI0_BS
0xF08	VMEbus Slave Image 0 Bound Address	VSI0_BD
0xF0C	VMEbus Slave Image 0 Translation Offset	VSI0_TO
0xF10	Reserved	
0xF14	VMEbus Slave Image 1 Control	VSI1_CTL
0xF18	VMEbus Slave Image 1 Base Address	VSI1_BS
0xF1C	VMEbus Slave Image 1 Bound Address	VSI1_BD
0xF20	VMEbus Slave Image 1 Translation Offset	VSI1_TO
0xF24	Reserved	
0xF28	VMEbus Slave Image 2 Control	VSI2_CTL
0xF2C	VMEbus Slave Image 2 Base Address	VSI2_BS

Table E–1: Universe II Control and Status Registers		
0xF30	VMEbus Slave Image 2 Bound Address	VSI2_BD
0xF34	VMEbus Slave Image 2 Translation Offset	VSI2_TO
0xF38	Reserved	
0xF3C	VMEbus Slave Image 3 Control	VSI3_CTL
0xF40	VMEbus Slave Image 3 Base Address	VSI3_BS
0xF44	VMEbus Slave Image 3 Bound Address	VSI3_BD
0xF48	VMEbus Slave Image 3 Translation Offset	VSI3_TO
0xF4C – 0xF60	Reserved	
0xF64	Location Monitor Control	LM_CTL
0xF68	Location Monitor Base Address	LM_BS
0xF6C	Reserved	
0xF70	VMEbus Register Access Image Control	VRAI_CTL
0xF74	VMEbus Register Access Image Base Address	VRAI_BS
0xF78 – 0xF7C	Reserved	
0xF80	VMEbus CSR Control	VCSR_CTL
0xF84	VMEbus CSR Translation Offset	VCSR_TO
0xF88	VMEbus AM Code Error Log	V_AMERR
0xF8C	VMEbus Address Error Log	VAERR
0xF90	VMEbus Slave Image 4 Control	VSI4_CTL
0xF94	VMEbus Slave Image 4 Base Address	VSI4_BS
0xF98	VMEbus Slave Image 4 Bound Address	VSI4_BD
0xF9C	VMEbus Slave Image 4 Translation Offset	VSI4_TO
0xFA0	Reserved	
0xFA4	VMEbus Slave Image 5 Control	VSI5_CTL
0xFA8	VMEbus Slave Image 5 Base Address	VSI5_BS
0xFAC	VMEbus Slave Image 5 Bound Address	VSI5_BD
0xFB0	VMEbus Slave Image 5 Translation Offset	VSI5_TO
0xFB4	Reserved	
0xFB8	VMEbus Slave Image 6 Control	VSI6_CTL
0xFBC	VMEbus Slave Image 6 Base Address	VSI6_BS
0xFC0	VMEbus Slave Image 6 Bound Address	VSI6_BD
0xFC4	VMEbus Slave Image 6 Translation Offset	VSI6_TO
0xFC8	Reserved	
0xFCC	VMEbus Slave Image 7 Control	VSI7_CTL
0xFD0	VMEbus Slave Image 7 Base Address	VSI7_BS
0xFD4	VMEbus Slave Image 7 Bound Address	VSI7_BD
0xFD8	VMEbus Slave Image 7 Translation Offset	VSI7_TO
0xFDC – 0xFEC	Reserved	
0xFF0	VME CR/CSR Reserved	
0xFF4	VMEbus CSR Bit Clear	VCSR_CLR
0xFF8	VMEbus CSR Bit Set	VCSR_SET
0xFFC	VMEbus CSR Base Address	VCSR_BS

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP