Harrier Application Specific Integrated Circuit (ASIC)

Programmer's Reference Guide

Parts 1 and 2

ASICHRA1/PG1 and ASICHRA2/PG1

May 2001 Edition

Printed in the United States of America.

Safety Summary

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual could result in personal injury or damage to the equipment.

The safety precautions listed below represent warnings of certain dangers of which Motorola is aware. You, as the user of the product, should follow these warnings and all other safety precautions necessary for the safe operation of the equipment in your operating environment.

Ground the Instrument.

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. If the equipment is supplied with a three-conductor AC power cable, the power cable must be plugged into an approved three-contact electrical outlet, with the grounding wire (green/yellow) reliably connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards and local electrical regulatory codes.

Do Not Operate in an Explosive Atmosphere.

Do not operate the equipment in any explosive atmosphere such as in the presence of flammable gases or fumes. Operation of any electrical equipment in such an environment could result in an explosion and cause injury or damage.

Keep Away From Live Circuits Inside the Equipment.

Operating personnel must not remove equipment covers. Only Factory Authorized Service Personnel or other qualified service personnel may remove equipment covers for internal subassembly or component replacement or any internal adjustment. Service personnel should not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, such personnel should always disconnect power and discharge circuits before touching components.

Use Caution When Exposing or Handling a CRT.

Breakage of a Cathode-Ray Tube (CRT) causes a high-velocity scattering of glass fragments (implosion). To prevent CRT implosion, do not handle the CRT and avoid rough handling or jarring of the equipment. Handling of a CRT should be done only by qualified service personnel using approved safety mask and gloves.

Do Not Substitute Parts or Modify Equipment.

Do not install substitute parts or perform any unauthorized modification of the equipment. Contact your local Motorola representative for service and repair to ensure that all safety features are maintained.

Observe Warnings in Manual.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed. You should also employ all other safety precautions which you deem necessary for the operation of the equipment in your operating environment.

Warning

To prevent serious injury or death from dangerous voltages, use extreme caution when handling, testing, and adjusting this equipment and its components.

.

Flammability

All Motorola PWBs (printed wiring boards) are manufactured with a flammability rating of 94V-0 by UL-recognized manufacturers.

EMI Caution

Caution

This equipment generates, uses and can radiate electromagnetic energy. It may cause or be susceptible to electromagnetic interference (EMI) if not installed and used with adequate EMI protection.

Lithium Battery Caution

This product contains a lithium battery to power the clock and calendar circuitry.

Caution

Danger of explosion if battery is replaced incorrectly. Replace battery only with the same or equivalent type recommended by the equipment manufacturer. Dispose of used batteries according to the manufacturer's instructions.

Attention

Il y a danger d'explosion s'il y a remplacement incorrect de la batterie. Remplacer uniquement avec une batterie du même type ou d'un type équivalent recommandé par le constructeur. Mettre au rebut les batteries usagées conformément aux instructions du fabricant.

Vorsicht

Explosionsgefahr bei unsachgemäßem Austausch der Batterie. Ersatz nur durch denselben oder einen vom Hersteller empfohlenen Typ. Entsorgung gebrauchter Batterien nach Angaben des Herstellers.

CE Notice (European Community)

This is a Class A product. In a domestic environment, this product may cause radio interference, in which case the user may be required to take adequate measures.

Motorola Computer Group products with the CE marking comply with the EMC Directive (89/336/EEC). Compliance with this directive implies conformity to the following European Norms:

> EN55022 “Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment”; this product tested to Equipment Class A
>
> EN55024 “Information Technology Equipment-Immunity characteristics-Limits and methods of measurement”

Board products are tested in a representative system to show compliance with the above mentioned requirements. A proper installation in a CE-marked system will maintain the required EMC/safety performance.

In accordance with European Community directives, a “Declaration of Conformity” has been made and is available on request. Please contact your sales representative.

Notice

While reasonable efforts have been made to assure the accuracy of this document, Motorola, Inc. assumes no liability resulting from any omissions in this document, or from the use of the information obtained therein. Motorola reserves the right to revise this document and to make changes from time to time in the content hereof without obligation of Motorola to notify any person of such revision or changes.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to the Motorola Computer Group website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of Motorola, Inc.

It is possible that this publication may contain reference to or information about Motorola products (machines and programs), programming, or services that are not available in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by Motorola, Inc.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

Motorola, Inc.
Computer Group
2900 South Diablo Way
Tempe, Arizona 85282

Contents

About This Manual

Summary of Changes xxi
Overview of Contents xxi
Manual Terminology xxii
Comments and Suggestions xxiii
Conventions Used in This Manual xxiv
Bus Naming xxiv
Bit Ordering xxv
Register and Bit Naming xxv
Register Descriptions xxvi

CHAPTER 1 Introduction

Overview 1-1
Features 1-1
Functional Blocks of Harrier 1-4

CHAPTER 2 Functional Descriptions

Clocking 2-1
SDRAM Interface 2-4
PowerPC Bus Slave 2-5
Responding to Address Transfers 2-5
Completing Data Transfers 2-5
Cache Coherency 2-5
SDRAM Controller 2-6
SDRAM Organization 2-6
SDRAM Accesses 2-6
SDRAM ECC 2-7
Refresh 2-10
Scrub 2-10
PowerPC to PCI Bridge 2-11
Data Flow Terminology 2-11
Block Diagram 2-12
Outbound Functions 2-14
PPC Decode 2-14

PPC Slave 2-16
Outbound FIFO 2-19
PCI Master 2-19
Store Gathering 2-20
Read Ahead 2-22
Passive Slave 2-23
Error Handling 2-24
Inbound Functions 2-25
PCI Decode 2-25
PCI Slave 2-30
Inbound FIFO 2-32
PPC Master 2-33
Write Posting 2-34
Read Ahead 2-35
Copyback Snarfing 2-35
Bus Hog 2-36
Error Handling 2-36
Generating PCI Cycles 2-36
FIFO Tuning 2-41
Write-Posting 2-41
Read-Ahead 2-41
Transaction Ordering 2-44
Endian Conversion 2-45
DMA Controller 2-48
Architecture 2-49
Operating Modes 2-50
Direction of Data Movement 2-52
Addressing and Transfer Sizes 2-53
Data Patterns 2-54
Linked-List Descriptors 2-56
Transfer Termination 2-57
Interrupts 2-58
Transfer Throttling 2-59
Message Passing 2-60
I2O Message Passing 2-60
IOP Message Unit 2-61
IMU Queue Structure 2-63
IMU Enable 2-64
IMU Interrupts 2-65
IOP Agent Identification 2-65
Generic Message Passing 2-66
Doorbell Registers 2-66

Message Passing Registers 2-67
Multiprocessor Interrupt Controller (MPIC) 2-68
MPIC Features 2-68
Architecture 2-68
CSR's Readability 2-69
Interrupt Source Priority 2-69
Processor's Current Task Priority 2-69
Nesting of Interrupt Events 2-70
Spurious Vector Generation 2-70
Interprocessor Interrupts (IPI) 2-70
8259 Compatibility 2-71
Harrier Internal Functional Interrupt 2-71
Harrier Internal Error Interrupt 2-71
Timers 2-72
Interrupt Delivery Modes 2-72
Block Diagram Description 2-73
Program Visible Registers 2-75
Interrupt Pending Register (IPR) 2-75
Interrupt Selector (IS) 2-75
Interrupt Request Register (IRR) 2-76
In-Service Register (ISR) 2-76
Interrupt Router 2-76
Programming Notes 2-78
External Interrupt Service 2-78
Reset State 2-79
Operation 2-80
Interprocessor Interrupts 2-80
Dynamically Changing I/O Interrupt Configuration 2-80
EOI Register 2-81
Interrupt Acknowledge Register 2-81
8259 Mode 2-81
Current Task Priority Level 2-81
I2C Interface 2-82
Byte Write 2-82
Random Read 2-85
Current Address Read 2-87
Page Write 2-89
Sequential Read 2-91
UART Interface 2-94
XPORT 2-94
PowerPC Slave 2-96

Xport Bus Master 2-96
Xport Bus Transaction Examples 2-97
Xport Bus Address Mapping 2-106
Xport Bus XAD Mapping 2-108
Hawk Compatibility Mode 2-109
Xport Bus Byte Mapping 2-110
Arbiters 2-113
PCI Arbiter 2-115
Watchdog Timers 2-120
Exceptions 2-122
Error Diagnostics 2-127
PowerPC Address Bus Timer 2-129
PowerPC Parity 2-130
Reset Signals 2-131
PPMC Features 2-132
Hardware Configuration 2-133

CHAPTER 3 Programming Model

Architectural Overview 3-1
Register Group Summary 3-3
PowerPC Control and Status (XCSR) Register Group 3-3
PowerPC Multi-Processor Interrupt Controller (XMPI) Register Group 3-14
PowerPC to PCI Configuration Space (XCFS) Register Group 3-19
PCI Configuration Space (PCFS) Register Group 3-21
PCI Message Passing (PMEP) Register Group 3-23
SDRAM Interface 3-25
SDRAM General Control Register 3-25
SDRAM Timing Control Register 3-27
SDRAM Bank (A,B,C,D,E,F,G, and H) Addressing Registers 3-30
SDRAM Scrub Control Register 3-32
SDRAM Scrub Address Counter 3-33
SDRAM Single-bit Error Status 3-33
SDRAM Single-bit Error Address Register 3-36
SDRAM Multi-bit Error Status 3-36
SDRAM Multi-bit Error Address Register 3-37
PowerPC to PCI Bridge 3-38
XCSR Register Group 3-38
Bridge PCI Control and Status Register 3-38
Bridge PowerPC Control and Status Register 3-41
PCI Interrupt Acknowledge Register 3-43

Outbound Translation Address (0, 1 and 2) Registers....3-44
Outbound Translation Offset/Translation Attribute (0, 1 and 2) Registers3-45
Outbound Translation Address (3) Register....3-47
Outbound Translation Offset/Translation Attribute (3) Registers....3-48
Passive Slave Address Registers....3-49
Passive Slave Offset/Translation Attribute Registers....3-50
XCFS Register Group....3-52
CONFIG_ADDRESS Register....3-52
CONFIG_DATA Register....3-54
PCFS Register Group....3-55
Vendor ID/Device ID Registers....3-55
Command/Status Registers....3-56
Revision ID/Class Code Registers....3-58
Cache Line Size/Master Latency Timer/Header Type Register....3-59
Message Passing Register Group Base Address Register....3-61
Inbound Translation Base Address (0, 1, 2 and 3) Registers....3-62
Subsystem Vendor ID/Subsystem ID Registers....3-65
Interrupt Line/Interrupt Pin/Minimum Grant/Maximum Latency Registers3-66
Message Passing Attribute Register....3-68
Inbound Translation Size/Offset (0, 1, 2 and 3) Registers....3-69
Inbound Translation Attribute (0, 1, 2 and 3) Registers....3-71
PCI Status Register....3-75
PCI General Purpose Register....3-76
DMA Controller....3-78
DMA Control Register....3-78
DMA Status Register....3-81
DMA Source Address Register....3-83
DMA Source Attribute Register....3-84
DMA Destination Address Register....3-86
DMA Destination Attribute Register....3-87
DMA Next Link Address Register....3-89
DMA Count Register....3-90
DMA Current Source Address Register....3-90
DMA Current Destination Address Register....3-91
DMA Current Link Address Register....3-92
Message Passing....3-93
XCSR Register Group....3-93
MP Generic Outbound Message (0 and 1) Registers....3-93
MP Generic Outbound Doorbell Register....3-94
MP Generic Inbound (0 and 1) Message Registers....3-94
MP Generic Inbound Doorbell Register....3-95

MP Generic Inbound Doorbell Mask Register 3-96
MP I20 Outbound Free_list Head Register 3-96
MP I20 Outbound Free_list Tail Register 3-97
MP I20 Outbound Post_list Head Register 3-98
MP I20 Outbound Post_list Tail Register 3-98
MP I20 Inbound Free_list Head Register 3-99
MP I20 Inbound Free_list Tail Register 3-100
MP I20 Inbound Post_list Head Register 3-100
MP I20 Inbound Post_list Tail Register 3-101
MP I20 Control Register 3-102
MP I20 Queue Base Register 3-103
PMEP Register Group 3-103
MP I20 Interrupt Status Register 3-103
MP I20 Interrupt Mask Register 3-104
MP I20 Inbound Queue Register 3-104
MP I20 Outbound Queue Register 3-105
MP Generic Outbound Message (0 and 1) Registers 3-106
MP Generic Outbound Doorbell Register 3-106
MP Generic Inbound Message (0 and 1) Registers 3-107
MP Generic Inbound Doorbell Register 3-108
MP Generic Interrupt Status Register 3-109
MP Generic Interrupt Mask Register 3-110
MP Generic Outbound Doorbell Mask Register 3-111
Multi-Processor Interrupt Controller 3-112
XCSR Register Group 3-112
MPIC Base Address Register 3-112
MPIC Control and Status/Interrupt Request Sample Registers 3-113
XMPI Register Group 3-113
Feature Reporting Register 3-114
Global Configuration Register 3-115
Vendor Identification Register 3-116
Processor Init Register 3-117
IPI Vector/Priority (0, 1, 2, and 3) Registers 3-118
Spurious Vector Register 3-119
Timer Frequency Register 3-119
Timer Current Count (0, 1, 2, and 3) Registers 3-120
Timer Base Count (0, 1, 2, and 3) Registers 3-121
Timer Vector/Priority (0, 1, 2, and 3) Registers 3-122
Timer Destination (0, 1, 2, and 3) Registers 3-123
External Source Vector/Priority (0 through 15) Registers 3-124
External Source Destination (0 through 15) Registers 3-125
Harrier Internal Functional/Error Interrupt Vector/Priority Register 3-126

Harrier Internal Functional/Error Interrupt Destination Register 3-127
Processor 0/Processor 1 IPI Dispatch (0, 1, 2, and 3) Registers 3-128
Processor 0/Processor 1 Current Task Priority Registers 3-129
Processor 0/Processor 1 Interrupt Acknowledge Registers 3-129
Processor 0/Processor 1 End-Of-Interrupt Registers 3-130
I2C Controller 3-131
I2C Clock Prescaler Register 3-131
I2C Control Register 3-132
I2C Transmitter Data Register 3-133
I2C Status Register 3-134
I2C Receiver Data Register 3-135
UART Controller 3-136
UART Core Registers 3-136
UART General Registers 3-147
Xport 3-150
Xport Address Range (0, 1, 2, 3) Registers 3-150
Xport Attributes (0, 1, 2, 3) Registers 3-151
Xport General Control Register 3-155
Arbiters 3-156
PCI Arbiter Register 3-156
PowerPC Arbiter Register 3-159
Watchdog Timers 3-161
Watchdog Timer Control Registers 3-161
Watchdog Timer Status Registers 3-163
Exceptions 3-164
Functional Exception Enable Register 3-164
Functional Exception Status Register 3-166
Functional Exception Mask Register 3-168
Functional Exception Clear Register 3-169
Error Exception Enable Register 3-170
Error Exception Status Register 3-173
Error Exception Clear Register 3-178
Error Exception Interrupt Enable Register 3-180
Error Exception Machine Check 0 Enable Register 3-183
Error Exception Machine Check 1 Enable Register 3-185
Error Diagnostics 3-187
Error Diagnostics Error Injection Register 3-188
Error Diagnostics PowerPC Address Register 3-189
Error Diagnostics PowerPC Attribute Register 3-190
Error Diagnostics PCI Address Register 3-191
Error Diagnostics PCI Attribute Register 3-192

Miscellaneous Functions 3-193
Vendor ID/Device ID Registers 3-193
Revision ID Register 3-193
Global Control and Status Register 3-194
PowerPC Clock Frequency Register 3-197
Count 32-bit Register 3-198
Miscellaneous Control and Status Register 3-199
General Purpose Registers 3-200
General Purpose Memory 3-201

CHAPTER 4 Performance

SDRAM Interface 4-1
Xport Bus Interface 4-4
Latency of Xport-bound Reads (Xport read bursting disabled) 4-5
Background Information 4-6
PowerPC to PCI Bridge 4-8
Inbound Performance 4-9

CHAPTER 5 Programming Considerations

Programming SDRAM Related Control Registers 5-1
Initializing SDRAM Related Control Registers 5-1
SDRAM Speed Attributes 5-1
SDRAM Refresh Period 5-1
SDRAM Size 5-2
I2C EEPROMs 5-2
SDRAM Base Address and Enable 5-2
SDRAM Control Registers Initialization Example 5-3
Optional Method for Sizing SDRAM 5-9
Operation without Firmware 5-13

APPENDIX A Related Documentation

Motorola Computer Group Documents A-1
Manufacturers' Documents A-2
Related Specifications A-4

List of Figures

Figure 1-1. Typical Harrier System Implementation 1-4
Figure 1-2. Harrier Block Diagram.....1-7
Figure 2-1. Block Diagram of PLL Implementation 2-1
Figure 2-2. PowerPC and PCI Clock Relationships 2-3
Figure 2-3. SDRAM Interface Block Diagram.....2-4
Figure 2-4. Harrier PowerPC/PCI Data Flow Naming Convention 2-11
Figure 2-5. PowerPC to PCI Bridge Block Diagram.....2-13
Figure 2-6. Outbound Address Decoding 2-15
Figure 2-7. Outbound Address Translation.....2-16
Figure 2-8. Inbound Address Decoding.....2-27
Figure 2-9. Inbound Address Translation 2-29
Figure 2-10. Spread I/O Address Translation 2-38
Figure 2-11. Big Endian/ Little Endian Data Swap 2-46
Figure 2-12. DMA Controller Block DIagram 2-49
Figure 2-13. DMA Controller Operating Modes 2-51
Figure 2-14. Examples of Pattern Writes.....2-55
Figure 2-15. IOP Message Unit 2-62
Figure 2-16. IMU Queue Structure.....2-64
Figure 2-17. MPIC Block Diagram 2-74
Figure 2-18. Programming Sequence for I2C Byte Write 2-84
Figure 2-19. Programming Sequence for I2C Random Read.....2-86
Figure 2-20. Programming Sequence for I2C Current Address Read 2-88
Figure 2-21. Programming Sequence for I2C Page Write 2-90
Figure 2-22. Programming Sequence for I2C Sequential Read.....2-93
Figure 2-23. Xport Block Diagram.....2-95
Figure 2-24. Xport Bus One-Beat Read Transaction.....2-98
Figure 2-25. Xport Bus Two One-beat Write Transactions 2-99
Figure 2-26. Xport Bus Two-Beat Read Transaction (No Bursting) 2-100
Figure 2-27. Xport Bus 4-beat Read Transaction with Burst Size of 4 2-101
Figure 2-28. Xport Bus 3-beat Write Transaction with Burst Size of 4.....2-102
Figure 2-29. Xport Bus, 8-beat Read Transaction with Burst Size of 4 2-103
Figure 2-30. Xport Bus One-Beat Read Transaction in Basic Mode.....2-104
Figure 2-31. Xport Bus One-Beat Write Transaction in Basic Mode.....2-104

Figure 2-32. Xport Bus One-Beat Read Transaction in Hawk Compatibility Mode 2-105
Figure 2-33. Xport Bus One-Beat Write Transaction in Hawk Compatibility Mode 2-106
Figure 2-34. Power Up Reset Timing - Timing Group 1 2-137
Figure 4-1. Timing Definitions for Table 4-1 4-3

List of Tables

Table 2-1. PowerPC/PCI Clocking Options....2-2
Table 2-2. SDRAM Single and Multi-Bit Error Reporting....2-9
Table 2-3. Map Decoder Priority....2-29
Table 2-4. PPC Master Transaction Profiles and Starting Offsets....2-34
Table 2-5. Memory and I/O Attributes....2-37
Table 2-6. Configuration Device Decode....2-40
Table 2-7. Harrier Read-Ahead Options....2-42
Table 2-8. DMA Controller Linked-List Descriptors....2-56
Table 2-9. Xport Bus Address Mapping....2-107
Table 2-10. Xport Bus XAD Mapping....2-108
Table 2-11. Hawk Compatible Address Mapping....2-109
Table 2-12. 8-bit Device Byte Lane Mapping....2-111
Table 2-13. 16-bit Device Byte Lane Mapping....2-111
Table 2-14. Hawk Data Compatibility Byte Lane Mapping....2-111
Table 2-15. 32-bit Device Byte Lane Mapping....2-112
Table 2-16. PPC Arbiter Pin Assignments....2-113
Table 2-17. PCI Arbiter Pin Description....2-115
Table 2-18. HEIR Encoding for Fixed Mode Priority....2-117
Table 2-19. HEIR Encoding for Mixed Mode Priority....2-117
Table 2-20. PRK Encoding....2-118
Table 2-21. WTxC Programming....2-121
Table 2-22. Exception Summary....2-123
Table 2-23. Error Exceptions and Address/Attribute Capture....2-128
Table 2-24. Harrier Hardware Configuration....2-133
Table 3-1. Harrier PowerPC and PCI Resources....3-1
Table 3-2. PowerPC Control and Status (XCSR) Register Group....3-4
Table 3-3. PowerPC Multi-Processor Interrupt Controller (XMPI) Register Group....3-15
Table 3-4. PowerPC to PCI Configuration Space (XCFS) Register Group....3-20
Table 3-5. PCI Configuration Space (PCFS) Register Group....3-22
Table 3-6. PCI Message Passing (PMEP) Register Group....3-24
Table 3-7. MXRR Control of Refresh Rate....3-25
Table 3-8. SDTC RWCB Example....3-26
Table 3-9. SDTC TRC Encoding....3-28

Table 3-10. SDTC TRAS Encoding 3-28
Table 3-11. SDTC TDPL Encoding 3-29
Table 3-12. SDTC TRP Encoding 3-29
Table 3-13. SDTC TRCD Encodiing 3-29
Table 3-14. SDBA SIZE Encoding 3-31
Table 3-15. SDSES.ESB Encoding 3-34
Table 3-16. Syndrome Code Ordered by Bit in Error 3-35
Table 3-17. BPCS PIM Encoding 3-40
Table 3-18. BXCS RBT Encoding 3-42
Table 3-19. BXCS SBT Encoding 3-42
Table 3-20. OTATx RXS Encoding 3-46
Table 3-21. BASE Encoding and Resource Size 3-49
Table 3-22. PSSZ Encoding 3-51
Table 3-23. CLAS Encoding 3-59
Table 3-24. BASE Encoding and Resource Size 3-64
Table 3-25. INTP INT Encoding 3-67
Table 3-26. ITSZx Encoding 3-70
Table 3-27. ITATx RXS/RMS Encoding 3-73
Table 3-28. Harrier Generated Transfer Types 3-75
Table 3-29. DCTL XTH Encoding 3-79
Table 3-30. DCTL PBT Encoding 3-80
Table 3-31. DSAT TYP Encoding 3-85
Table 3-32. DSAT PRC Encoding 3-86
Table 3-33. DDAT TYP Encoding 3-88
Table 3-34. DDAT PWC Encoding 3-88
Table 3-35. MICT QSZ Encoding 3-102
Table 3-36. Cascade Mode Encoding 3-115
Table 3-37. Tie Mode Encoding 3-116
Table 3-38. FENS1-0 Status 3-138
Table 3-39. UART Interrupt Control Functions 3-138
Table 3-40. Receiver FIFO Trigger Level 3-140
Table 3-41. WLS1-0 Encoding 3-142
Table 3-42. UART Clock Selection 3-148
Table 3-43. Baud Rates and Divisors 3-148
Table 3-44. XPATx RVENx Encoding 3-152
Table 3-45. XPATx DW Encoding 3-152
Table 3-46. XPATx AD Encoding 3-153
Table 3-47. XPATx BLE Encoding 3-153
Table 3-48. XPATx BRD Encoding 3-154

Table 3-49. XPATx BWD Encoding 3-154
Table 3-50. PARB PRI Encoding 3-156
Table 3-51. PARB PRK Encoding 3-157
Table 3-52. PARB HIE (Fixed Mode) Encoding 3-157
Table 3-53. PARB HIE (Mixed Mode) Encoding 3-158
Table 3-54. XARB FBR/FSR/FBW/FSW Encoding 3-160
Table 3-55. XARB PRK Encoding 3-160
Table 3-56. WTxC RES Encoding 3-162
Table 3-57. GCSR XBS Encoding 3-195
Table 3-58. GCSR BTO Encoding 3-196
Table 3-59. GCSR MID Encoding 3-196
Table 3-60. GCSR RAT Encoding 3-197
Table 4-1. PowerPC (60x) Bus to SDRAM Estimated Access Timing at 100 MHz with PC100 SDRAM's 4-1
Table 4-2. PowerPC (60x) Bus Performance for Xport Bus Bound Cycles 4-5
Table 4-3. Number of Xport Data Beats for Different PowerPC (60x) Transfer Sizes 4-7
Table 4-4. Outbound Performance Matrix 4-8
Table 5-1. Deriving TRAS, TRP, TRCD and TRC Control Bit Values from SPD 5-5
Table 5-2. Programming the SDRAM SIZ Bits 5-7
Table 5-3. Programming the SDRAM Refresh Period 5-8
Table 5-4. Address Lists for Different Bank Size Checks 5-10
Table A-1. Motorola Computer Group Documents A-1
Table A-2. Manufacturers' Documents A-2
Table A-3. Related Specifications A-4

About This Manual

The *Harrier ASIC Programmer's Reference Guide* provides chip level information, including register bit descriptions for the Harrier ASIC. It describes the architecture of the Harrier ASIC by providing the functional description, programming model/register descriptions, block diagrams, performance data, and programming issues. Most of the information for programming the Harrier chip within a PowerPC PMC or CompactPCI board level system are contained in this manual. This manual is intended to be used in conjunction with other Motorola Programming Guides, whose product incorporates the Harrier ASIC as PowerPC/PCI bus bridge, SDRAM interface, PCI interface, DMA controller, Message Passing device, UART and I^2C interface, interrupt controller, arbiter, and Xport device, or portions of the aforementioned functions.

This manual is intended for anyone who wants to program SBC boards with one or more Harrier ASICs in order to design OEM systems, supply additional capability to an existing compatible system, or work in a lab environment for experimental purposes. A basic knowledge of computers and digital logic is assumed.

The printed version of this document is bound in two parts:

Part 1 (ASICHRA1/PG1) contains Chapter 1 and 2, plus a Table of Contents, List of Figures and List of Tables for those chapters only. It also contains an Index for both parts.

Part 2 (ASICHRA2/PG1) contains Chapter 3, 4, and 5, and Appendix A, plus a Table of Contents, List of Figures and List of Tables for those chapters and appendix only. It also contains an Index for both parts.

The pdf version of this document is formatted in one book, so that all cross reference and other hyperlink text work correctly.

Summary of Changes

This is the initial publication of this document; consequently, there are no changes at this time.

Overview of Contents

Chapter 1, *Introduction*, provides an overview of the Harrier, its basic features, and a description of its functional blocks.

Chapter 2, *Functional Descriptions*, describes the operational characteristics of all functional components within the Harrier.

Chapter 3, *Programming Model*, provides an architectural overview, a register group summary, and a detailed description of all registers.

Chapter 4, *Performance*, provides details of several interfaces including the SDRAM interface, the Xport Bus interface, and the PowerPC to PCI Bridge.

Chapter 5, *Programming Considerations*, provides explanations for programming certain registers, as well as what the implications of those programming changes might have.

Appendix A, *Related Documentation*, provides a listing of other Motorola documents, third party documents, and industry specifications related to this product.

Manual Terminology

Throughout this manual, special character symbols are used to identify the numeric format of data and address parameters. These symbols precede the parameters and identify their numeric format as follows:

$	dollar	specifies a hexadecimal character
%	percent	specifies a binary number
&	ampersand	specifies a decimal number

For example: “12” is the decimal number twelve, and “$12” is the decimal number eighteen.

Unless otherwise specified, all address references are in hexadecimal.

An asterisk (*) following the signal name for signals which are *level significant* denotes that the signal is *true* or valid when the signal is low.

An asterisk (*) following the signal name for signals which are *edge significant* denotes that the actions initiated by that signal occur on high to low transition.

Note In some instances, an underscore character (_) following the signal name is used to indicate an active low signal.

In this manual, *assertion* and *negation* are used to specify forcing a signal to a particular state. In particular, assertion and assert refer to a signal that is active or true; negation and negate indicate a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent.

Data and address sizes for MPC60*x* chips are defined as follows:

- A *byte (BYTE)* is eight bits, numbered 0 through 7, with bit 0 being the least significant.
- A *half-word (HWORD)* is 16 bits, numbered 0 through 15, with bit 0 being the least significant.
- A *word* or *single word (WORD)* is 32 bits, numbered 0 through 31, with bit 0 being the least significant.
- A *double word (DWORD)* is 64 bits, numbered 0 through 63, with bit 0 being the least significant.

The terms *control bit* and *status bit* are used extensively in this document. The term *control bit* is used to describe a bit in a register that can be set and cleared under software control. The term *true* is used to indicate that a bit is in the state that enables the function it controls. The term *false* is used to indicate that the bit is in the state that disables the function it controls. In all tables, the terms 0 and 1 are used to describe the actual value that should be written to the bit, or the value that it yields when read. The term *status bit* is used to describe a bit in a register that reflects a specific condition. The status bit can be read by software to determine operational or exception conditions.

Comments and Suggestions

Motorola welcomes and appreciates your comments on its documentation. We want to know what you think about our manuals and how we can make them better. Mail comments to:

Motorola Computer Group
Reader Comments DW164
2900 S. Diablo Way
Tempe, Arizona 85282

You can also submit comments to the following e-mail address:
reader-comments@mcg.mot.com

In all your correspondence, please list your name, position, and company. Be sure to include the title and part number of the manual and tell how you used it. Then tell us your feelings about its strengths and weaknesses and any recommendations for improvements.

Conventions Used in This Manual

The following typographical conventions are used in this document:

bold

is used for user input that you type just as it appears; it is also used for commands, options and arguments to commands, and names of programs, directories and files.

italic

is used for names of variables to which you assign values. Italic is also used for comments in screen displays and examples, and to introduce new terms.

`courier`

is used for system output (for example, screen displays, reports), examples, and system prompts.

<**Enter**>, <**Return**> or <**CR**>

<**CR**> represents the carriage return or Enter key.

CTRL

represents the Control key. Execute control characters by pressing the Ctrl key and the letter simultaneously, for example, **Ctrl-d**.

Bus Naming

The names "PowerPC" and "60x" are used interchangeably throughout this document.

Note All references to PowerPC bus support via the Harrier ASIC are specifically related to the 60x bus mode, and is not intended to imply support of any other PowerPC bus mode.

Bit Ordering

The bit ordering convention for Harrier depends on which bus group a signal belongs. All busses or bit fields relating to the PowerPC bus use Big-Endian bit ordering (0=MSB). All remaining busses and bit fields use Little-Endian bit ordering (0=LSB).

Register and Bit Naming

A register consists of a collection of 8, 16, 24, or 32-bits. Some or all of the bits within a register are specifically defined, while the remaining bits are undefined.

A register name is designated with bold and capital lettering as follows:

RGST

A group of registers is called a Register Group. A Register Group is designated by a four letter descriptor. The first letter of the descriptor designates the addressing space that the Register Group resides in. A

Register Group residing within PowerPC address space begins with the letter "X". A Register Group that resides within PCI address space begins with the letter "P".

A Register Group is designated with italic lettering as follows:

XGRP

There are cases where a register from one Register Group may be referenced while discussing another Register Group. A register can be referenced hierarchically as follows:

***XGRP*.RGST**

There may be some cases where it is desirable to associate a bit name to a register name. In these cases, the bit name may be appended to the register name and/or register hierarchy as follows:

RGST.BIT
***XGRP*.RGST.**BIT

Register Descriptions

All register descriptions follow a fixed convention. The possible operations for each bit within a register are as follows:

R - The bit is a read only status bit.

R/W - The bit is a readable and writable.

R/C - The bit is cleared by writing a one to itself.

The possible states of the bits after local and power-up reset are as defined below.

P - The bit is affected by power-up reset (PURST_)

L - The bit is affected by local reset (RST_)

X - The bit is not affected by reset.

V - The effect of reset on the bit is variable.

Most registers can be read from or written to as 1, 2, 4, or 8 byte entities. Some registers may have special restrictions, in which case these will be discussed on an individual basis.

All blank bit fields are considered reserved and read as 0’s.

Introduction 1

This chapter provides a brief description of the Harrier ASIC, a list of features, block diagrams from a system level implementation and from a functional chip level, as well as a description of each functional block within the Harrier.

Overview

The Harrier is a multi-function ASIC that offers a single chip solution for PowerPC based processor systems. A complete system may be created using a single Harrier ASIC; however, multiple Harrier ASICs may reside on the same PowerPC bus to provide additional I/O and peripheral support capabilities. Harrier supports a maximum of four PowerPC bus masters. Figure 1-1 on page 1-4 shows a typical single Harrier, dual processor system implementation.

Features

- PowerPC 60x Bus Interface
  - Optimized for 100 MHz operation
  - Address and data bus parity
  - Supports PowerPC bus pipelining
- SDRAM Interface
  - X-1-1-1 cycle time for all burst accesses to SDRAM
  - Double-Bit-Error detect, Single-Bit-Error correct across 72 bits
  - 8 banks with up to 256MB
  - Uses -8, -10, or PC100 SDRAMs
  - Built-in Refresh/Scrub
  - Error Notification

    Software Programmable Interrupt on Single/Double-Bit Error

Error address and Syndrome Log Registers for Error Logging

- PCI Interface
  - Fully compliant with *PCI Local Bus Specification Revision 2.1*
  - 32-bit addressing, 32-bit or 64-bit data
  - 33 MHz and 66 MHz operation
  - Supports Memory, I/O, and Configuration addressing space
  - Multi-level write posting buffers for writes to either PowerPC or PCI bus
  - Read-ahead buffer for reads from either PowerPC or PCI bus
  - Eight independent software programmable address translation map decoders
  - Store-Gathering
  - Copy-Back Snarfing
- DMA Controller
  - Single Channel
  - Direct or Linked-List
  - PowerPC to PCI, PCI to PowerPC, PowerPC to PowerPC, and PCI to PCI
  - Fixed or incrementing pattern to PowerPC or PCI
- Message Passing
  - Fully compliant with *Intelligent I/O (I_2O) Architecture Specification Version 1.5*
  - IOP Message Unit
  - Generic functions
- 2 UART interfaces
- 2 I^2C bus *master* interfaces
- Interrupt Controller
  - MPIC compliant

  - All control registers are directly accessible from the PowerPC bus
  - Supports 16 external interrupt sources and two processors
  - Multiprocessor interrupt control allows any interrupt source to be directed to either processor
  - Multilevel cross processor interrupt control for multiprocessor synchronization
  - Four 31-bit tick timers
- Arbitration
  - Internal or external PowerPC and PCI bus arbitration
- Xport
  - "Static RAM" style control, address and data signals
  - 4 individually programmable channels
  - Programmable access time
  - Programmable data widths of 8, 16 or 32 bits
  - Intended devices include Flash, ROM, External Control Registers and FIFO’s.
- Supports synchronous PowerPC/PCI clock ratios of 5:2, 3:2, 3:1, 2:1, and 1:1
- Voltage
  - 2.5V power supply for I/O’s and internal core
  - 3.3V power supply for I/O’s
- 720-pin flip-chip package

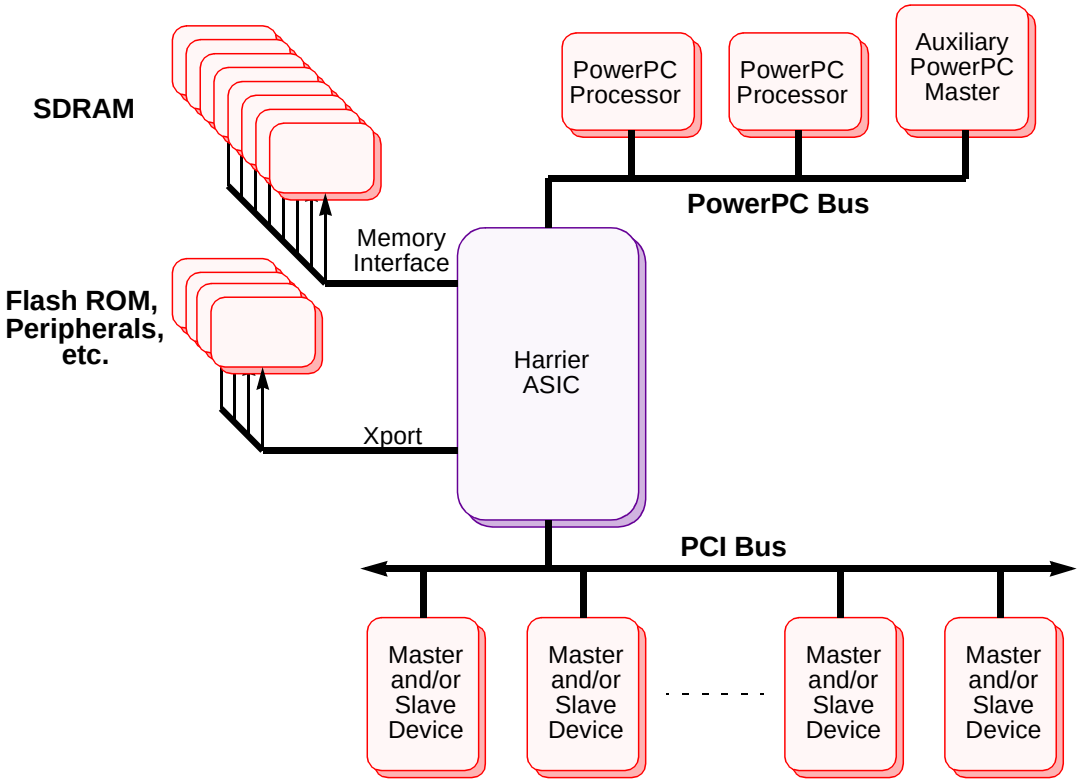


Figure 1-1. Typical Harrier System Implementation

Functional Blocks of Harrier

The following paragraphs describe the major functional blocks contained in the Harrier ASIC. Figure 1-2 is a block diagram of these major elements.

SDRAM Interface: This functional block is a high performance PowerPC to SDRAM memory interface. It provides PowerPC masters with exceptional bandwidth and low latency connections to system memory.

PowerPC to PCI Bridge: This functional block provides an interface between the PowerPC bus and a 33 MHz/66 MHz PCI bus. It supports either 32-bit or 64-bit data widths. It has deep write-posting buffers and advanced read-ahead algorithms to ensure a very high bandwidth interface between the PowerPC bus and the PCI bus.

DMA Controller: A highly flexible DMA controller allows fast and efficient block data transfers between any domain, including PCI to PCI, PCI to PowerPC, PowerPC to PCI and PowerPC to PowerPC. An option exists that allows complex patterns to be written to PCI or PowerPC spaces.

I_2O and Generic Message Passing: This functional block offers a fully I_2O compliant Message Passing Unit, allowing the Harrier to participate in I_2O Message Passing as either a Host or an I/O Processor (IOP). Additional hardware is provided that allow the use of more generic forms of message passing.

Multi-Processor Interrupt Controller: This functional block is a dual processor version of the OPIC interrupt controller that supports 16 external interrupts, 4 internal timer interrupts (with timers), and one internal source interrupt. The control and status resources for this function are offered directly to the PowerPC bus.

I^2C Controller: This functional block offers two standard *master-only* I^2C interfaces providing flexibility in chip to chip communication.

UART: This functional block offers two full duplex Universal Asynchronous Receiver/Transmitter (UART) interfaces that support communication with modems or other serial peripheral devices.

Xport: A multi-purpose four channel expansion port that can be used to connect Flash, ROM or various peripheral devices to the Harrier ASIC.

Arbiters: This functional block contains a PowerPC and a PCI arbiter. Each arbiter offers a wide range of programmable options, allowing the system designer to fine-tune system configurations for the maximum level of performance. If desired, external arbitration can be selected for either, or both, arbiters.

Watchdog Timers: This functional block offers two completely independent watchdog timers. The use of these timers is optional. These timers are designed to increase the integrity of software execution. An output pin is associated with each timer, allowing the timer to create virtually any system exception such as interrupt, reset, machine check, etc.

Error Diagnostics: This functional block contains support logic that helps capture information related to various system errors. Programmable options exist that enable the system error to assert various forms of interrupt exceptions.

PowerPC Address Bus Timer: This is a timer function that makes it possible for software to recover after an erroneous PowerPC access. It contains PowerPC bus tracking logic that detects non-responsive address tenures. Once detected, the address and data tenures are automatically closed and an exception is generated.

PowerPC Parity: This functional block can generate address and data parity on the PowerPC bus. Upon detection of either an address or data parity error, the Harrier can optionally generate an exception. Additional hardware allows for the intention injection of address and/or data parity errors, allowing further testing of the party error exception path.

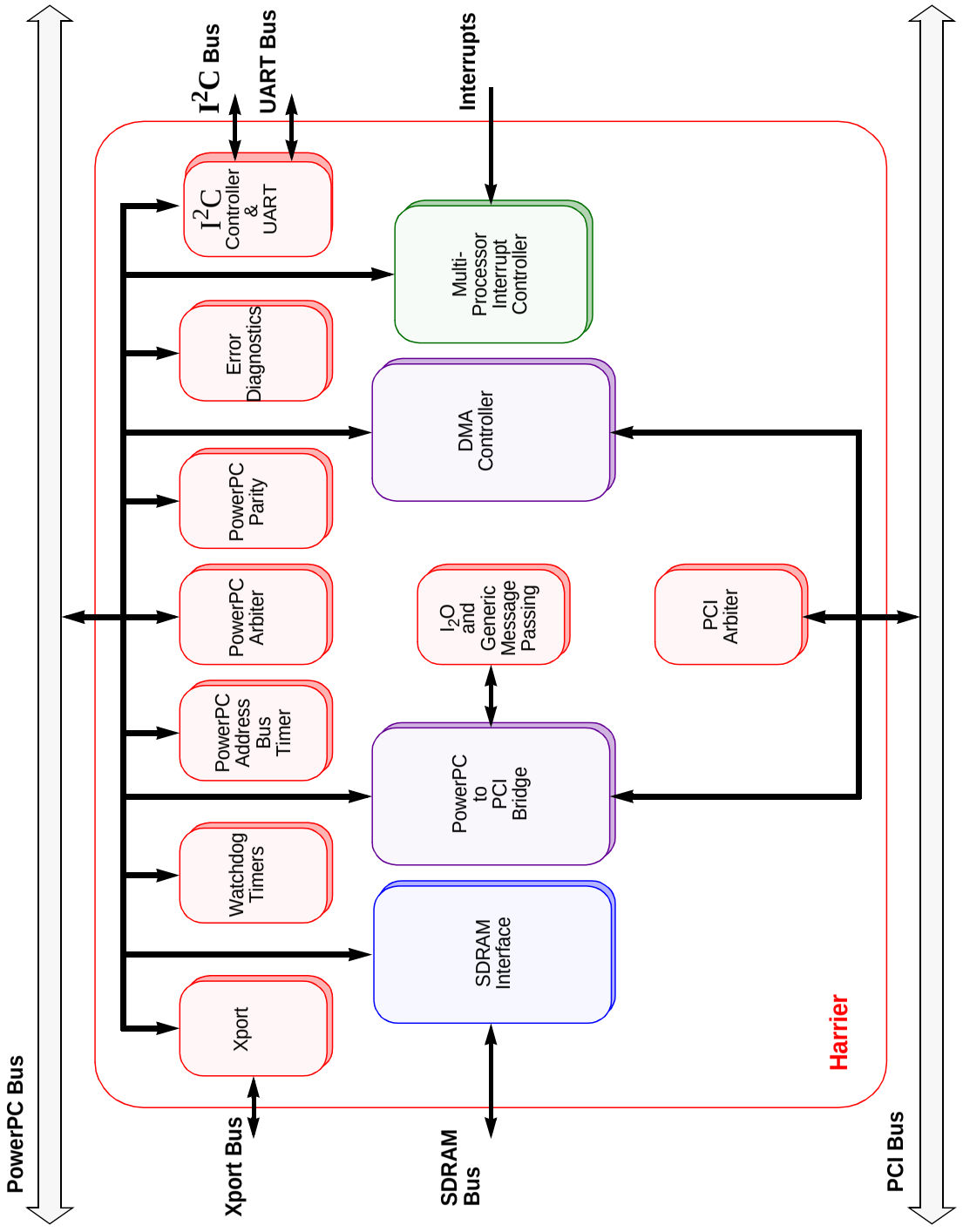


Figure 1-2. Harrier Block Diagram

Functional Descriptions

2

This chapter describes in detail the functions of the Harrier ASIC as listed in Chapter 1.

Clocking

The Harrier uses a fully integrated (internal loop filter) version of the PLL macrocell with an operating frequency between 400 MHz and 800 MHz. The ratio of the PLL's operating frequency to its reference signal frequency is 8:1 and allows the Harrier to support synchronous PowerPC-to-PCI clock ratios of 1:1, 2:1, 3:1, 3:2, and 5:2. Figure 2-1 shows the block diagram of the Harrier's PLL implementation. A sample of clocking options and their associated PowerPC and PCI frequencies is shown in Table 2-1.

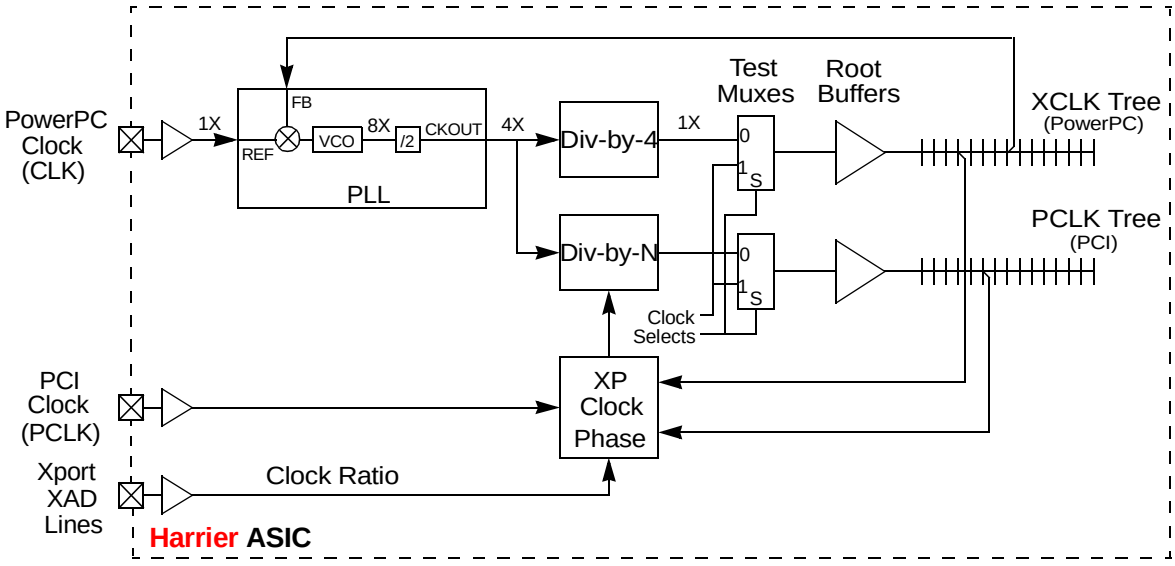


Figure 2-1. Block Diagram of PLL Implementation

Table 2-1. PowerPC/PCI Clocking Options

PowerPC Clock Frequency (MHz)	Ratio (PowerPC:PCI)	PCI Clock Divisor (N)	PCI Clock Frequency (MHz)
66.67	1:1	4	66.67
	2:1	8	33.33
	3:2	6	44.44
75.00	3:2	6	50.00
	5:2	10	30.00
83.33	3:2	6	55.55
	5:2	10	33.33
100.00	3:1	12	33.33
	3:2	6	66.67

Harrier's PLL will phase lock the external PowerPC clock to the internal XCLK tree. This phase locking creates a zero clock insertion delay on the XCLK tree. The clock insertion delay of the PCLK tree is balanced to that of the XCLK tree, which effectively makes the internal PCLK phase locked to the external PowerPC clock. The system is responsible for maintaining a fully synchronous and minimum skew relationship between the external PowerPC and PCI clocks.

The PLL ensures a synchronous relationship exists between the four clock domains, however there must be additional logic to make sure the phase cycle of the internal PCLK tree tracks that of the external PCI clock. Shortly after the PowerPC clock starts running, Harrier's internal PCLK tree starts running in a completely random (but synchronous) fashion with respect to the external PCI clock. The Harrier then periodically samples the external PCI clock and adjusts the phase cycle of the internal PCLK tree until a match is met between the phase cycles of the external PCI clock and the internal PCLK tree. In order to properly lock to the external PCI clock phase cycle, the system must hold reset asserted and the PCI clock must be running while the PLL is attempting to lock to the PowerPC clock. The

PLL takes up to 100 µs to lock to the PowerPC clock. If reset is asserted some time after the PLL has locked to the PowerPC clock then the system must hold reset active for a minimum of 500 PowerPC clock periods.

The following figure shows the synchronous relationships that must exist between the PowerPC and PCI clocks.

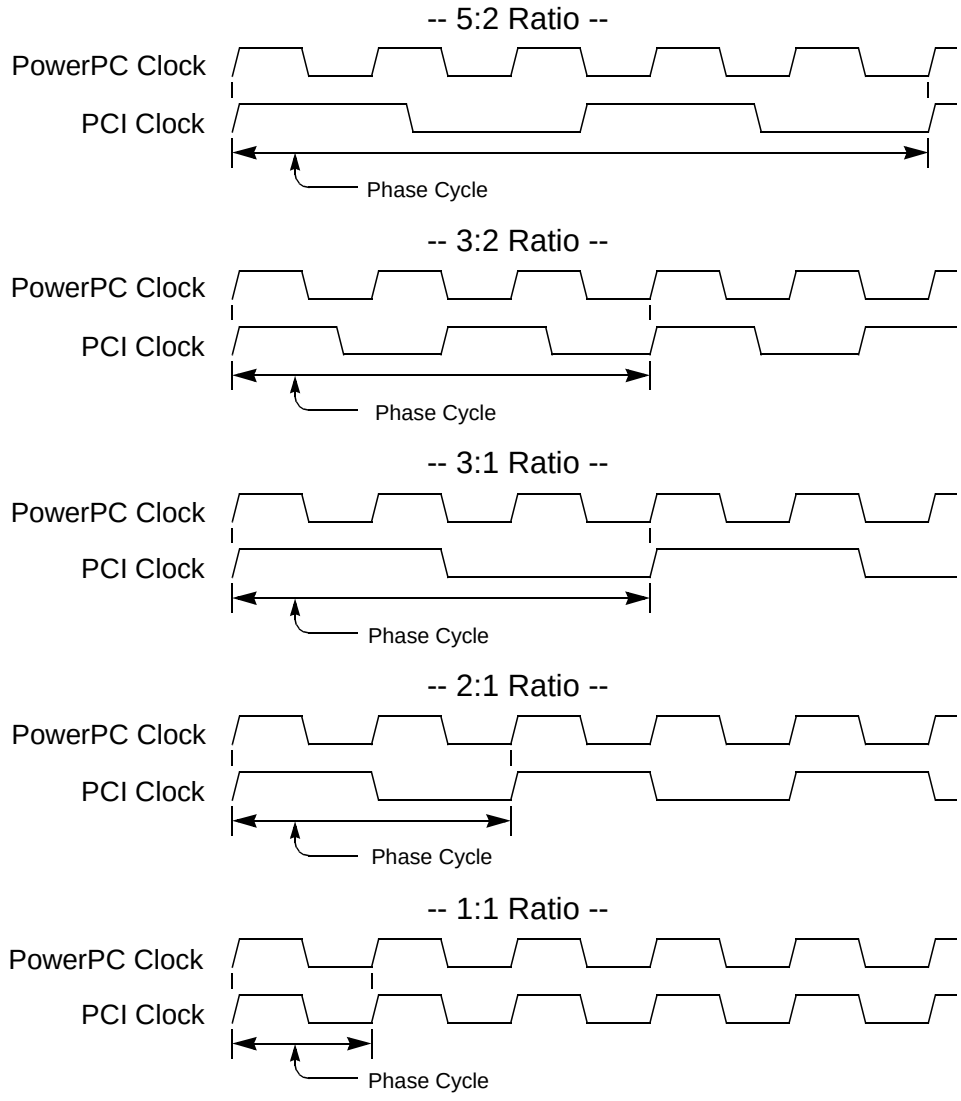


Figure 2-2. PowerPC and PCI Clock Relationships

The Harrier must be told what system clock ratio is being used so that the internal clock phasing logic knows how to lock the internal PCLK to the external PCI clock. This information is given to the Harrier in the form of a state placed on three XAD lines at the release of power-up reset. Please consult the section titled *Hardware Configuration* on page 2-133 for more information.

SDRAM Interface

The SDRAM interface provides PowerPC bus masters with high performance access to 8 banks of ECC SDRAM. The interface's major blocks consist of a PowerPC (60x) slave and an SDRAM controller. The following figure shows a simplified block diagram.

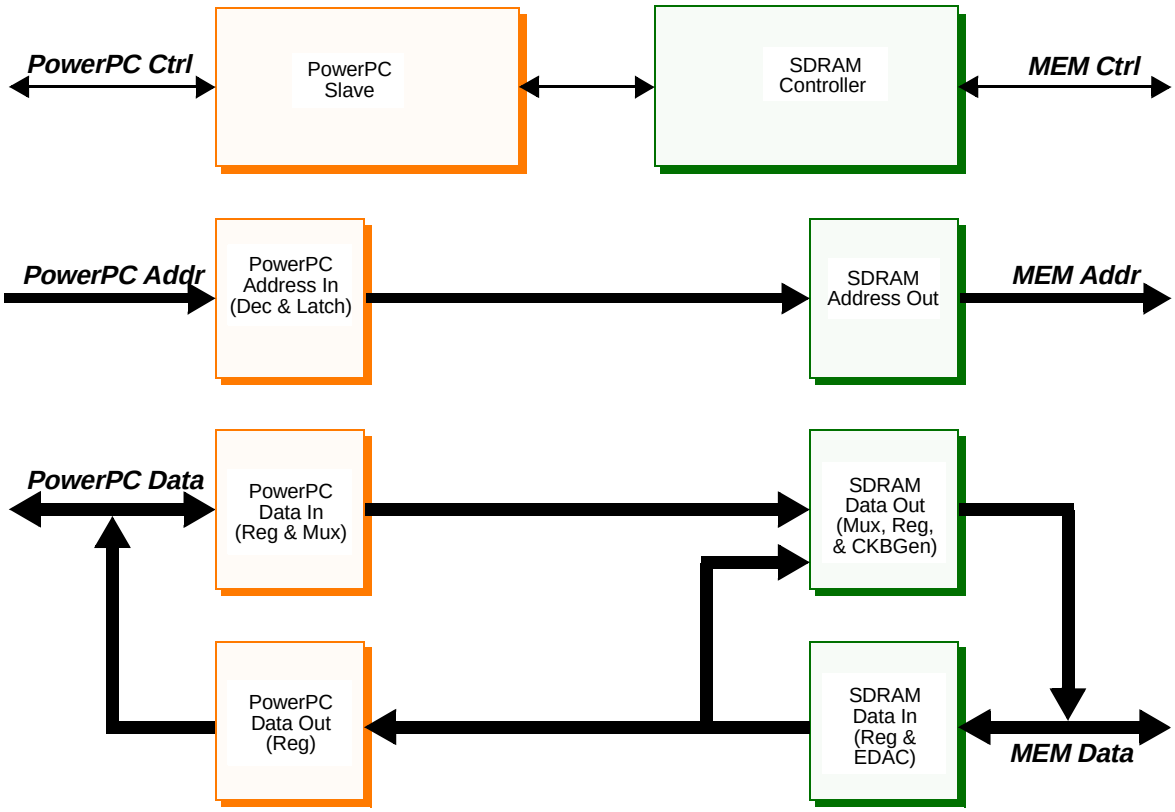


Figure 2-3. SDRAM Interface Block Diagram

Many of the SDRAM interface's functions are software programmable. For a description of the related control and status registers see the section titled *SDRAM Interface* on page 3-25.

The following sections describe the SDRAM interface's major blocks and functions.

PowerPC Bus Slave

The PowerPC bus slave works closely with the SDRAM controller to provide low latency and high throughput access to SDRAM. The slave responds to all transfer sizes and responds to most transfer types.

Responding to Address Transfers

When the PowerPC slave responds to an address transfer, it asserts AACK_ immediately if no previous data tenure is in process. If a data transfer is in process, the slave waits and asserts AACK_ when the transfer completes.

Completing Data Transfers

If an address transfer has an associated data transfer, the SDRAM controller starts an access to SDRAM as soon as it completes all previous SDRAM activity. If the data transfer is a read, the PowerPC slave starts transfering data to the PowerPC bus as soon as the SDRAM controller has data ready and the PowerPC data bus is available. If the data transfer is a write, the PowerPC slave starts latching data from the PowerPC bus as soon as any previously latched data is no longer needed and the PowerPC data bus is available.

Cache Coherency

The PowerPC slave supports cache coherency to SDRAM by monitoring and responding to the ARTRY_ signal. When ARTRY_ asserts, if the access is an SDRAM read, the PowerPC slave does not source the data for that access. If the access is an SDRAM write, SDRAM controller does not write the data for that access. Depending upon when the retry occurs, the controller may cycle the SDRAM but it will not transfer data.

2

SDRAM Controller

SDRAM Organization

SDRAM is organized as 1, 2, 3, 4, 5, 6, 7 or 8 banks, 72 bits wide with 64 of the bits being normal data and the other 8 being checkbits. The 72 bits of SDRAM for each bank can be made up of x4, x8, or x16 components or of 72-bit DIMMs that are made up of x4 or x8 components. 72-bit, unbuffered DIMMs can be used as long as AC timing is met and they use the components listed. All components must be organized with 4 internal banks.

SDRAM Accesses

Four-beat Reads/Writes

Because of the burst nature of SDRAM, the SDRAM interface performs best when responding to PowerPC burst (four-beat) accesses.

When a PowerPC master begins a burst read to SDRAM, the SDRAM controller begins an access. When the access time is reached, the SDRAM provides all four beats of data, one on each clock. Consequently, the PowerPC slave can provide the four beats of data with zero idle clocks between each beat.

When a PowerPC master begins a burst write to SDRAM, as soon as the PowerPC data transfer begins, the PowerPC slave latches and acknowledges the PowerPC data so that the PowerPC master is freed to continue with new accesses. The SDRAM controller then performs an SDRAM access to write the latched data.

Single-beat Reads/Writes

Because of start-up, addressing, and completion overhead, single-beat accesses to and from the PowerPC bus do not achieve data rates as high as do four-beat accesses. Single-beat writes are the slowest because they require that the controller perform a read followed by a write to the SDRAM in order to complete. Single-beat accesses can be held to a minimum by using the data cache in copyback mode.

Address Pipelining

The SDRAM interface takes advantage of the fact that PowerPC processors can do address pipelining. Many times while a data transfer is finishing, the PowerPC master begins a new address transfer. The SDRAM controller can begin the next SDRAM access earlier when this happens, thus increasing throughput.

Holding Open Pages

Further savings come when the new address is close enough to a previous one that it falls within an open page in the SDRAM array. When this happens, the controller can transfer the data for the next cycle without having to wait to activate a new page in SDRAM.

SDRAM Speeds and PowerPC Access Times

The SDRAM that the Harrier controls uses the PowerPC clock. The SDRAM interface accommodates operation at several different PowerPC clock frequencies using SDRAMs that have various speed characteristics. Refer to the section titled *SDRAM Timing Control Register* on page 3-27 for related programming information.

Performance related information may be found within the section titled *SDRAM Interface* on page 4-1.

SDRAM ECC

The SDRAM controller uses single-bit error correction and double-bit error detection across 64 bits of data using 8 check bits.

Cycle Types

To support ECC, the controller always deals with SDRAM using full width (72-bit) accesses. When the PowerPC bus master requests any-size read of SDRAM, the controller reads the full width at least once. When the PowerPC bus master requests a four-beat write to SDRAM, the controller writes all 72 bits 4 times. When the PowerPC bus master requests a single-beat write to SDRAM, the SDRAM controller performs a full width read cycle to SDRAM, merges in the appropriate PowerPC bus write data, and writes the full width back to SDRAM.

Error Reporting

The Harrier checks data from SDRAM during PowerPC single- and four-beat reads, during single-beat writes, and during scrubs. There is no data from SDRAM to check during burst writes. The following table shows the actions Harrier takes for the different errors that can occur during a PowerPC access.

Table 2-2. SDRAM Single and Multi-Bit Error Reporting

Error Type	Single-Beat/Four-Beat Read	Single-Beat Write	Four-Beat Write	Scrub
Single-Bit Error	Terminate the PowerPC bus cycle normally. Provide corrected data to the PowerPC bus master. Assert Interrupt or Machine Check if so enabled.	Terminate the PowerPC bus cycle normally. Correct the data read from SDRAM, merge with the write data, and write the corrected, merged data to SDRAM Assert Interrupt or Machine Check if so enabled.	N/A	This cycle is not seen on the PowerPC bus. Write corrected data back to SDRAM if so enabled. Assert Interrupt or Machine if so enabled.
Double-Bit Error	Terminate the PowerPC bus cycle normally. Provide miss-corrected, raw SDRAM data to the PowerPC bus master. Assert Interrupt or Machine Check if so enabled.	Terminate the PowerPC bus cycle normally. Do not perform the write portion of the read-modify-write cycle to SDRAM. Assert Interrupt or Machine Check if so enabled.	N/A	This cycle is not seen on the PowerPC bus. Do not perform the write portion of the read-modify-write cycle to SDRAM. Assert Interrupt or machine check if so enabled.
Triple- (or greater) Bit Error	Some of these errors are detected correctly and are treated the same as double-bit errors. The rest could show up as “no error” or “single-bit error”, both of which are incorrect.			

Error Logging

Harrier logs single and double-bit SDRAM errors. When an error occurs, Harrier records the address and syndrome bits associated with the data in error. The section titled *SDRAM Single-bit Error Status* on page 3-33 describes how the error logging control and status bits operate.

Harrier can be programmed to generate interrupts/machine checks when it logs SDRAM errors. See the section titled *Exceptions* on page 3-164 for more details.

Refresh

The Harrier performs refresh by doing a burst of 4 CBR refresh cycles to all SDRAM banks once every “4-Row Refresh Interval” (Refer to the section titled *SDRAM General Control Register* on page 3-25 for more information).

Scrub

Once every so many refresh bursts, Harrier replaces the refresh burst with a scrub cycle. The scrub cycle involves an 8-byte read followed conditionally by an 8-byte write. The 8-byte write occurs only if the read detects a single-bit error and ***GCSR*.SDSC**.SCWE is set. Otherwise, the 8-byte write does not occur.

***GCSR*.SDSC**.SCPA controls the frequency with which refresh bursts are replaced with scrubs. Refer to the section titled *SDRAM Scrub Control Register* on page 3-32 for additional information.

If so enabled, Harrier logs single and/or double-bit errors. Logged scrub errors can be enabled as error exceptions. Refer to the section titled *Error Exception Enable Register* on page 3-170 for more information.

PowerPC to PCI Bridge

The PowerPC to PCI Bridge contains the data paths and control logic that allows multiple PowerPC processors to interface with a 32/64-bit PCI Local Bus.

Data Flow Terminology

The following figure is a high level view that demonstrates the difference between the naming conventions of Outbound and Inbound traffic on a board using the Harrier ASIC. Review the figure in conjunction with the text in the following paragraph.

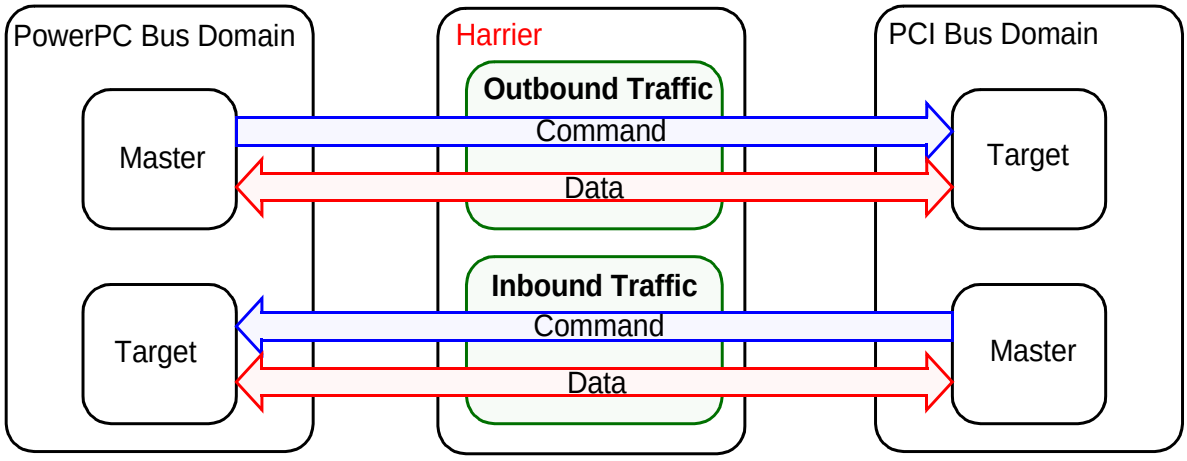


Figure 2-4. Harrier PowerPC/PCI Data Flow Naming Convention

The *Outbound* and *Inbound* naming convention relates to the direction of command information with respect to the PowerPC bus. A transaction that originates from the PowerPC bus and is bound for the PCI bus is considered *Outbound* traffic. Conversely, a transaction that originates from the PCI bus and is bound for the PowerPC bus is considered *Inbound* traffic.

2

Block Diagram

A functional block diagram of Harrier's PowerPC to PCI Bridge is shown in Figure 2-5 on page 2-13.

The control logic is subdivided into the following functions: PPC Master, PPC Slave, PCI Master, and PCI Slave. The data path logic is subdivided into the following functions: Outbound FIFO, Inbound FIFO, PPC Input, PCI Input, PPC Output, and PCI Output. Address decoding is handled in the PPC Decode and PCI Decode blocks. The control register logic is contained in the PPC Registers and PCI Registers blocks. The clock phasing and reset control logic is contained within the PPC/PCI Clock block

The FIFO structure has been designed to allow independent data transfer operations to occur between inbound and outbound transactions. The Outbound FIFO is used to support outbound transactions, while the Inbound FIFO is used to support inbound transactions. Both the Outbound FIFO and the Inbound FIFO support a command path, a read data path and a write data path. The split between read and write data paths allow Harrier to accept posted write transactions while servicing delayed read transactions. The data paths also include logic to handle the PowerPC/PCI Endian function.

All outbound transactions use the PPC Slave and PCI Master functions for maintaining bus tracking and control. During both write and read transactions, the PPC Slave places command information into the Outbound FIFO. The PCI Master draws this command information from the Outbound FIFO when it is ready to process the transaction. During write transactions, write data is captured from the PowerPC bus within the PPC Input block. This data is fed into the write data path of the Outbound FIFO.The PCI Output block removes the data from the FIFO and presents it to the PCI bus. During read transactions, read data is captured from the PCI bus within the PCI Input block. From there, the data is fed into the read data path of the Outbound FIFO. The PPC Output block removes the data from the FIFO and presents it to the PowerPC bus.

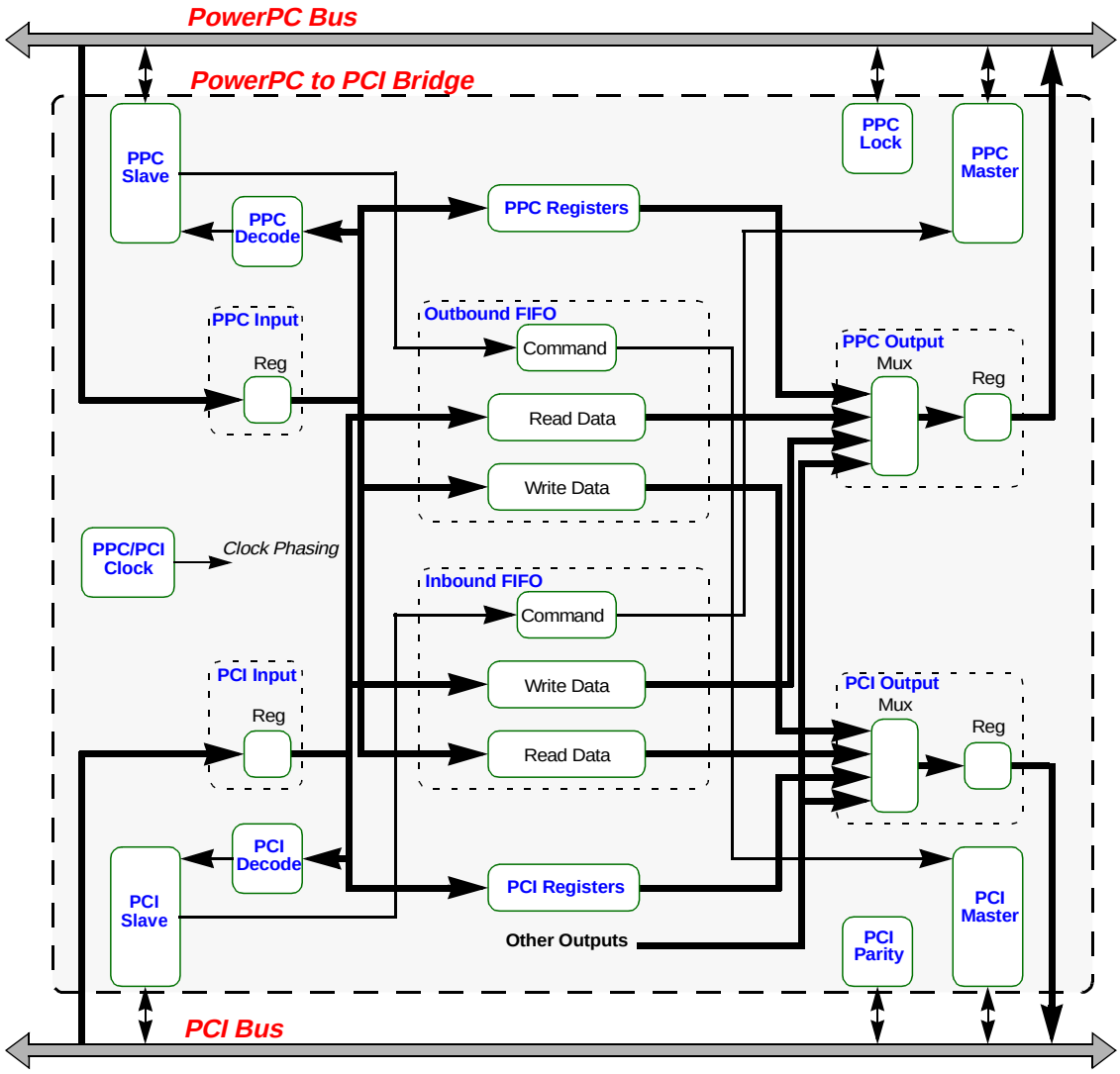


Figure 2-5. PowerPC to PCI Bridge Block Diagram

All inbound transactions use the PCI Slave and PPC Master functions for maintaining bus tracking and control. During both write and read transactions, the PCI Slave places command information into the Inbound FIFO. The PPC Master draws this command information from the Inbound FIFO when it is ready to process the transaction. During write transactions, write data is captured from the PCI bus within the PCI Input block. This data is fed into the write data path of the Inbound FIFO. The PPC Output

block removes the data from the FIFO and presents it to the PowerPC bus. During read transactions, read data is captured from the PowerPC bus within the PPC Input block. From there, the data is fed into the read data path of the Inbound FIFO. The PCI Output block removes the data from the FIFO and presents it to the PCI bus.

Cache line locking (via PCI Lock) is handled by the PPC Lock and PCI Slave blocks. The PCI Slave accommodates the PCI bus portion of lock, and the PPC Lock accommodates the PowerPC bus portion of lock.

Parity checking and generation for the PCI bus is handled by the PCI Parity block. Parity checking and generation for the PowerPC bus is handled outside of the PowerPC to PCI Bridge.

Outbound Functions

An outbound transaction is originated from the PowerPC bus and is targeted to the PCI bus. The key functional elements are the PPC Slave, the Outbound FIFO, and the PCI Master. This section describes in detail the elements of the Harrier that are associated with outbound transactions.

PPC Decode

Harrier maps either PCI Memory space or PCI I/O space into PowerPC address space using four programmable map decoders. Each map decoder is accompanied by some address translation logic, and is collectively referred to as an Outbound Translation Function.

The most significant 16 bits of the PowerPC address are compared with the address range of each Outbound Translation Function, and if the address falls within the specified range, the access is passed on to PCI. An example of this is shown in the following figure.

.

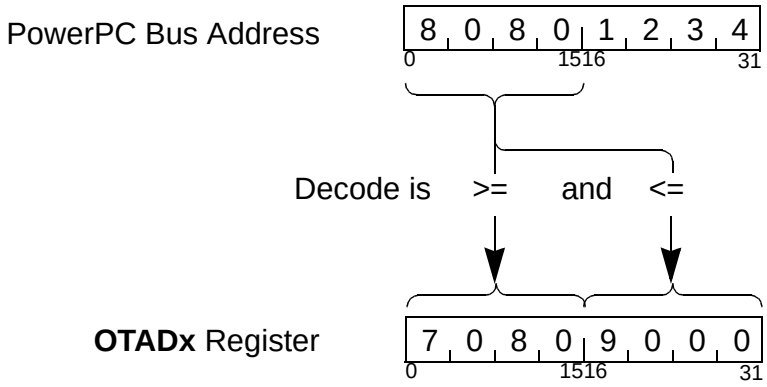


Figure 2-6. Outbound Address Decoding

There are no limits imposed by the Harrier on how large of an address space an Outbound Translation Function can represent. There is a lower limit of a minimum of 64 KBytes due to the resolution of the address compare logic.

Each Outbound Translation Function has an associated set of attributes. These attributes are used to enable the map decoder, write-posting, store-gather and read-ahead, and to define the PCI transfer characteristics.

Each Outbound Translation Function also includes a programmable 16-bit address offset. The offset is added to the 16 most significant bits of the PowerPC address, and the result is used as the PCI address. This offset provides a high degree of decoupling between PCI address space and PowerPC address space. An example of this is shown in the following figure.

.

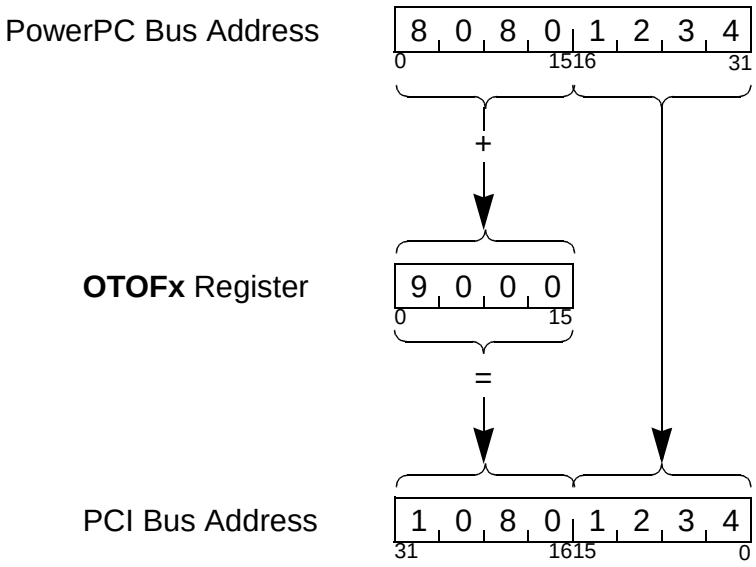


Figure 2-7. Outbound Address Translation

Care should be taken to assure that all functions decode unique address ranges since overlapping address ranges will lead to undefined operation.

PPC Slave

The PPC Slave provides the interface between the PowerPC bus and the Outbound FIFO. The PPC Slave is responsible for tracking and maintaining coherency to the PowerPC bus protocol.

The PPC Slave accepts three basic types of transactions:

1. Posted Write: Can either be in the form of a single beat or a burst transaction. May optionally be part of Store Gathering.
2. Compelled Write: Can only be single beat transaction without Store Gathering. Always performed as a *delayed transaction*.
3. Compelled Read: Can either be in the form of a single beat or a burst transaction. May optionally be part of Read Ahead. Always performed as a *delayed transaction*.

The PPC Slave supports a single level *delayed transaction* protocol. A delayed transaction is a sequence of events used to process large latency transactions that would otherwise consume a large amount of PowerPC bus bandwidth. In general, all compelled transactions will be handled by the Harrier as a delayed transaction. A delayed transaction is processed in the following manner:

1. A PowerPC master issues a bus cycle with the intention of accessing a compelled resource hosted by the Harrier. The Harrier will either accept or reject the transaction, and in either case the PowerPC bus cycle will be terminated with ARTRY_ asserted.

2. The Harrier only accepts a new transaction if the Outbound FIFO is in a ‘favorable state’. A ‘favorable state’ is when:

   a. There are no entries for a compelled transaction within the Command FIFO, and

   b. There is enough room in the Command FIFO (and the Write Data FIFO if servicing a write) to accept the current transaction.

   Note that if the Harrier is in the process of serving a previously accepted delayed transaction, the associated entry within the Command FIFO will violate rule (a) above and the Outbound FIFO will not be in a ‘favorable state’.

   If conditions are favorable, the Harrier accepts the transaction and a new entry is posted within the Command FIFO. If the transaction is a write, then write data is accepted from the data bus and posted in the Write Data FIFO. The PPC Slave records which PowerPC master originated the transaction and retains a copy of the address and attributes of the transaction for future reference. The PowerPC bus cycle is terminated with ARTRY_.

   If conditions are not favorable, then the Harrier does not accept the transaction, a new entry is not posted, and data will not be accepted.

3. The Harrier continues to service the delayed transaction to completion. While doing so, the PowerPC master continuously re-

attempts the transaction, and the Harrier continuously terminates the PowerPC bus cycles with ARTRY_ asserted.

4. The PCI Master is primarily responsible for completing the delayed transaction. If the transaction is a compelled write, then the PCI Master attempts to empty the contents of the Outbound FIFO to the PCI bus. If the transaction is a read, then the Outbound FIFO is filled from the PCI bus.

5. The PPC Slave is notified by the PCI Master upon completion of the transaction, at which time the PPC Slave is ready to close out the delayed transaction. The PPC Slave waits for the PowerPC master to once again retry the transaction. The PPC Slave validates the address, attributes, and ownership of the current PowerPC bus cycle with those of previously accepted delayed transaction bus cycle. If there is a match, then the PPC Slave will complete both the address tenure and the data tenure. At this point, a delayed transaction is considered complete.

The PPC Slave cannot perform compelled burst write transactions. The PowerPC bus protocol mandates that the qualified retry window must occur no later than the assertion of the first TA_ of a burst transaction. If the Harrier were to attempt a compelled linkage for all beats within a burst write, there is a possibility that the transaction could be interrupted. The interruption would occur at a time past the latest qualified retry window and the PPC Slave would be unable to retry the transaction. Therefore, all burst write transactions are posted regardless of the write-posting attribute within the associated map decoder register.

The PPC Slave can accept posted writes to PCI when there is a delayed transaction within the Command FIFO. The PPC Slave mandates that the delayed transaction within the Command FIFO must be a compelled read cycle and there is enough room in the Command FIFO and Write Data FIFO to accept the current write.

Outbound FIFO

The Outbound FIFO is used to hold data between the PPC Slave and the PCI Master to ensure that optimum data throughput is maintained. The Outbound FIFO consists of three major components; the command path, the read data path, and the write data path.

The command path incorporates a 50-bit by 8 entry FIFO that is used to hold command information being passed between the PPC Slave and the PCI Master. If write-posting has been enabled, then up to eight single beat, burst, or store-gathered transactions may be posted. If this limit is exceeded then any pending PowerPC transactions are retried until the PCI Master has completed a portion of the previously posted transactions and created some room within the command FIFO.

Each data path uses a 256 byte (32 entries/8 cache lines by 64-bit) FIFO. The operation of the write data path is completely independent of the read data path. This allows the Harrier to accept write-posted transactions while servicing a delayed read transaction. If a read data path FIFO limit is reached, then the PCI Master stops prefetching until the PPC Slave has emptied the FIFO beyond a certain programmable threshold. If a write data path FIFO limit is reached, then the PPC Slave continually issues retries until the PCI Master has managed to empty some portion of the FIFO.

The Harrier does not support byte merging or byte collapsing. Each and every single beat byte transaction presented to the PPC Slave will be presented to the PCI bus as a unique single beat transfer. Store-Gathering is supported for word operand transfers. See *Store Gathering* on page 2-20 for more information.

The Harrier does not snoop the command FIFO. Use caution when using Store-Gathering, write-posting or read-ahead within coherent address space.

PCI Master

The PCI Master, in conjunction with the capabilities of the PPC Slave, generally attempts to move data in either single beat or four beat transactions. If Store-Gathering is not enabled, the PCI Master supports 32-bit and 64-bit transactions in the following manner:

- All PowerPC single beat transactions, regardless of the byte count, are subdivided into one or two 32-bit transfers, depending on the alignment and size of the transaction. This includes single beat 8-byte transactions.
- All PowerPC burst transactions are transferred in 64-bit mode if the PCI bus has 64-bit mode enabled. If at any time during the transaction the PCI target indicates it can not support 64-bit mode, the PCI Master continues to transfer the remaining data within that transaction in 32-bit mode.

If Store-Gathering is enabled, the PCI Master supports 32-bit and 64-bit transactions in the following manner:

- Once a Store-Gather collection is to be transferred, the PCI Master attempts to transfer the entire collection as a single contiguous burst in 64-bit mode if the PCI bus has 64-bit mode enabled. If at any time during the transaction the PCI target indicates it can not support 64-bit mode, the PCI Master continues to transfer the remaining data within the collection in 32-bit mode.

The PCI Master can support Critical Word First (CWF) burst transfers. The PCI Master divides this transaction into two parts. The first part starts on the address presented with the CWF transfer request and continues up to the end of the current cache line. The second transfer starts at the beginning of the associated cache line and works its way up to (but not including) the dword addressed by the CWF request.

Even though the PCI Master can support burst transactions, a majority of the transaction types handled are single beat transfers. Typically PCI space is not configured as cache-able, therefore burst transactions to PCI space would not naturally occur. It must be supported since it is conceivable that bursting could happen. For example, nothing prevents the processor from loading up a cache line with PCI write data and manually flushing the cache line.

Store Gathering

The data transfer rate of outbound traffic is inherently slow due to the inability of the processor to perform anything but single beat bus cycles when moving data. The Harrier has an optional Store-Gathering mode that

condenses multiple contiguous single beat outbound transactions into large PCI burst transactions. This option may be individually enabled for each Outbound Translation Function from within the **OTATx** register.

The PPC Slave only attempts to perform store-gathering on word (32-bit), dword (64-bit) and cache line operands. The PPC Slave continues to collect words indefinitely until either a forced flush condition occurs or the Outbound FIFO approaches the full state. If forced to flush or unload the FIFO, the PPC Slave closes the current store-gathering collection. The PCI Master then immediately attempts to move the remainder of the collection to the PCI bus.

There are two groups of events that can force a collection flush; Mandatory Flush Events and Optional Flush Events. A Mandatory Flush is caused by normal address or data flow events.The following events are considered Mandatory Flush Events:

- An outbound write transfer size other than an aligned 32-bit word.
- An outbound write to a non-contiguous address, including byte misalignment, of an existing collection.
- An outbound read cycle.
- An outbound write of any size or address from a processor other than the one responsible for the current collection.

There are two Optional Flush Events; the Store-Gather Backup Timer and the Store-Gather Sync Flush. The Store-gather Backup Timer is a programmable timer that watches for large gaps of time between contributions to a collection. The timer is reset anytime a contribution is made to a collection. If another contribution is not made before the timer times-out, the current collection will be flushed. The timer may be disabled if so desired. The characteristics of the Store-Gather Timer apply globally to all applicable outbound traffic and is determined by the SBT field within the **BXCS** register.

The Store-Gather Sync Flush is an option that causes a collection to be flushed whenever the PPC Slave detects a Sync bus cycle from the processor responsible for the current collection. This is also a globally programmable option and is controlled by the SSF field within the **BXCS** register.

Read Ahead

The Harrier has an optional Read Ahead mode that enables the PCI Master to prefetch data whenever a read transaction occurs on the PowerPC bus. The prefetched data would be readily available from the Outbound FIFO if multiple contiguous address read transactions occur on the PowerPC bus.This option may be individually enabled for each Outbound Translation Function from within the **OTATx** register.

The PPC Slave only attempts to perform read ahead actions on word (32-bit), dword (64-bit) and cache line operands. The PPC Slave keeps the read ahead command open until a forced flush condition occurs. If forced to flush, the PPC Slave closes the current read ahead command and clears the Outbound FIFO. Once the PCI Master sees that the read ahead command is closed, it discontinues with the read and close the command.

There are two groups of events that can force a collection flush; Mandatory Flush Events and Optional Flush Events. A Mandatory Flush is caused by normal address or data flow events.The following events are considered Mandatory Flush Events:

- An outbound read transfer size other than an aligned 32-bit word.
- An outbound read from a non-contiguous address, including byte misalignment, of an existing collection.
- An outbound write cycle.
- An outbound read of any size or address from a processor other than the one responsible for the current collection.

There are two Optional Flush Events; the Read Ahead Backup Timer and the Read Ahead Sync Flush. The Read Ahead Backup Timer is a programmable timer that watches for large gaps of time between contiguous address reads. The timer is reset anytime a read is performed

on read ahead enabled mapped region. If the next address read is not made before the timer times-out, the current read ahead command is closed and the Outbound FIFO is cleared. The timer may be disabled if so desired. The characteristics of the Read Ahead Timer apply globally to all applicable outbound traffic and is determined by the RBT field within the **BXCS** register.

The Read Ahead Sync Flush is an option that will close the read ahead command and clear the Outbound FIFO whenever the PPC Slave detects a Sync bus cycle from the processor responsible for the current read ahead. This is also a globally programmable option and is controlled by the RSF field within the **BXCS** register.

Passive Slave

The PPC Slave supports a special function called *Passive Slave* which allows a user defined block of PowerPC memory space to be mapped to PCI memory space. Any writes to this block of PowerPC memory space by any PowerPC bus master is copied to PCI memory space. Any reads from this block of PowerPC memory space is ignored.

The PPC Slave does not actively respond to cycles that are mapped to the Passive Slave function. The PPC Slave passively tracks PowerPC bus cycles, and if a cycle qualifies as a Passive Slave transaction the PPC Slave posts the write command and write data as they are being acknowledge by the PowerPC slave.

Since the Passive Slave function uses the outbound path, it is also restricted by the Outbound FIFO limitations. If the Command FIFO or Write Data FIFO does not have enough room to service the current write or if the Command FIFO contains a previously posted delayed write transaction, the PPC Slave will assert ARTRY_ at the earliest possible retry window to prevent the PowerPC slave from acknowledging the cycle. The PPC Slave will continue to retry the cycle until the Outbound FIFO is in a ‘favorable state’.

The Passive Slave function has an associated set of registers. The **PSAD** register defines the PowerPC memory space base address. The **PSOF** register defines the offset to be added to the upper 16 bits of the PowerPC address bus to determine the PCI bus address. The **PSSZ** register defines

the resource size starting from the base address in the **PSAD** register. The resource size is programmable from 4kbytes to 2Gbytes. The **PSAT** register is used to enable the Passive Slave function and enable store-gathering.

Error Handling

There is only one distinct type of error associated with the Outbound Function.

- PCI Bus Errors

PCI Bus Errors:

While processing a transaction, the PCI Master can encounter either a Master Abort or a Target Abort. Both types of errors will terminate a transaction and cause an Error Exception to be generated. There is no visibility of either type of error to the PPC Slave, therefore it is the sole responsibility of the PCI Master to maintain the Outbound FIFO in the event that an error occurs.

The PCI Master takes the same action regardless of which error type is encountered. There are four basic actions taken:

- If the transaction was a write and Store-Gathering was not used, then the PCI Master will invalidate the remaining portion of the aborted transaction by discarding the contents of the Outbound FIFO. A transaction of this type will always be either a single beat or a burst transaction, and the exact number of words to be discarded is known by the PCI Master.
- If the transaction was a write and Store-Gathering was used, then the PCI Master will continually discard the contents of the Outbound FIFO until the transaction is closed by the PPC Slave. Once closed, The PCI Master will remove the remaining un-transferred portion of the transaction from the Outbound FIFO.

  Since the PPC Slave cannot detect the occurrence of a PCI error, a Store-Gather collection is not affected by the error and continues without interruption until a forced flush event occurs.

- If the transaction was a read and read-ahead was not used, then the PCI Master invalidates the remaining portion of the aborted transaction by filling the Outbound FIFO with all ones. A transaction of this type will always be either a single beat or a burst transaction, and the exact number of words to be filled is known by the PCI Master.
- If the transaction was a read and read-ahead was used, then the PCI Master will continually fill the Outbound FIFO with all ones until the transaction is closed by the PPC Slave.

  Since the PPC Slave cannot detect the occurrence of a PCI error, a read-ahead collection will not be affected by the error and will continue without interruption until a forced flush event occurs.

Inbound Functions

An inbound transaction is originated from the PCI bus and is targeted to the PowerPC bus. The key functional elements are the PCI Slave, the Inbound FIFO, and the PPC Master. This section describes in detail the elements of the Harrier that are associated with inbound transactions.

PCI Decode

The Harrier provides three resources to PCI:

- *PCFS* Register Group mapped into PCI Configuration space
- PowerPC bus address space mapped into PCI Memory or I/O space
- *PMEP* Register Group into PCI Memory space

PCFS Register Group:

The *PCFS* Register Group is mapped within PCI configuration space according to how the system connects the Harrier's DEVSEL_ pin. The Harrier provides a configuration space that is fully compliant with *PCI Local Bus Specification 2.1*. There are five base registers within the standard 64 byte header. One register is dedicated to mapping the *PMEP* Register Group into PCI Memory space. The remaining four base registers are associated with Inbound Translation Functions that map PowerPC

address space into PCI address space. Additional control associated with each Inbound Translation Function is contained within device specific registers mapped above the 64 byte header.

PowerPC Bus Address Space:

The Harrier maps PowerPC address space into PCI Memory or I/O space using four programmable map decoders. Each map decoder is accompanied by some address translation logic, and is collectively referred to as an Inbound Translation Function.

An Inbound Translation Function is mapped into PCI Memory or I/O space using its own PCI compliant Base Address Register (BAR). A BAR is used by a PCI master to determine a resource size, and is ultimately written with a value indicating where in PCI Memory or I/O space the resource is to reside.

The upper portion of the PCI address is compared with the upper programmable portion of each BAR, and if the address falls within the specified range, the access is passed on to the PowerPC bus. An example of this is shown in the figure below.

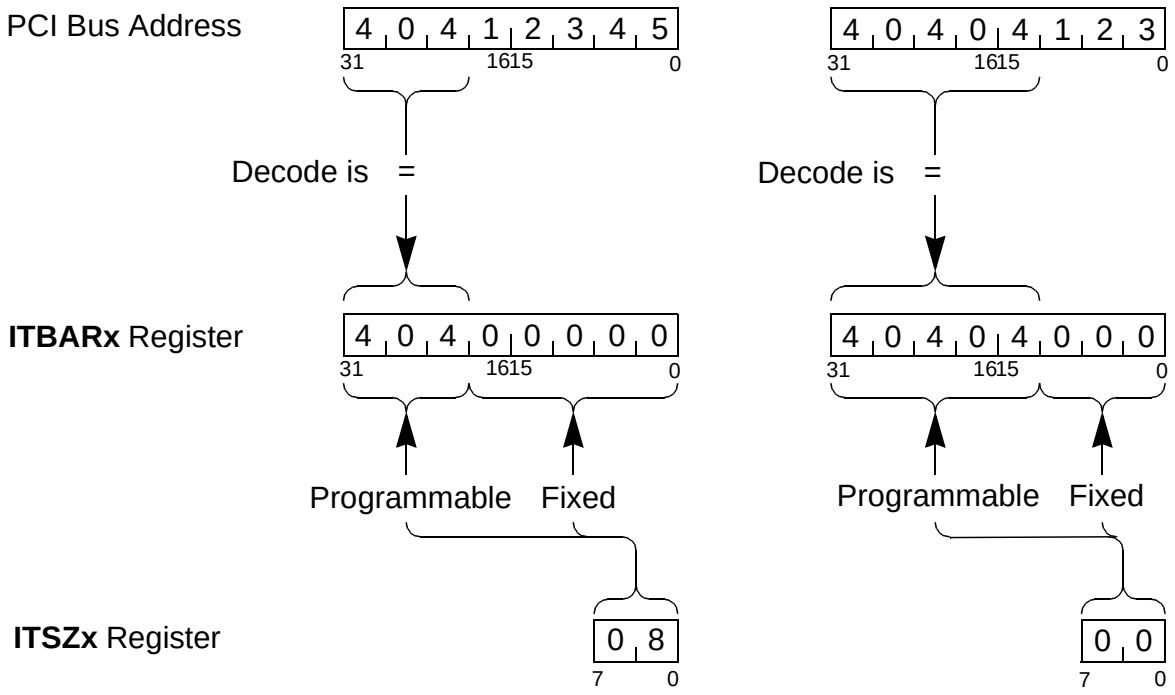


Figure 2-8. Inbound Address Decoding

The size of each Inbound Translation Function is independent of each other, and may be programmed to offer as small as 4KBytes or as large as 2 GBytes. The read-write characteristics of a BAR will change to reflect the size of the resource. The BAR limits the placement of an Inbound Translation Function to binary boundaries according to resource size. For example, a 4KByte resource can be located on any 4KByte boundary.

Each Inbound Translation Function has an independent set of attributes. These attributes are used to enable read and write accesses, select either Memory or I/O space, enable read-ahead and write-posting, and define the PowerPC bus transfer characteristics.

Each Inbound Translation Function also includes a programmable 16-bit address offset that is added to the PCI address in a two step process. The first step is to subtract the base address from the PCI address. This essentially zero’s out all bit positions of the PCI address that correspond to programmable bit positions within the BAR. The second step is to add the programmable offset to the 16 most significant bits of the results of the first step, and the result is used as the PowerPC address. This offset provides a high degree of decoupling between PowerPC address space and PCI address space. An example of this is show in the figure below.

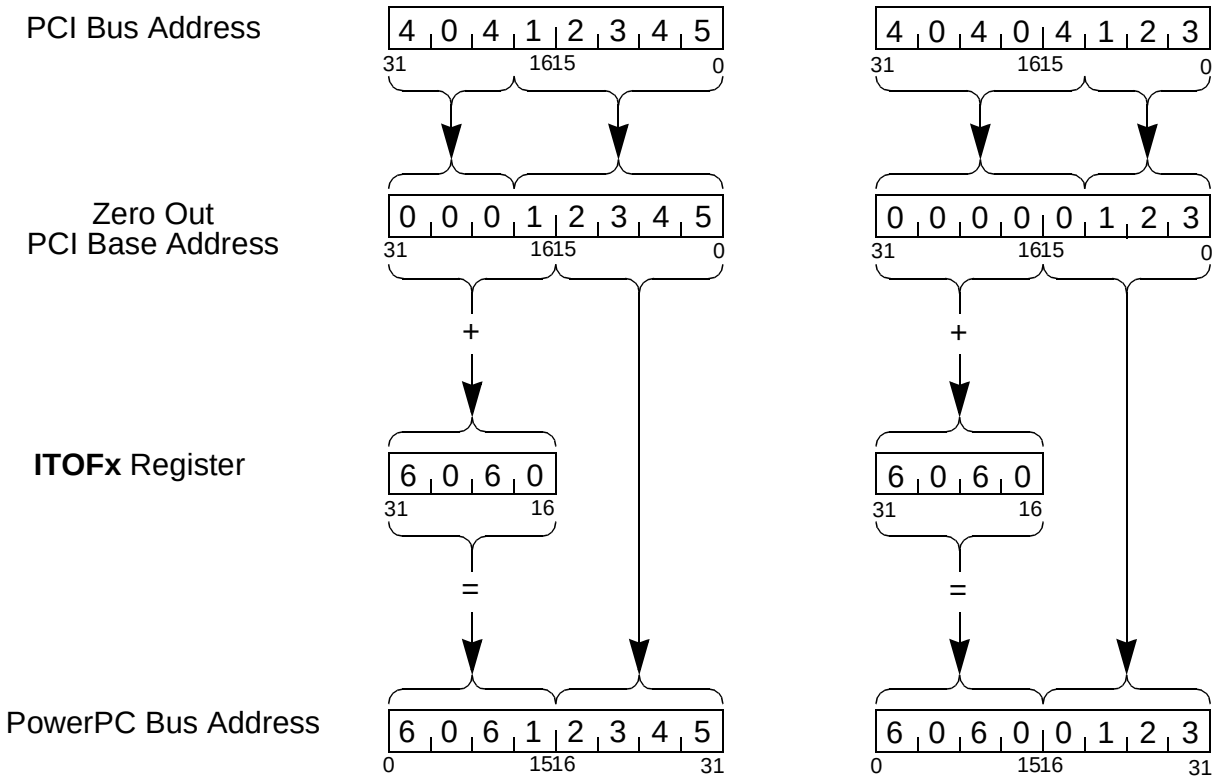


Figure 2-9. Inbound Address Translation

All address decoding is prioritized so that programming multiple functions to respond to the same address is not a problem. When the PCI address falls into the range of more than one function, only the highest priority one will respond. The functions are prioritized as shown in the table below.

Table 2-3. Map Decoder Priority

Decoder	Priority
Inbound Translation Function 0	highest
Inbound Translation Function 1	|
Inbound Translation Function 2	V
Inbound Translation Function 3	lowest

PCI Message Passing Register Group:

The Harrier offers a 4K byte block of register space that supports I_2O and Generic Message Passing. This block can be located anywhere within PCI Memory space using a traditional PCI defined base register within the 64-byte configuration header. Refer to the section titled *PCI Message Passing (PMEP) Register Group* on page 3-23 for more information.

PCI Slave

The PCI Slave provides the control logic needed to interface the PCI bus to the Inbound FIFO. The PCI Slave can accept either 32-bit or 64-bit transactions, however it can only accept 32-bit addressing. There is no limit to the length of the transfer that the PCI Slave can handle.

The PCI Slave accepts four basic types of transactions:

- **Posted Write**: Is accepted either in the form of a single beat or a burst transaction. It generally accepts data with zero target wait states, and inserts wait states whenever either the data or command FIFOs reach saturation. It initiates a disconnect in accordance with PCI initial and subsequent latency requirements.
- **Compelled Write**: Is accepted either in the form of a single beat or a burst transaction, however it is always terminated after the transfer of one data beat. It inserts wait states until the beat of data has been transferred on the PowerPC bus. It initiates a disconnect in accordance with PCI initial and subsequent latency requirements.
- **Posted Read** (read-ahead enabled): Is accepted either in the form of a single beat or a burst transaction. It initiates a sequence of contiguous cache-line reads on the PowerPC bus. It generally returns read data to the PCI Slave before issuing an initial latency disconnect. It incorporates a *delayed transaction* protocol to be compliant with the PCI Slave's initial and subsequent latency requirements.
- **Compelled Read** (read-ahead disabled): Is accepted either in the form of a single beat or a burst transaction, however it is always terminated after the transfer of one data beat. It initiates a single beat read on the PowerPC bus. It generally returns read data to the PCI

Slave before issuing an initial latency disconnect. It incorporates a *delayed transaction* protocol to be compliant with the PCI Slave's initial and subsequent latency requirements.

The Harrier is designed to always comply with the PCI Slave's initial and subsequent latency requirements. It supports a programmable initial latency, providing either a 16 clock target latency or a 32 clock host bridge cache hit latency. During posted writes that cannot be serviced immediately (i.e. a full FIFO), the Harrier always waits as long as the initial latency allows before issuing a disconnect. When performing a read, the Harrier also waits for the maximum allowable initial latency period before converting the read to a *delayed transaction*.

The PCI Slave supports a single level *delayed transaction* protocol. A delayed transaction is a sequence of events used to process large latency transactions that would otherwise consume a large amount of PCI bus bandwidth. In general, all read transactions are handled by the Harrier as a delayed transaction. A delayed transaction is processed in the following manner:

- A PCI master issues a bus cycle with the intention of accessing a resource hosted by the Harrier. The Harrier attempts to service the transaction before the initial latency period expires. If unable to provide service, the Harrier terminates the PCI bus cycle with a disconnect-retry.
- The Harrier only accepts a new transaction if the Inbound FIFO is in a 'favorable state'. A 'favorable state' is when:
  - There are no entries for a previously issued but uncompleted delayed transaction within the Command FIFO, and
  - There is enough room in the Command FIFO to accept the current transaction.

  If conditions are favorable, the Harrier accepts the transaction, posts the new entry in the Command FIFO, and the PCI Slave retains a copy of the address and attributes of the transaction for future reference.

  If conditions are not favorable, the Harrier does not accept the transaction and a new entry is not posted.

- The Harrier proceeds to service the delayed transaction to completion. While doing so, the PCI master is continuously re-attempting the transaction. Each time the transaction is re-attempted, the Harrier waits for the expiration of the initial latency period before issuing a disconnect-retry.
- If another PCI master attempts a read from the Harrier, while the Harrier is currently processing a delayed read transaction, the Harrier does not wait for the expiration of the initial latency period but immediately issues a disconnect-retry.
- The PPC Master is primarily responsible for completing the delayed transaction by filling the Inbound FIFO from the PowerPC bus.
- The PCI Slave is notified by the PPC Master upon completion of the transaction, at which time the PCI Slave is ready to close out the delayed transaction. The PCI Slave will wait for the PCI master to once again retry the transaction. The PCI Slave will validate the address and attributes of the current PCI bus cycle with those of previously accepted delayed transaction bus cycle. If there is a match, then the PCI Slave will complete both the address tenure and the data tenure. At this point, a delayed transaction is considered complete.

The Harrier's delayed read mechanism only guarantees the completion of the first data beat of a read cycle. When a delayed read completion is finally established between the Harrier and the PCI master, the Harrier continues to provide burst read data as needed. If a disconnect is issued any time after the first beat of data has been transferred, the PCI master re-attempts the transaction and the Harrier considers this to be an entirely new transaction.

Inbound FIFO

The Inbound FIFO is used to hold data between the PCI Slave and the PPC Master to ensure that optimum data throughput is maintained. The Inbound FIFO consists of three major components; the command path, the read data path, and the write data path.

The command path incorporates a 52-bit by 8 entry FIFO which is used to hold command information being passed between the PCI Slave and the PPC Master. If write-posting is enabled, up to eight single beat or burst transactions may be posted. If this limit is exceeded, any pending PCI transactions are retried until the PPC Master completes a portion of the previously posted transactions and creates some room in the command FIFO.

Each data path uses a 256 byte (32 entries/8 cache lines by 64-bit) FIFO. The operation of the write data path is completely independent of the read data path. This allows the Harrier to accept write-posted transactions while servicing a delayed read transaction. If a read data path FIFO limit is reached, then the PPC Master stops prefetching until the PCI Slave manages to empty the FIFO beyond a certain programmable threshold. If a write data path FIFO limit is reached, then the PCI Slave continually issues retries until the PPC Master manages to empty some portion of the FIFO.

The Harrier does not support byte merging or byte collapsing. Each and every single beat byte transaction presented to the PCI Slave is presented to the PowerPC bus as a unique single beat transfer.

The Harrier does not support PowerPC or PCI bus snooping. Care should be exercised when using write-posting or read-ahead within coherent address space.

PPC Master

The PPC Master can transfer data either in 1-to-8 byte single beat transactions or 32 byte four beat burst transactions. This limitation is strictly imposed by the PowerPC bus protocol. The PPC Master attempts to move data using burst transfers whenever possible. If a transaction starts on a non-cache line address, the PPC Master performs as many single beat transactions as needed until the next highest cache line boundary is reached. If a write transaction ends on a non-cache line boundary, then the PPC Master finishes the transaction with as many single beat transactions

as needed to complete the transaction. The table below shows the relationship between starting addresses and PowerPC bus transaction types when write-posting and read-ahead are enabled.

Table 2-4. PPC Master Transaction Profiles and Starting Offsets

Start Offset (i.e. from 0x00,0x20,0x40, etc.)	Write Profile	Read Profile	Notes
0x...00 -> 0x....07	Burst @ 0x00 Burst @ 0x20	Burst @ 0x00 Burst @ 0x20	Most efficient
0x....08 -> 0x....0f	Single @ 0x08 Single @ 0x10 Single @ 0x18 Burst @ 0x20	Burst @ 0x00 Burst @ 0x20	Discard read beat 0x00
0x....10 -> 0x....17	Single @ 0x10 Single @ 0x18 Burst @ 0x20	Burst @ 0x00 Burst @ 0x20	Discard read beat 0x00 and 0x08
0x....18 -> 0x....1f	Single @ 0x18 Burst @ 0x20	Single @ 0x18 Burst @ 0x20	

Write Posting

The Harrier has an optional write-posting mode that may be enabled by the WPE field within the **ITATx** registers. While the PCI Slave is filling the Inbound FIFO with write data, the PPC Master can be moving previously posted write data onto the PowerPC bus. In general, the PowerPC bus is running at a higher clock rate than the PCI bus, which means the PCI bus can transfer data in a continuous uninterrupted burst, while the PowerPC bus transfers data in distributed multiple bursts.

The Harrier write-posting function does not incorporate any programmable tuning options. Please refer to the section titled *FIFO Tuning* on page 2-41 for more information.

Read Ahead

The Harrier has an optional read-ahead mode controlled by the RAE bit in the **ITATx** registers that allows the PPC Master to prefetch data in bursts and store it in the Inbound FIFO. The contents of the Inbound FIFO is then used to satisfy the data requirements for the remainder of the PCI read transaction.

Upon completion of a prefetched read transaction, any residual read data left within the Inbound FIFO is invalidated (discarded). When servicing a delayed read transaction, the contents of the Inbound FIFO is invalidated upon disconnect if at least one data beat of the delayed transaction has been transferred.

The PPC Master never performs prefetch reads beyond the address range mapped within the Inbound Translation Function map decoders. As an example, assume the Harrier has been programmed to respond to PCI address range $10000000 through $1001FFFF with an offset of $2000. The PPC Master performs its last read on the PowerPC bus at cache line address $3001FFFC or dword address $3001FFF8.

Copyback Snarfing

Many times an inbound read takes place in coherent memory space that is held within the processor's cache. When this happens, the processor detects a snoop hit on the Harrier's bus cycle. The processor then performs a copy-back write cycle to return the modified cache-line to local memory. The Harrier has an optional Copy-back Snarfing mode that enables the PPC Master to "snarf" the processor's copy-back write cycle. Snarfing means that the PPC Master listens to the processor write cycle and grabs a copy of the data for its own use. A successful snarf of a cache-line means a savings of PowerPC bus bandwidth since the PPC Master does not have to initiate its own read cycle from local memory.

The PPC Master pays close attention to the copy-back write cycle. It must make sure that the cycle following a snoop hit is an associated copy-back cycle. If the cycle does not meet the criteria of a valid copy-back cycle, then the PPC Master simply performs a read of the needed cache-line.

The Copy-back Snarfing mode can be controlled by the CSE field within the **BXCS** register. The default state for this option is enabled.

Bus Hog

The Harrier has an optional mode called Bus Hog. When Bus Hog is enabled, the PPC Master continually requests the PowerPC bus for the entire duration of each PCI transfer. When Bus Hog is not enabled, the PPC Master structures its bus request actions around its desire to satisfy FIFO thresholds. The Bus Hog mode was primarily designed to assist with system level debugging and is not intended for normal modes of operation. It is brute force method of guaranteeing that all inbound transactions are performed without any intervention by host CPU transactions. The Bus Hog mode can be controlled by the BHG field within the **BXCS** register. The default state for BHG is disabled.

Error Handling

There is only one distinct type of error associated with the Inbound Function: Delayed Transaction Time-out.

Delayed Transaction Time-out:

The Harrier expects all delayed transactions to be continuously retried by a PCI master until the transaction has completed. Besides being in violation of the PCI specification, it is considered a serious error if a PCI master aborts (i.e. stops retrying the transaction) a delayed transaction before the Harrier finishes the transaction. In the event that this happens, the Harrier has special logic that facilitates recovery from this type of error. An 8 microsecond timer starts counting once the PCI Slave is ready to close out a delayed transaction. If a PCI master does not come back to complete the transaction before the timer expires, then an Error Exception is generated. The delayed transaction is invalidated, and the contents of the Inbound FIFO are discarded. Unless the transaction was targeted to read sensitive PowerPC address space, the effects of an invalidated read transaction are minimal.

Generating PCI Cycles

There are four basic types of bus cycles that can be generated on the PCI bus:

- Memory and I/O

- Configuration
- Special Cycle
- Interrupt Acknowledge

Memory and I/O Cycles:

Each Outbound Translation Function may be configured to generate PCI I/O or Memory accesses according to the MEM and IOM fields within the **OTATx** registers as shown in the following table.

Table 2-5. Memory and I/O Attributes

MEM	IOM	PCI Cycle Type
1	x	Memory
0	0	Contiguous I/O
0	1	Spread I/O

If the MEM bit is set, the Harrier performs Memory addressing on PCI. The Harrier takes the PowerPC address, applies the offset specified in the **OTOFx** register, and maps the result directly to the PCI bus.

The IBM CHRP specification describes two approaches for handling PCI I/O addressing: contiguous or spread address modes. When the MEM bit is cleared, the IOM bit is used to select between these two modes whenever a PCI I/O cycle is performed.

The Harrier performs contiguous I/O addressing when the MEM bit is clear and the IOM bit is clear. The Harrier takes the PowerPC address, applies the offset specified in the **OTOFx** register, and maps the result directly to the PCI bus.

The Harrier performs spread I/O addressing when the MEM bit is clear and the IOM bit is set. It takes the PowerPC address, applies the offset specified in the **OTOFx** register, and maps the result to the PCI bus as shown in the figure below.

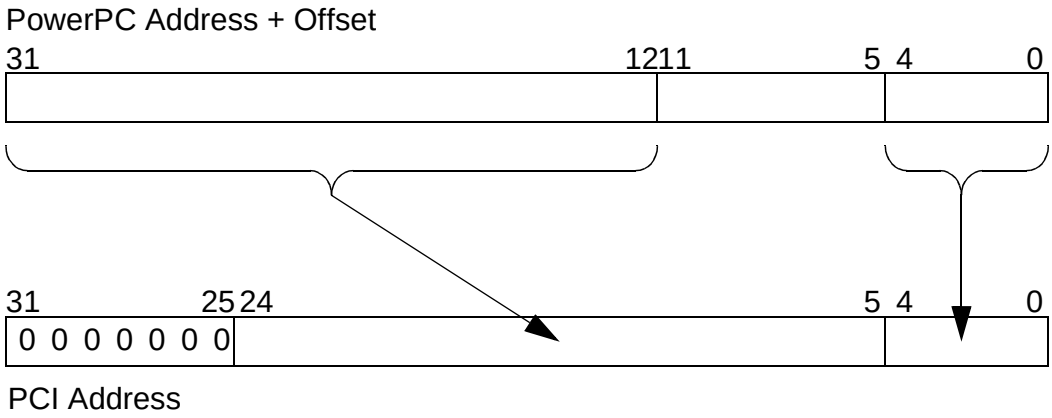


Figure 2-10. Spread I/O Address Translation

Spread I/O addressing allows each PCI device's I/O registers to reside on a different PowerPC memory page so device drivers can be protected from each other using memory page protection.

All I/O accesses must be performed within natural word boundaries. Any I/O access that is not contained within a natural word boundary causes an unpredictable operation. For example, an I/O transfer of 4 bytes starting at address $80000010 is considered a valid transfer. An I/O transfer of 4 bytes starting at address $80000011 is considered an invalid transfer since it crosses the natural word boundary at address $80000013/$80000014.

Configuration Cycles:

The Harrier uses Configuration Mechanism #1 as defined in *PCI Local Bus Specification 2.1* to generate configuration cycles. Please refer to this specification for a complete description of this function.

Configuration Mechanism #1 uses an address register/data register format. Performing a configuration access is a two step process. The first step is to place the address of the configuration cycle within the **CONFIG_ADDRESS** register. Note that this action does not generate any

cycles on the PCI bus. The second step is to either read or write configuration data into the **CONFIG_DATA** register. If the **CONFIG_ADDRESS** register has been set up correctly, the Harrier passes this access on to the PCI bus as a configuration cycle.

The addresses of the **CONFIG_ADDRESS** and **CONFIG_DATA** registers are actually embedded within PCI I/O space. If the **CONFIG_ADDRESS** register has been set incorrectly or the access to either the **CONFIG_ADDRESS** or **CONFIG_DATA** register is not 1,2, or 4 bytes wide, the Harrier passes the access on to PCI as a normal I/O Space transfer.

The **CONFIG_ADDRESS** register is located at offset $0CF8 from the bottom of PCI I/O space. The **CONFIG_DATA** register is located at offset $0CFC from the bottom of PCI I/O space. Outbound Translation Function 3 is designed so **OTAD3**,**OTOF3**, and **OTAT3** must be used for mapping PCI Configuration (consequently I/O) space. This register group is initialized at reset to allow PCI I/O accesses starting at address $80000000. The power up location (i.e. Little Endian disabled) of the **CONFIG_ADDRESS** register is $80000CF8, and the **CONFIG_DATA** register is located at $80000CFC.

The **CONFIG_ADDRESS** register must be prefilled with four fields; the Register Number, the Function Number, the Device Number, and the Bus Number.

The Register Number and the Function Number are passed along to the PCI bus as part of the lower address bits.

When performing a configuration cycle, the Harrier uses the upper 20 address bits to drive IDSEL lines. During the address phase of a configuration cycle, only one of the upper address bits is set. The device that has its IDSEL connected to the address bit being asserted is selected

for a configuration cycle. The Harrier decodes the Device Number to determine which of the upper address lines to assert. The decoding of the 5 bit Device Number is show in the table below.

Table 2-6. Configuration Device Decode

Device Number	Address Bit
00000	AD31
00001 - 01010	All Zeros
01011	AD11
01100	AD12
(etc.)	(etc.)
11101	AD29
11110	AD30
11111	All Zeros

The Bus Number determines which bus is the target for the configuration read cycle. The Harrier always hosts PCI bus #0. Any access that is performed on the PCI bus connected to the Harrier must have zero programmed into the Bus Number. If the configuration access is targeted for another PCI bus, then that bus number should be programmed into the Bus Number field. The Harrier detects a non zero field and converts the transaction to a Type 1 Configuration cycle.

Special Cycles:

The Harrier supports the method stated in *PCI Local Bus Specification 2.1* using Configuration Mechanism #1 to generate Special Cycles. To prime the Harrier for a Special Cycle, the host processor must write a 32 bit value to the **CONFIG_ADDRESS** register. The contents of the write are defined in the section titled "CONFIG_ADDRESS Register" in Chapter 3. After the write to **CONFIG_ADDRESS** has been accomplished, the next write to the **CONFIG_DATA** register causes the Harrier to generate a Special Cycle on the PCI bus. The write data is driven onto AD[31:0] during the Special Cycle data phase.

Interrupt Acknowledge Cycles:

Performing a read from the **PIAC** register initiates a single PCI Interrupt Acknowledge cycle. Any single byte or combination of bytes may be read from, and the actual byte enable pattern used during the read is passed on to the PCI bus. Upon completion of the PCI interrupt acknowledge cycle, the Harrier presents the resulting vector information obtained from the PCI bus as read data.

FIFO Tuning

The selection of a FIFO size and threshold levels used in conjunction with write-posting and read-ahead determines the performance characteristics of a bridge device. The Harrier is designed to work with an average transfer size of 128 bytes or more, however there are mechanisms in place to improve performance for smaller transfer sizes.

The tuning of write transactions is much easier than the tuning of read transactions. Write transactions always have a finite transaction size. Read transactions involve a bit of speculation. An incorrect speculation usually results in a certain level of system performance degradation.

Write-Posting

The characteristics of the inbound data write FIFO are identical to those of the outbound data write FIFO. The Harrier does not provide programmable options associated with these FIFOs. Both FIFO sizes are fixed at 256 bytes (8 cache lines). The start threshold is fixed at the half way mark (i.e. 4 cache lines), and the stop threshold is fixed at the almost empty mark (i.e. <= 1 cache line).

Read-Ahead

The characteristics of the inbound data read FIFO are identical to those of the outbound data read FIFO. Both FIFOs offer virtual sizing and a variable threshold to control the effects of read-ahead. Each Outbound Translation Function offers a unique set of parameters. Each Inbound Translation Function offers a distinct set of parameters for PCI Read/Read Line commands and for PCI Read Multiple commands.

The table below summarizes the options provided by the Harrier.

Table 2-7. Harrier Read-Ahead Options

Virtual FIFO Size	FIFO Threshold	Initial Read Size	Subsequent Resume	Subsequent Stop
64 Bytes	Half	2 cache lines	FIFO <= 1 cache line	FIFO = 2 cache lines
	Empty		FIFO = 0 cache lines	
128 Bytes	Half	4 cache lines	FIFO <= 2 cache lines	FIFO = 4 cache lines
	Empty		FIFO = 0 cache lines	
256 Bytes	Half	8 cache lines	FIFO <= 4 cache lines	FIFO = 8 cache lines
	Empty		FIFO = 0 cache lines	

Selecting an optimal set of parameters is related to the expected average transfer size. The ideal solution is to match the prefetch size to the transfer size. The next level of success is to minimize the amount of unnecessary prefetching. A example of a poorly optimized solution would be to select a very large virtual FIFO size for very small transfers. This solution results in a large portion of the prefetched bandwidth being wasted.

Some general guidelines and characteristics associated with the outbound path are as follows:

- Select the smallest FIFO size that is still greater than the expected outbound transaction size.
- A continuous burst with a small FIFO size creates multiple smaller burst transfers on PCI. The increase amount of transfers on PCI makes the overall transaction more susceptible to bandwidth degradation due to initial latencies on PCI.

- A continuous burst with a large FIFO size creates fewer but longer burst transfers on PCI, and is less susceptible to initial latency bandwidth degradation.
- Selection of an empty threshold works well if the transactions are almost always smaller than the virtual FIFO size. If the transaction size exceeds the FIFO size, then the transaction bandwidth will be severely degraded since wait states will be incurred between when the FIFO goes empty and when the PCI Master can start filling the FIFO again.
- Boundary cases may not work as expected. The PPC Slave is not very efficient at determining when a read-ahead collection has completed. For example, if the processor reads 64 bytes and stops, the PPC Slave will still consider the collection open until told otherwise by the processor. If configured for an empty threshold with a 64 byte FIFO, the completion of the transaction will more than likely result in another prefetch by the PCI Master.

  Note that a transaction size that is slightly smaller than the FIFO size with an empty threshold will behave as expected. In this case, the processor/PPC Slave has no problems closing a collection before the PCI Master starts the next prefetch read.

Some general guidelines and characteristics associated with the inbound path are as follows:

- Select the smallest FIFO size that is still greater than the expected inbound transaction size.
- A continuous burst with a small FIFO size creates smaller groups of burst transfers on PowerPC. The increase number of groups on PowerPC makes the overall transaction more susceptible to bandwidth degradation due to initial latencies on PowerPC and within the SDRAM Memory Controller.
- A continuous burst with a large FIFO size creates larger groups of burst transfers on PowerPC, and is less susceptible to initial latency bandwidth degradation.

- Selection of an empty threshold works well if the transactions are almost always smaller than the virtual FIFO size. If the transaction size exceeds the FIFO size, then the transaction bandwidth will be severely degraded since wait states will be incurred between when the FIFO goes empty and when the PPC Master can start filling the FIFO again.
- Boundary cases work exactly as expected. The PCI bus explicitly indicates the completion of a burst read transaction. This indication can propagate fast enough to the PPC Master to inhibit any additional prefetch reads.

Transaction Ordering

All transactions will be completed on the destination bus in the same order that they are issued on the originating bus. A read or a compelled write transaction will force all previously issued write posted transactions to be flushed from the FIFO. All write posted transfers will be completed before a read or compelled write is begun to assure that all transfers are completed in the order issued.

The Harrier can accept posted-write transactions while servicing a delayed read, however the ordering on the destination bus always allows the read transaction to complete before the write transactions.

The Harrier's design eliminates any adverse affects that can occur when control register bits are changed during transaction processing. In some special cases, a register access may be delayed while Harrier performs the necessary change, but in general the register access time is unaffected.

The *PCI Local Bus Specification 2.1* states that posted write buffers in both directions must be flushed before completing a read in either direction. Harrier supports this by providing two optional FIFO flushing options. The FBR (Flush Before Read) bit within the **BXCS** register controls the flushing of inbound write-posted data when performing outbound read transactions. The FBR (Flush Before Read) bit within the **BPCS** register controls the flushing of outbound write-posted data when performing inbound read transactions. Both FBR functions are completely independent of each other, however both functions must be enabled to guarantee full compliance with *PCI Local Bus Specification 2.1*.

When the FBR bit within the **BXCS** register is set, the Harrier handles outbound read transactions in the following manner:

- Outbound write-posted transactions are flushed by the nature of the FIFO architecture. The Harrier holds the processor with wait states until the outbound FIFO (Outbound FIFO) is empty.
- Inbound write-posted transactions are flushed as a result of a decision by the PPC Slave to accept or reject a read transaction. When the PPC Slave is asked to service an outbound read transaction, the read is only accepted if there is no inbound write-posted activity. A non-zero value on the inbound FIFO (Inbound FIFO) count is considered write-posted activity. If the read may not be accepted, the PPC Slave issues a retry to the processor.

When the FBR bit within the **BPCS** register is set, the Harrier processes inbound read transactions in the following manner:

- Inbound write-posted transactions are flushed by the nature of the FIFO architecture. The Harrier holds the PCI Master with wait states until the inbound FIFO (Inbound FIFO) is empty.
- Outbound write-posted transactions are flushed as a result of a decision by the PCI Slave to accept or reject a read transaction. When the PCI Slave is asked to service an inbound read transaction, the read will only be accepted if there is no outbound write-posted activity. A non-zero value on the outbound FIFO (Outbound FIFO) count is considered write-posted activity. If the read may not be accepted, the PCI Slave will issue a disconnect-retry to the PCI bus.

Endian Conversion

The Harrier supports the natural Endian mode for each bus. The PCI bus is inherently Little-Endian and the PowerPC bus is inherently Big-Endian. All inbound and outbound data must be swapped such that all PCI resources appear as Big-Endian from the perspective of the PowerPC bus.

Conversely, all PowerPC resources appear as Little-Endian from the perspective of the PCI bus. The figure below demonstrates this data swapping function.

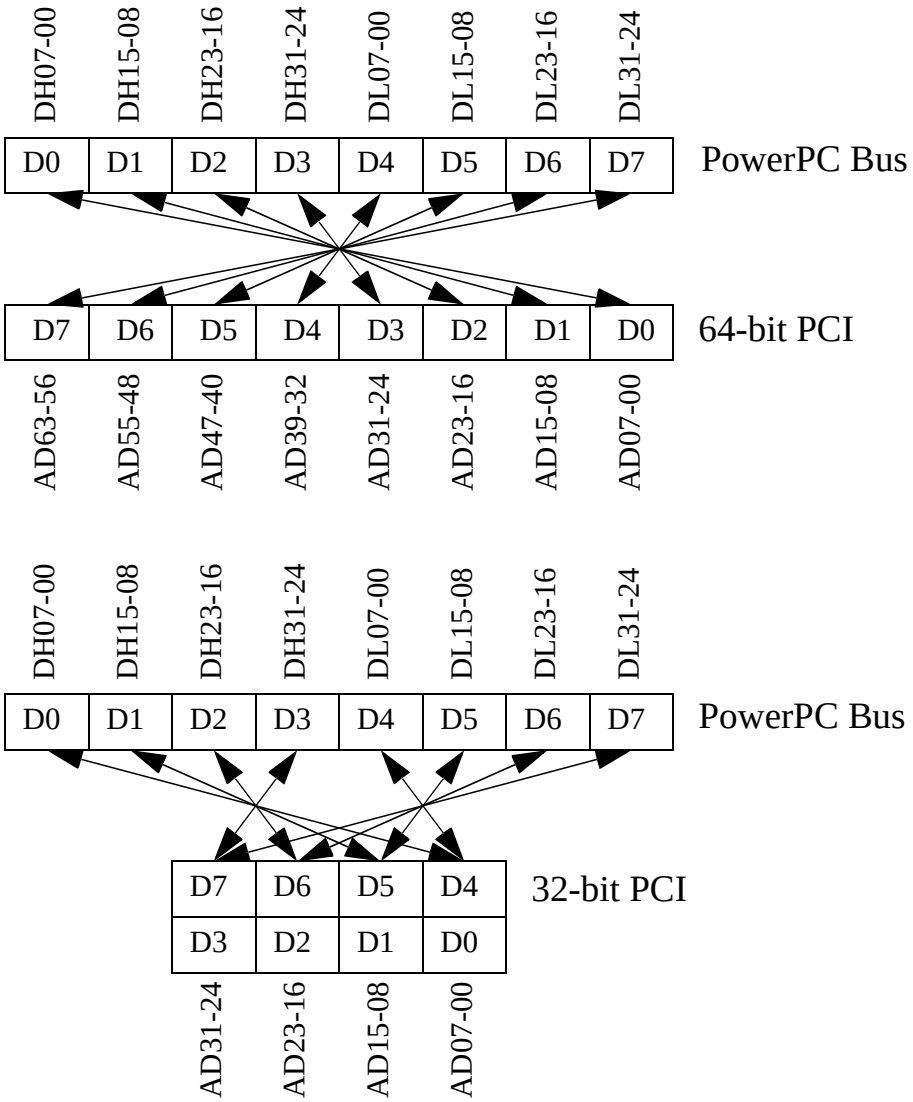


Figure 2-11. Big Endian/ Little Endian Data Swap

The Register Groups are generally represented in the Endian mode associated with the bus that may access the registers. The *XCSR* and *XMPI* PowerPC Register Groups are represented as a Big-Endian resource. The *PCFS* and *PMEP* PCI Register Groups are represented as a Little-Endian resource.

The *XCFS* Register Group, and specifically the **CONFIG_ADDRESS** and **CONFIG_DATA** registers are actually represented as PCI space to the processor and are subject to data swapping.

The **PIAC** register within the *XCSR* Register Group also represents PCI space to the processor and is subject to data swapping.

2

DMA Controller

The Harrier has a single channel DMA Controller that facilitates the transfer of large blocks of data without processor intervention. The DMA Controller is programmed by a simple set of registers that reside within the PowerPC accessible *XCSR* Register Group. If so desired, an Inbound Translation Function may be set up to allow access to the control registers from PCI space. Particular emphasis has been placed on the ability of the DMA Controller to provide a very high data throughput rate and a very simple programming interface.

Please refer to the section titled *PowerPC Control and Status (XCSR) Register Group* on page 3-3 for a summary of the DMA Controller register set.

A block diagram of the DMA Controller is shown in the following figure.

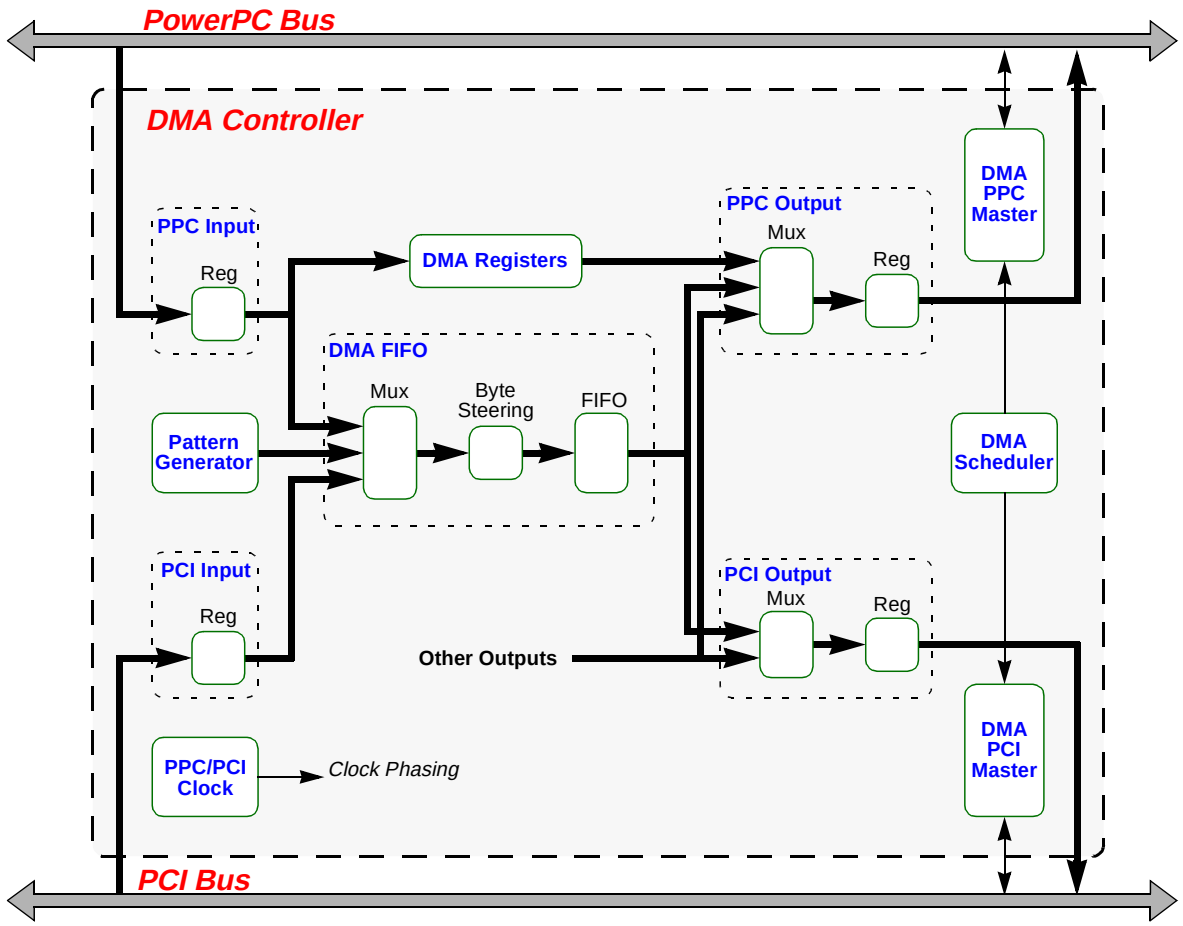


Figure 2-12. DMA Controller Block Diagram

Architecture

The DMA Controller is a stand-alone function that does not infringe on other Harrier resources. The core of the DMA Controller is the DMA FIFO. This FIFO is a 64-bit by 64 entry (512 byte) FIFO. The FIFO is used for all transfers regardless of the direction of the transfer. All interactions with the PowerPC bus are handled by the DMA PPC Master. This module is optimized to transfer data over the PowerPC bus in multiple cache-line bursts. All interactions with the PCI bus are handled by the DMA PCI Master. This module strives to transfer data in continuous 64-bit burst

transfers, although 32-bit burst transfers are supported if used within a 32-bit PCI environment. The DMA Scheduler is a module that oversees the entire DMA operation.

The cache-coherency rules for the inbound and outbound FIFOs apply to the DMA FIFO. The Harrier does not snoop the PowerPC bus for data held within the DMA FIFO, therefore caution should be exercised when using the DMA Controller to move data within coherent memory space.

The DMA Controller does not need to participate in any contention handling protocol. There will never be a time when the completion of a transaction on either the PCI bus or the PowerPC bus is depending on an action from the opposing bus. The DMA Controller works in a read-ahead and write-posted manner, and there will be no compelled options available for DMA transfers.

Normal Endian rules apply for all DMA data movement. Please refer to the section titled *Endian Conversion* on page 2-45 earlier in this chapter for more information.

Operating Modes

The following figure demonstrates the operating modes of the DMA Controller.

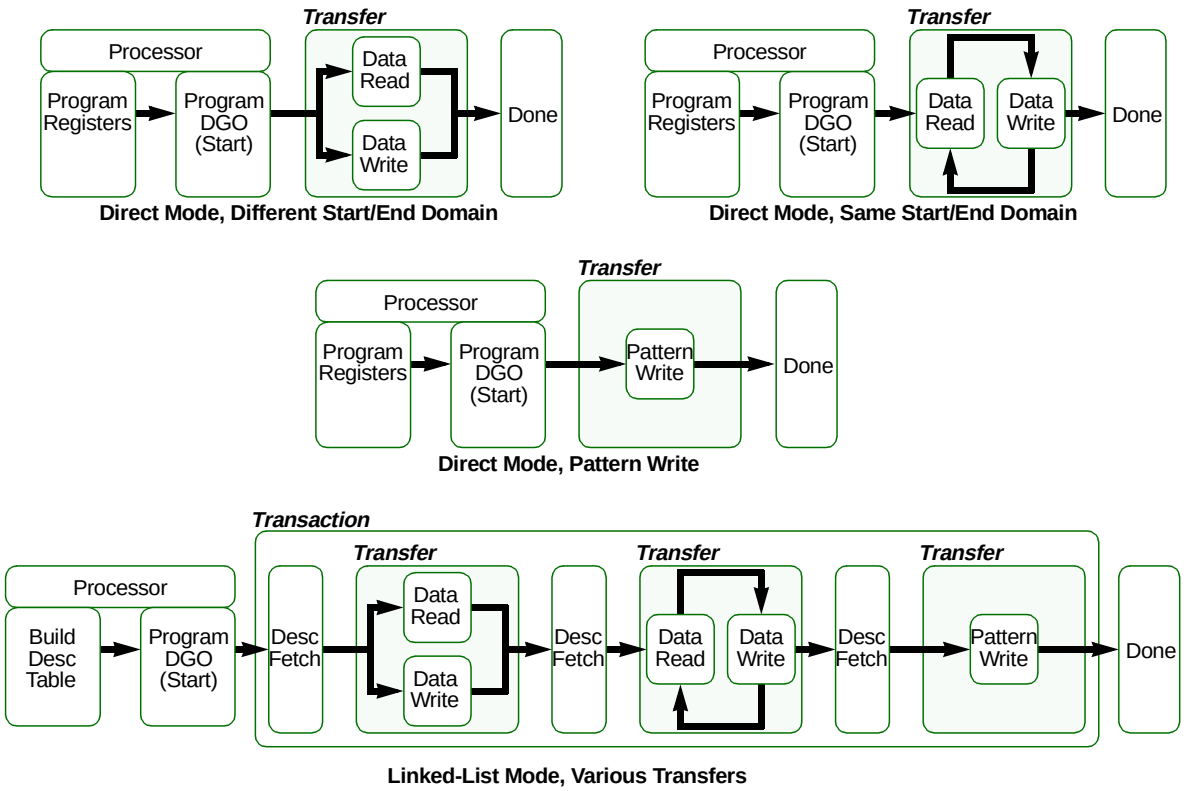


Figure 2-13. DMA Controller Operating Modes

The DMA Controller considers all DMA activity to consist of *transfers* and *transactions*. A *transfer* is a coordinated movement of data from one place to another. The amount of data moved during a *transfer* is only limited by 32-bit PCI and PowerPC addressing, and the direction of data movement during a *transfer* remains fixed. A *transaction* always consists of one or more *transfers*. Each *transfer* within a *transaction* is completely independent of any other. The amount of data moved and the direction of data movement may change between *transfers* within a *transaction*. There are no limits on the number of *transfers* that may take place during a *transaction*.

There are two operating modes for the DMA Controller; Direct Mode and Linked-List Mode. Direct Mode performs one transfer according to a programmed set of values within the DMA control registers. Once the transfer has completed, an indication will be given to the DMA control registers and an optional interrupt will be given to the processor.

Linked-List Mode performs numerous transfers within a transaction. Information about each transfer is obtained from descriptors placed somewhere in PowerPC bus address space. The Harrier fetches these descriptors as needed during the transaction. Once all the descriptors are fetched and the associated transfers are complete, an indication of the completed transaction is given to the DMA control registers and an optional interrupt is given to the processor.

Direction of Data Movement

There are six possible directions for data movement within a transfer. Each is explained below:

1. **PowerPC to PCI**: Data is read from the PowerPC bus and written to the PCI bus. The DMA PPC Master will attempt to fill the DMA FIFO at the same time that the DMA PCI Master will attempt to empty the DMA FIFO.

2. **PCI to PowerPC**: Data is read from the PCI bus and written to the PowerPC bus. The DMA PCI Master will attempt to fill the DMA FIFO at the same time that the DMA PPC Master will attempt to empty the DMA FIFO.

3. **PowerPC to PowerPC**: Data is read from the PowerPC bus and written back sometime later to the PowerPC bus. The DMA PPC Master will fill the DMA FIFO to a certain point, after which the DMA PPC Master will empty the DMA FIFO.

4. **PCI to PCI**: Data is read from the PCI bus and written back sometime later to the PCI bus. The DMA PCI Master will fill the DMA FIFO to a certain point, after which the DMA PCI Master will empty the DMA FIFO.

5. **Data Pattern to PowerPC**: A data pattern is written into the DMA FIFO and then written to the PowerPC bus. The Pattern Generator will attempt to fill the DMA FIFO at the same time that the DMA PPC Master will attempt to empty the DMA FIFO The data pattern may either be a fixed pattern or an incrementing pattern.

6. **Data Pattern to PCI**: A data pattern is written into the DMA FIFO and then written to the PCI bus. The Pattern Generator will attempt to fill the DMA FIFO at the same time that the DMA PCI Master will attempt to empty the DMA FIFO. The data pattern may either be a fixed pattern or an incrementing pattern.

Accesses to the PowerPC bus share some of the same attribute options that are offered by the Inbound Translation Functions. For example, the CRI (Cache-line Read Invalidate) option may be used for PowerPC bus reads, and the CWF (Cache-line Write Flush) option may be used for PowerPC bus writes.

Accesses to the PCI bus are under complete user control. The user can specify the actual PCI command to be used during both read and write cycles. This means that a DMA transfer can take place in either Memory, IO, or Configuration space. The DMA PCI Master has been optimized for accesses to PCI Memory space. Transfers to either IO or Configuration space may result in a protocol violation (i.e. bursting into IO space is discouraged, and in some cases IO addressing during single beat transfers may be incorrect) It is the user's responsibility to select the appropriate command type for each transfer.

When accessing the PCI bus, it is possible to fix the source and/or destination address. This option exists to support fixed address FIFO type devices. The fixed address option has some limitations on transfer addresses and sizes.

Addressing and Transfer Sizes

There are no restrictions on either addressing or transfer sizes. There are no dependencies between the source address and the destination address. The source address can be at any byte, and the destination address can be at any other byte. The transfer size can be as small as 1 byte and as large as 4GBytes.

Data Patterns

The DMA Controller has the option of writing data patterns to either PowerPC or PCI space. Any size transfer may be used, and there are no restrictions on the starting address. A starting data pattern is supplied by software. The DMA Controller can be programmed to work in terms of 8-bit patterns or 32-bit patterns. Software may also specify whether the pattern should be static or incrementing. The figure below shows some examples of pattern writes.

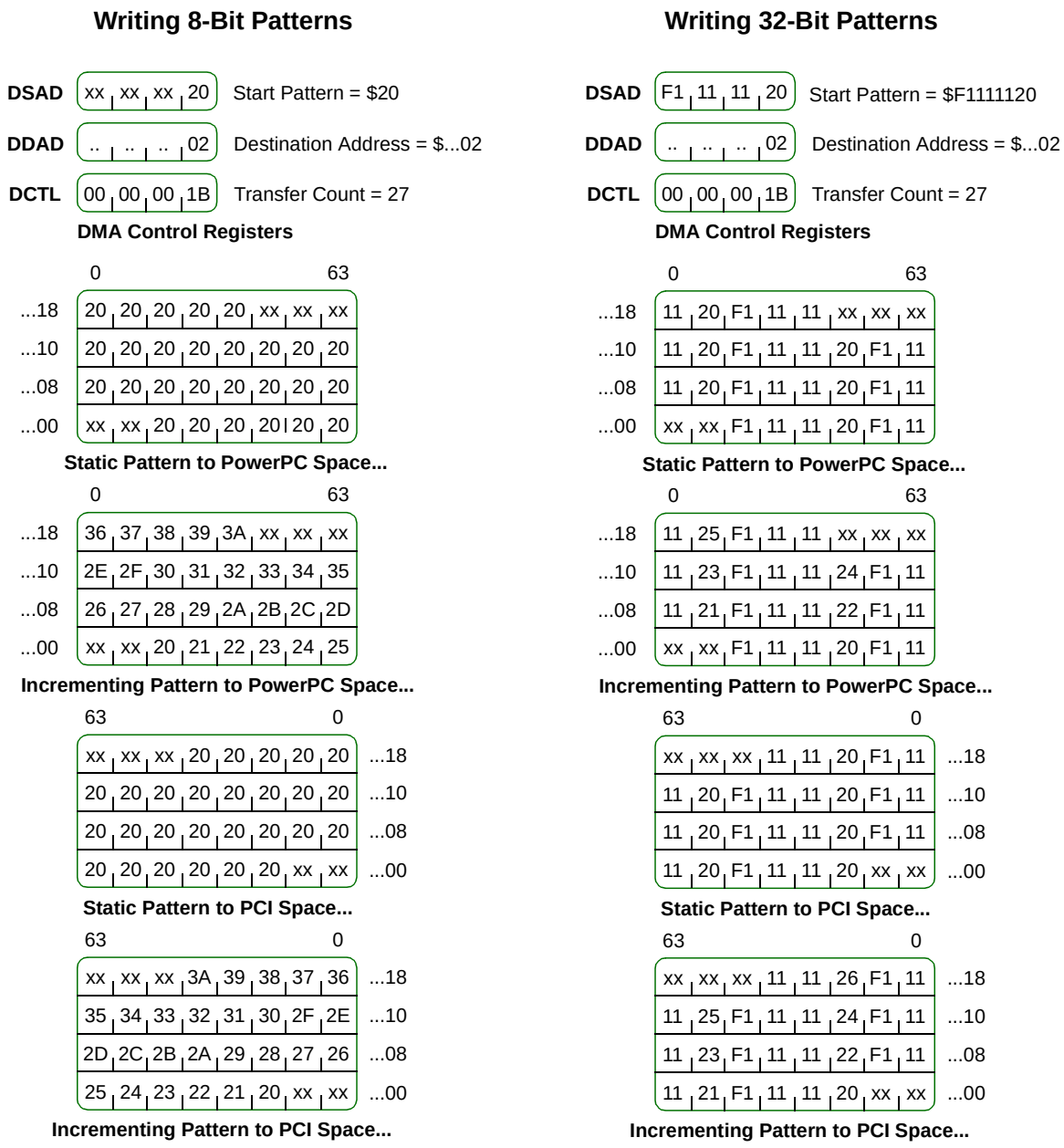


Figure 2-14. Examples of Pattern Writes

There are two issues associated with writing 32-bit patterns. When writing to PCI space, the pattern will not be subjected to Endian byte swapping. In addition, a transfer count that is not an even multiple of four will result in the rounding off of the *last* data pattern written to either PowerPC space or PCI space. The rounding off will occur on the pattern according to the address space being written to. A pattern written to PCI space will be rounded off starting from the left (or MSB) side of the pattern. A pattern written to PowerPC space will be rounded off starting from the right (or LSB) side of the pattern.

Linked-List Descriptors

The DMA PPC Master is responsible for fetching descriptors from PowerPC address space when using the Linked-List Mode. Each descriptor consumes 32 bytes and must be cache-line aligned. This cache-line structure helps minimize PowerPC bus bandwidth used when fetching descriptors.

The table below shows the format of a descriptor.

Table 2-8. DMA Controller Linked-List Descriptors

Offset	Bits					
	0		31	32		63
$00	DSAD			DSAT		
$08	DDAD			DDAT		
$10	DNLA			DCNT		
$18						

Each field in the descriptor corresponds to a DMA control register. When a descriptor is loaded by the DMA Controller, each field is placed into it's corresponding DMA control register. A complete description of each control register may be found in the section titled *Bridge PowerPC Control and Status Register* on page 3-41.

The descriptors are linked together by the **DNLA** register (i.e. the DNLA field within a descriptor). This field contains the address within PowerPC address space where the next descriptor may be found. The LLA field (Last Link-descriptor Address) within the **DNLA** indicates that this is the last descriptor.

Descriptors will not be prefetched by the DMA PPC Master. A Linked-List Mode transaction will be started by the DMA PPC Master reading one descriptor. The DMA Controller will then perform the transfer associated with that descriptor. If there are more descriptors to be executed, then the fetching of the next descriptor will not occur until the current transfer has completed.

Transfer Termination

There are four ways that DMA activity may be terminated:

1. **Transfer or Transaction Completion**: In most cases a Direct Mode transfer or a Linked-List Mode transaction will simply finish without intervention or error. When in Direct Mode, the end of the transfer is considered completion. When in Linked-List Mode, the end of the last transfer of a transaction is considered completion. Upon completion, the DMA Controller will return a “done” status to the **DSTA** register and optionally interrupt the processor.

2. **Commanded Stop**: This is an option available during Linked-List Mode transactions. Software may set the **DCTL.**PAU bit at any time during a transaction. When the DMA Controller reaches a transfer boundary (i.e. ready to fetch the next descriptor), it will stop all DMA activity. If there are more Linked-List transfers to be performed, then the DMA Controller will return a “paused” status to the **DSTA** register and optionally interrupt the processor. If the last transfer of a transaction has completed, then the DMA Controller will return a “done” status to the **DSTA** register.

   Once stopped, a transaction may started again at any time. The DMA Controller will simply pick up where it left off. The first descriptor fetch will occur from the address that was placed within the **DNLA** register during the previously completed transfer.

3. **Commanded Abort**: This option exists for either Direct Mode or Linked-List Mode. Software may set the **DCTL**.ABT bit at any time. When set, the DMA Controller will abort all DMA activity. This is considered a non-recoverable termination, and it will take affect almost immediately after the bit has been set. If the commanded abort took affect before the completion of a DMA transaction, then the DMA Controller will return an “abort” status to the **DSTA** register and will optionally interrupt the processor. If the transaction completed before the commanded abort took affect, then the DMA Controller will return a “done” status to the **DSTA** register.

4. **Detected Error Abort**: The DMA Controller is sensitive to the following system errors:
   - DMA PCI Master received a master abort
   - DMA PCI Master received a target abort
   - DMA PCI Master exceeds maximum retry count
   - DMA PPC Master obtained an address bus time-out

If any of these conditions are encountered, the DMA controller will abort all DMA activity. This is considered a non-recoverable termination, and it will take affect almost immediately after the condition has been detected. Once all DMA activity has ceased, the DMA Controller will return the appropriate error status to the **DSTA** register and will optionally interrupt the processor.

The **DSTA** register only indicates that an error has been detected and the DMA transaction was aborted. Further details associated with the error may be found within the Error Diagnostics function. Refer to the section titled *Error Diagnostics* on page 2-127 for more details.

Interrupts

There is only one interrupt generated by the DMA Controller. Status on this interrupt may be obtained from within the Functional Exception Status (**FEST**) register, and masking is supported within the Functional Exception Mask (**FEMA**) register. The DMA interrupt may be cleared

from within the Functional Exception Clear (**FECL**) register. An interrupt will be generated anytime the DMA Controller changes from a busy state to a done, paused, or aborted state.

Transfer Throttling

It is possible that the PCI or PowerPC bandwidth consumed by the DMA Controller could swamp the system with DMA activity. This is particularly a problem if PCI or PowerPC arbitration is in favor of the Harrier.

The PCI bus has a built-in system check on excessive bandwidth consumption. The Master Latency Timer will be a driving factor for how much PCI bandwidth a DMA transfer may take. In addition, the PCI Back-off Timer may be used to further control bandwidth consumption. The PCI Back-off Timer determines how long the DMA PCI Master will wait between starting burst transfers when a transfer is interrupted by the PCI Master Latency Timer. This timer is controlled by the PBT field within the **DCTL** register. Note that if the Master Latency Timer does not expire, then the DMA PCI Master will attempt to complete a DMA transfer as one continuous PCI burst in accordance with the abilities of the DMA FIFO.

Bandwidth consumption on the PowerPC bus is controlled by the transfer characteristics of the DMA PPC Master. The DMA PPC Master will attempt to transfer data in user programmable block sizes of 256 bytes, 512 bytes, 1024 bytes, or continuous. The DMA PPC Master relies on the Harrier latching request arbitration protocol. A transfer will proceed with the PowerPC bus request asserted until the programmable transfer limit is reached. Once this limit is reached, the DMA PPC Master will remove its request and wait for two clock periods after the completion of the last address tenure. This short back-off time will allow other pending PowerPC masters a chance at obtaining the bus. After the two clock period expires, the DMA PPC Master will reassert its request if there is more data to be transferred.

2

Message Passing

The Harrier incorporates hardware dedicated to supporting message passing. There are two levels of support offered; I_2O Message Passing and Generic Message Passing.

I_2O Message Passing

The *Intelligent I/O (I_2O) Architecture Specification Version 1.5* describes a comprehensive hardware and software solution to managing the development of platform independent device drivers. The specification advocates the use of distributed embedded I/O processing and uses a system of message passing to move information between processes.

An I_2O system consists of a Host and at least one I/O Platform (IOP). Predefined messages may be passed from the Host to an IOP, from an IOP to the Host, and from an IOP to an IOP. These messages, called Message Frames, are a minimum of 64 bytes long and reside within shared memory structures. A Message Frame is identified by a Message Frame Address (MFA). The MFA points to the beginning address of a Message Frame within shared memory.

The MFA exists in one of two states. A *Free* MFA is a pointer to an allocated Message Frame that is “empty”. A *Post* MFA is a pointer to an allocated Message Frame that is “full”.

An IOP will allocate a set of free Inbound Message Frames and corresponding MFAs within it’s shared local memory. These are used to receive messages from the Host or other IOPs. The IOP will also allocate a set of free Outbound Message Frames and corresponding MFAs within it’ s shared local memory. These are used to send messages to the Host or other IOPs.

A message is passed from one agent to another by sending the recipient a post MFA. The sequence for sending a post MFA is as follows:

- The sending agent retrieves a free MFA from the receiving agent. This MFA points to an “empty” Message Frame in the receiving agents local memory.

- The sending agent writes the Message Frame into the receiving agents local memory pointed to by the MFA.
- The MFA now represents a Message Frame that is “full”. The sending agent sends the post MFA back to the receiving agent. This will generate an interrupt to the receiving agent.
- The receiving agent recognizes the post MFA and processes the new Message Frame. Once complete, the receiving agent allocates the MFA as free.

IOP Message Unit

The Harrier can participate in the I_2O protocol as either a Host or an IOP. Participation as a Host requires no additional hardware. Participation as an IOP requires an IOP Message Unit (IMU). The Harrier provides an IMU that is fully compatible with the I_2O specification. The Harrier shows the key elements of the IMU from the system perspective.

The FIFOs for the inbound and outbound queues and the Message Frames physically reside within local memory. The Harrier implements all of the pointer registers, the inbound and outbound queue ports, and the interrupt structure in hardware.

The inbound queue is accessed from PCI through the Message Passing I_2O Inbound Queue (**MIIQ**) register. The outbound queue is accessed from PCI through the Message Passing I_2O Outbound Queue (**MIOQ**) register. The **MIIQ** and **MIOQ** registers reside within a relocatable 4K byte block of PCI Memory space called the *PMEP* (PCI Message Passing) Register Group. This block is located into PCI memory space using the **MPBAR** register within the *PCFS* (PCI Configuration Space) Register Group. All of these components are fully compliant with the I_2O specification. Please refer to the section titled *PowerPC to PCI Bridge* on page 3-38 for more information.

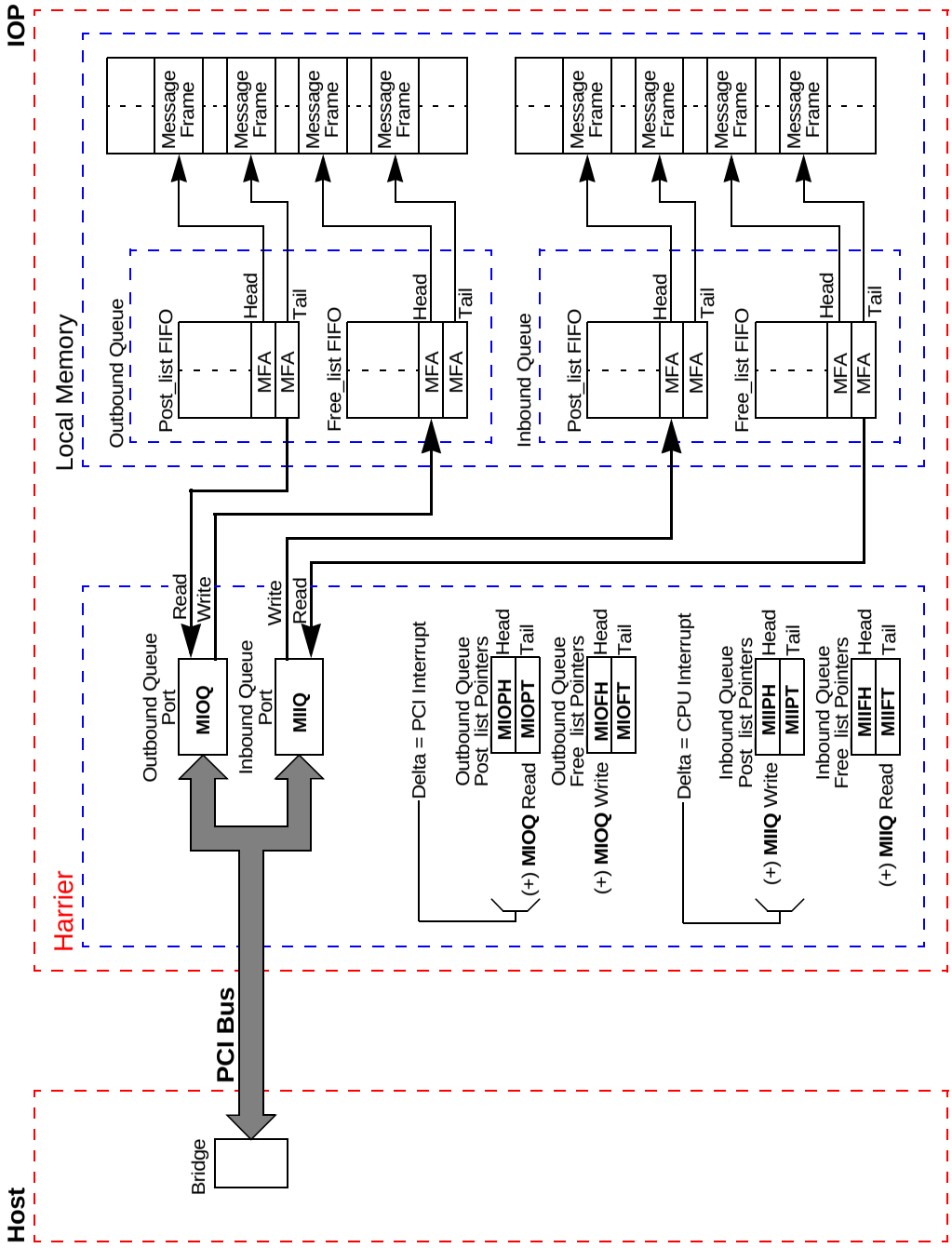


Figure 2-15. IOP Message Unit

All of the pointer registers are located within the *XCSR* Register Group. These registers represent the head and tail pointers into PowerPC address space for the inbound and outbound Free_list and Post_list FIFOs. Some of these registers are automatically updated by certain actions taken to the inbound and outbound ports. For example, a write by a PCI master to the **MIIQ** register will automatically increment the Inbound Post_list Head (**MIIPH**) register. The automatic actions taken for each pointer register are designated by a (+) symbol in the previous figure. Those registers that do not have the (+) symbol next to it must be manually maintained by the processor.

IMU Queue Structure

In order for hardware to correctly maintain the pointers, the FIFO sizes and locations must be known. The Message Passing I_2O Queue Base Address (**MIQB**) register within the *XCSR* Register Group is used to specify the base address within PowerPC address space of the entire inbound and outbound queue structure. This is a 1MB structure locatable only on 1MB boundaries. Within this structure is four FIFOs. The base address of each FIFO is on one of four 256 KByte boundaries as shown in the following figure.

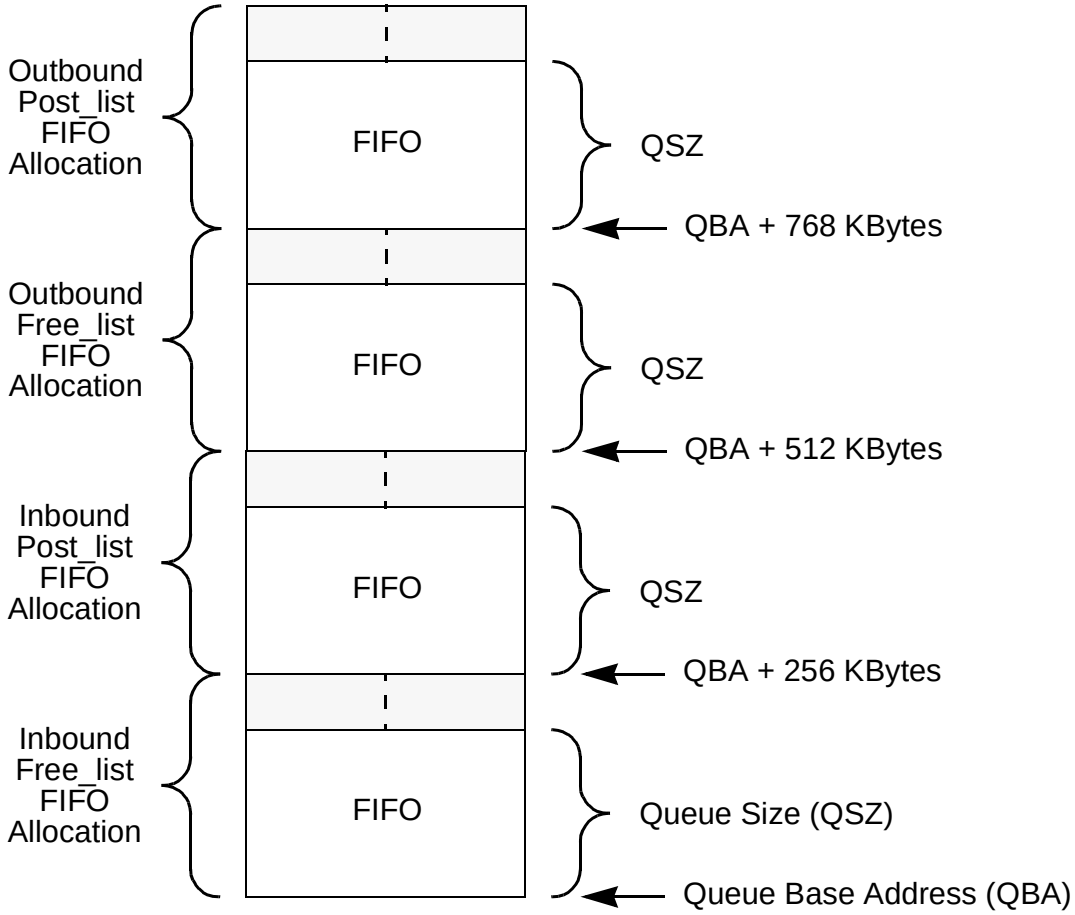


Figure 2-16. IMU Queue Structure

Each FIFO is the same size. The size of the FIFOs is programmable through the QSZ field within the Message Passing I_2O Control (**MICT**) register.

IMU Enable

The I_2O specification states that an IMU must be optionally enabled or disabled. By default, the IMU comes up out of reset in the disabled state. The processor must first setup all the queue sizes and interrupt logic. Then the processor must allocate some Message Frames and MFAs. The

Inbound Free_list FIFO must be filled with these MFAs. Once the processor is ready, the IMU is enabled. This is done by setting the ENA bit within the **MICT** register.

While the IMU is disabled, the I_2O specification states that any accesses to the Queue ports (**MIIQ** and **MIOQ**) must not affect the IMU. Writes to these ports will be discarded. Reads from these ports will return all FFFF_FFFF. The head and tail pointers will not be affected.

IMU Interrupts

I_2O is an interrupt driven protocol. As a Host, the Harrier must be able to recognize and respond to PCI interrupts. This is typically handled by routing the PCI interrupts through the MPIC. As an IOP, the Harrier must be able to generate a PCI interrupt. A PCI interrupt will be generated whenever there is a difference between the Outbound Post_list Queue head and tail pointers. This is an indication to the Host that there is a “full” MFA that needs servicing within the IOP’s outbound queue. The interrupt will remain asserted until the Host reads all of the outbound MFA entries.

A Host may obtain interrupt status by reading the Message Passing I_2O Interrupt Status (**MIST**) register. The Host may also control the masking of this interrupt through the Message Passing I_2O Mask (**MIMS**) register. Both of these registers reside within the *PMEP* Register Group.

The IMU can also generate a processor interrupt. This interrupt is routed to the MPIC and is asserted whenever there is a difference between the head and tail pointers for the Inbound Post_list Queue. This interrupt is an indication that the Host has placed a “full” MFA within the IOP’s inbound queue. The interrupt will remain asserted until the processor adjusts the head and tail pointers to match. The processor may obtain interrupt status by reading the Functional Exception Status (**FEST**) register within the *XCSR* Register Group. The processor can mask this interrupt through the Functional Exception Mask (**FEMA**) register.

IOP Agent Identification

The I_2O specification has identified a special PCI class code for I_2O IOP Agents. The class code is represented with the read-only **CLAS** register within the *PCFS* Register Group. The Harrier must present different class

codes depending on the level of I_2O participation. The Harrier must present itself as a “Bridge Device” if not participating in I_2O, or if it is participating as a I_2O Host. If acting as an IOP, then the Harrier must present itself as a “I_2O Controller”. Please refer to the section titled *Revision ID/Class Code Registers* on page 3-58 for more information on class codes.

The selection of a class code depends on the absence or existence of an external resistor during the release of reset. Please refer to the section titled *Hardware Configuration* on page 2-133 in this section for more information.

Generic Message Passing

The Harrier provides hardware that supports a more generic form of message passing. This hardware is not related to the I_2O Message Passing hardware. The support offered is in the form of Doorbell registers and Message Passing registers.

Doorbell Registers

The Harrier has two Doorbell registers. One register is for inbound traffic and one is for outbound traffic. A Doorbell register consists of an array of bits that may be individually set by a sending agent. The receiving agent receives an interrupt anytime a single doorbell bit is set. The receiving agent may scan the Doorbell register to determine which “doorbell” was set, and will write a one to the corresponding doorbell bit to clear the associated interrupt.

Inbound traffic uses the Message Passing Generic Inbound Doorbell (**MGID**) register. This register is used when a PCI master wishes to assert a doorbell interrupt to the processor. There are a total of twenty-eight interrupt doorbells. A PCI master may access this register from within the *PMEP* Register Group, and the processor may access this register from within the *XCSR* Register Group.

The processor may obtain status pertaining to an interrupt by reading the **FEST** register, and may mask the interrupt within the **FEMA** register.

Outbound traffic will use the Message Passing Generic Outbound Doorbell (**MGOD**) register. This register is used when the processor wishes to assert a doorbell interrupt to a PCI master. There are a total of twenty-eight interrupt doorbells. The processor may access this register from within the *XCSR* Register Group, and a PCI master may access this register from within the *PMEP* Register Group.

A PCI master may obtain status pertaining to an interrupt by reading the **MGOD** register. The masking of the interrupt is through the Message Passing Generic Interrupt Mask (**MGMS**) register

Message Passing Registers

The Harrier has four Message Passing registers. Two of the registers are used for inbound traffic and two are used for outbound traffic. A Message Passing register is a 32-bit location that may be written with a unique 32-bit message by a sending agent. The receiving agent will receive an interrupt whenever a new message has been written, and will read the Message Passing register to obtain the message.

Inbound traffic will use the Message Passing Generic Inbound Message (**MGIM0** and **MGIM1**) registers. These registers are used when a PCI master wishes to send a message to the processor. A PCI master may access this register from within the *PMEP* Register Group, and the processor may access this register from within the *XCSR* Register Group.

The processor may obtain status pertaining to either interrupt by reading the **FEST** register, and may individually mask each interrupt from within the **FEMA** register.

Outbound traffic will use the Message Passing Generic Outbound Message (**MGOM0** and **MGOM1**) registers. These registers are used when the processor wishes to send a message to a PCI master. The processor may access this register from within the *XCSR* Register Group, and a PCI master may access this register from within the *PMEP* Register Group.

A PCI master may obtain status pertaining to an interrupt by reading the Message Passing Generic Interrupt Status (**MGST**) register. The interrupts may be individually masked within the Message Passing Generic Interrupt Mask (**MGMS**) register.

2

Multiprocessor Interrupt Controller (MPIC)

The MPIC is a multi-processor structured intelligent interrupt controller. Its functions are explained in the following subsections.

MPIC Features

- MPIC programming model
- Supports two processors
- Supports 16 external interrupts
- Supports 15 programmable Interrupt & Processor Task priority levels
- Supports the connection of an external 8259 for ISA/AT compatibility
- Distributed interrupt delivery for external I/O interrupts
- Direct/Multicast interrupt delivery for Interprocessor and timer interrupts
- Four Interprocessor Interrupt sources
- Four timers
- Processor initialization control

Architecture

The Harrier implements an address decoder for placing the MPIC registers in PowerPC address space. Access to these registers require PowerPC bus mastership. These accesses include interrupt and timer initialization and interrupt vector reads. The MPIC is run from PowerPC clock.

The MPIC receives interrupt inputs from 16 external sources, four interprocessor sources, four timer sources, and two Harrier internal source interrupts (the Harrier internal functional exception and the Harrier internal error exception). The externally sourced interrupts 1 through 15

have two modes of activation; low level or active high positive edge. External interrupt 0 can be either level or edge activated with either polarity. The Harrier internal interrupt requests are active low level sensitive interrupts. The Interprocessor and timer interrupts are event activated.

When the MPIC is in the 8259 pass-through mode and if the ***XCSR***.**MCSR**.OPI bit is set, interrupts from external source number 0 are inverted and passed directly to processor 0 interrupt pin (IRQ0_).

When the MPIC is out of 8259 pass-through mode and if the ***XCSR***.**MCSR**.OPI bit is set, the Harrier internal interrupts are passed on to the MPIC. If the **MCSR**.OPI bit is cleared the Harrier internal interrupts are passed directly to the processor 0 interrupt pin (IRQ0_).

CSR's Readability

Unless explicitly specified, all registers are readable and return the last value written. The exceptions are the IPI dispatch registers and the EOI registers which return zeros on reads, the interrupt source ACT bit which returns current interrupt source status, the interrupt acknowledge register which returns the vector of the highest priority interrupt which is currently pending, and reserved bits which returns zeros. The interrupt acknowledge register is also the only register which exhibits any read side-effects.

Interrupt Source Priority

Each interrupt source is assigned a priority value in the range from 0 to 15 where 15 is the highest. In order for delivery of an interrupt to take place the priority of the source must be greater than that of the destination processor. Therefore setting a source priority to zero inhibits that interrupt.

Processor's Current Task Priority

Each processor has a current task priority register which is set by system software to indicate the relative importance of the task running on that processor. The processor will not receive interrupts with a priority level

equal to or lower than its current task priority. Therefore setting the current task priority to 15 prohibits the delivery of all interrupts to the associated processor.

Nesting of Interrupt Events

A processor is guaranteed never to have an in service interrupt preempted by an equal or lower priority source. An interrupt is considered to be in service from the time its vector is returned during an interrupt acknowledge cycle until an End of Interrupt (EOI) is received for that interrupt. The EOI cycle indicates the end of processing for the highest priority in service interrupt.

Spurious Vector Generation

Under certain circumstances the MPIC will not have a valid vector to return to the processor during an interrupt acknowledge cycle. In these cases the spurious vector from the spurious vector register is returned. The following cases cause a spurious vector fetch.

- IRQ_ is asserted in response to an externally sourced interrupt which is activated with level sensitive logic and the asserted level is negated before the interrupt is acknowledged.
- IRQ_ is asserted for an interrupt source which is masked using the mask bit in the Vector-Priority register before the interrupt is acknowledged.

Interprocessor Interrupts (IPI)

Processor 0 and 1 can generate interrupts which are targeted for the other processor or both processors. There are four Interprocessor Interrupts (IPI) channels. The interrupts are initiated by writing a bit in the IPI dispatch registers. If subsequent IPI’s are initiated before the first is acknowledged, only one IPI will be generated. The IPI channels deliver interrupts in the Direct Mode and can be directed to more than one processor.

8259 Compatibility

The MPIC provides a mechanism to support PC-AT compatible chip sets using the 8259 interrupt controller architecture. After power on reset, the MPIC defaults to 8259 pass-through mode. In this mode, if the **MCSR**.OPI bit is set, interrupts from external source number 0 (the interrupt signal from the 8259 is connected to this external interrupt source on the MPIC) are inverted and passed directly to processor 0 (IRQ0_). If the pass-through mode is disabled and the **MCSR**.OPI bit is set, the 8259 interrupts are delivered using the priority and distribution mechanisms of the MPIC.

MPIC does not interact with the vector fetch from the 8259 interrupt controller.

Harrier Internal Functional Interrupt

Functional exceptions generated by the Harrier internal modules (DMA, message unit, abort switch and UARTs) are grouped together and sent to the MPIC interrupt logic as a singular interrupt source (the Harrier Internal Functional Interrupt). Please see the section titled *Exceptions* on page 2-122 for more information about the Harrier's functional exceptions. This Harrier internal functional interrupt request is an active low level sensitive interrupt. The interrupt delivery mode for this interrupt is distributed.

Harrier Internal Error Interrupt

The Harrier detected error exceptions are grouped together and sent to the MPIC interrupt logic as a singular interrupt source (Harrier Internal Error Interrupt). This Harrier internal error interrupt request is an active low level sensitive interrupt. The interrupt delivery mode for this interrupt is distributed.

When the **MCSR**.OPI bit is cleared the Harrier internal functional interrupt and the Harrier internal error interrupt are combined and directly passed on to IRQ0_ pin.

For system implementations where the MPIC controller is not used, the combined Harrier Internal Interrupt is made available by a signal that is external to Harrier. Presumably this signal would be connected to an

externally sourced interrupt input of an MPIC controller in a different device. Since the MPIC specification defines external I/O interrupts to operate in the distributed mode, the delivery mode of this interrupt should be consistent.

Timers

A divide-by-eight pre-scaler is synchronized to the PowerPC bus clock. The output of the prescaler enables the decrement of the four timers. The timers may be used for system timing or to generate periodic interrupts. Each timer has four registers which are used for configuration and control. They are:

- Current Count Register
- Base Count Register
- Vector-Priority Register
- Destination Register

Interrupt Delivery Modes

The direct and distributed interrupt delivery modes are supported. Note that the direct deliver mode has sub modes of multicast or non-multicast. The Inter Processor interrupts and Timer interrupts operate in the direct delivery mode. The externally sourced or I/O interrupts operate in the distributed mode.

In the direct delivery mode, the interrupt is directed to one or both processors. If it is directed to two processors (i.e. multicast), it will be delivered to two processors. The interrupt is delivered to the processor when the priority of the interrupt is greater than the priority contained in the task register for that processor, and when the priority of the interrupt is greater than any interrupt which is in-service for that processor. An interrupt is considered to be in service from the time its vector is returned during an interrupt acknowledge cycle until an EOI is received for that interrupt. The EOI cycle indicates the end of processing for the highest priority in service interrupt.

In the distributed delivery mode, the interrupt is pointed to one or more processors but it will be delivered to only one processor. Therefore, for externally sourced or I/O interrupts, multicast delivery is not supported. The interrupt is delivered to a processor when the priority of the interrupt is greater than the priority contained in the task register for that processor, and when the priority of the interrupt is greater than any interrupt which is in-service for that processor, and when the priority of that interrupt is the highest of all interrupts pending for that processor, and when that interrupt is not in-service for the other processor. If both destination bits are set for each processor, the interrupt will be delivered to the processor that has a lower task register priority. Note, due to a deadlock condition that can occur when the task register priorities for each processor are the same and both processors are targeted for interrupt delivery, the interrupt will be delivered to processor 0 or processor 1 as determined by the TIE mode. Additionally, If priorities are set the same for competing interrupts, external int. 0 is given the highest priority in hardware followed by external int.1 through 15 and then followed by timer 0 through timer 3 and followed by IPI 0 and 1. For example, if both ext0 and ext1 interrupts are pending with the same assigned priority; during the following interrupt acknowledge cycles, the first vector returned shall be that of ext0 and then ext1. This is an arbitrary choice.

Block Diagram Description

The description of the block diagram shown in Figure 2-17 on page 2-74 focuses on the theory of operation for the interrupt delivery logic. If the preceding section is a satisfactory description of the interrupt delivery modes and the reader is not interested the logic implementation, this section can be skipped.

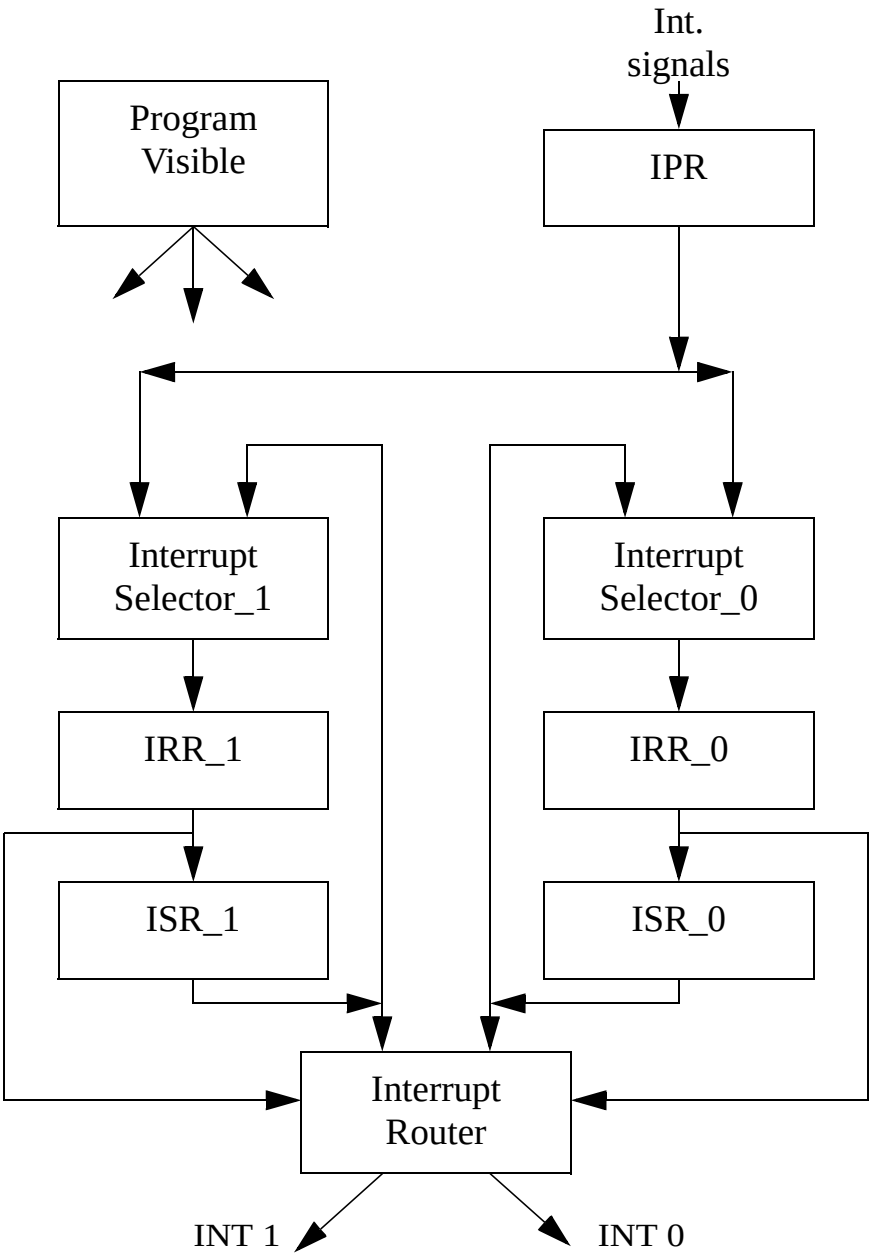


Figure 2-17. MPIC Block Diagram

Program Visible Registers

These are the registers that software can access. They are described in detail in the MPIC Register section.

Interrupt Pending Register (IPR)

The interrupt signals to MPIC are qualified and synchronized to the clock by the IPR. If the interrupt source is internal to the Harrier or external with their Sense bit = 0 (edge sensitive), a bit is set in the IPR. That bit is cleared when the interrupt associated with that bit is acknowledged. If the interrupt source is external and level activated, the output from the IPR is not negated until the level into the IPR is negated.

Externally sourced interrupts are qualified based upon their Sense and/or Poll bits in the Vector-Priority register. IPI and Timer Interrupts are generated internally to the Harrier and are qualified by their Destination bit. Since the internally generated interrupts use direct delivery mode with multicast capability, there are two bits in the IPR, one for each processor, associated with each IPI and Timer interrupt source.

The MASK bits from the Vector-Priority registers is used to qualify the output of the IPR. Therefore, if an interrupt condition is detected when the MASK bit is set, that interrupt will be requested when the MASK bit is lowered.

Interrupt Selector (IS)

There is a Interrupt Selector (IS) for each processor. The IS receives interrupt requests from the IPR. If the interrupt requests are from an external source, they are qualified by the destination bit for that interrupt and processor. If they are from an internal source, they have been qualified. The output of the IS will be the highest priority interrupt that has been qualified. This output is the priority of the selected interrupt and its source identification. The IS will resolve an interrupt request in two PowerPC clock ticks.

The IS also receives a second set of inputs from the ISR. During the End Of Interrupt cycle, these inputs are used to select which bits are to be cleared in the ISR.

Interrupt Request Register (IRR)

There is a Interrupt Request Register (IRR) for each processor. The IRR always passes the output of the IS except during Interrupt Acknowledge cycles. This guarantees that the vector which is read from the Interrupt Acknowledge Register is not changing due to the arrival of a higher priority interrupt. The IRR also serves as a pipeline register for the two tick propagation time through the IS.

In-Service Register (ISR)

There is a In-Service Register (ISR) for each processor. The contents of the ISR is the priority and source of all interrupts which are in-service. The ISR receives a bit-set command during Interrupt Acknowledge cycles and a bit-clear command during End Of Interrupt cycles.

The ISR is implemented as a 41 bit register with individual bit set and clear functions. Fifteen bits are used to store the priority level of each interrupt which is in-service. Twenty-six bits are used to store the source identification of each interrupt which is in service. Therefore there is one bit for each possible interrupt priority and one bit for each possible interrupt source.

Interrupt Router

The Interrupt Router monitors the outputs from the ISR's, Current Task Priority Registers, Destination Registers, and the IRR's to determine when to assert a processor's IRQ_ pin.

When considering the following rule sets, it is important to remember that there are two types of inputs to the Interrupt Selectors. If the interrupt is a distributed class interrupt, there is a single bit in the IPR associated with this interrupt and it is delivered to both Interrupt Selectors. This IPR bit is qualified by the destination register contents for that interrupt before the Interrupt Selector compares its priority to the priority of all other requesting interrupts for that processor. If the interrupt is programmed to be edge sensitive, the IPR bit is cleared when the vector for that interrupt is returned when the Interrupt Acknowledge register is examined. On the other hand, if the interrupt is a direct/multicast class interrupt, there are two bits in the IPR associated with this interrupt. One bit for each processor.

Then one of these bits are delivered to each Interrupt Selector. Since this interrupt source can be multicast, each of these IPR bits must be cleared separately when the vector is returned for that interrupt to a particular processor.

If one of the following sets of conditions are true, the interrupt pin for processor 0 (IRQ0_) is driven active.

- Set1
  - The source ID in IRR_0 is from an external source.
  - The destination bit for processor 1 is a 0 for this interrupt.
  - The priority from IRR_0 is greater than the highest priority in ISR_0.
  - The priority from IRR_0 is greater than the contents of task register_0.
- Set2
  - The source ID in IRR_0 is from an external source.
  - The destination bit for processor 1 is a 1 for this interrupt.
  - The source ID in IRR_0 is not present is ISR_1.
  - The priority from IRR_0 is greater than the highest priority in ISR_0.
  - The priority from IRR_0 is greater than the Task Register_0 contents.
  - The contents of Task Register_0 is less than the contents of Task Register_1.
- Set3
  - The source ID in IRR_0 is from an internal source.
  - The priority from IRR_0 is greater than the highest priority in ISR_0.
  - The priority from IRR_0 is greater than the Task Register_0 contents.

There is a possibility for a priority tie between the two processors when resolving external interrupts. In that case, the interrupt will be delivered to processor 0 or processor 1 as determined by the TIE mode bit. This case is not defined in the above rule set.

Programming Notes

The following subsections discuss Programming Notes that are specific to the MPIC portion of the Harrier.

External Interrupt Service

The following summarizes how an external interrupt is serviced:

- An external interrupt occurs.
- The processor state is saved in the machine status save/restore registers. A new value is loaded into the Machine State Register(MSR). The External Interrupt Enable bit in the new MSR (MSRee) is set to zero. Control is transferred to the O/S external interrupt handler.
- The external interrupt handler calculates the address of the Interrupt Acknowledge register for this processor (MPIC Base Address + 0x200A0 + (processor ID shifted left 12 bits)).
- The external interrupt handler issues an Interrupt Acknowledge request to read the interrupt vector from the MPIC. If the interrupt vector indicates the interrupt source is the 8259, the interrupt handler issues a second Interrupt Acknowledge request to read the interrupt vector from the 8259. The MPIC does not interact with the vector fetch from the 8259.
- The interrupt handler saves the processor state and other interrupt-specific information in system memory and re-enables for external interrupts (the MSRee bit is set to 1). The MPIC blocks interrupts from sources with equal or lower priority until an End-of-Interrupt is received for that interrupt source. Interrupts from higher priority interrupt sources continue to be enabled. If the interrupt source was the 8259, the interrupt handler issues an EOI request to the MPIC.

This resets the In-Service bit for the 8259 with in the MPIC and allows it to recognize higher priority interrupt requests, if any, from the 8259. If none of the nested interrupt modes of the 8259 are enabled, the interrupt handler issues an EOI request to the 8259.

- The device driver interrupt service routine associated with this interrupt vector is invoked.
- If the interrupt source was not the 8259, the interrupt handler issues an EOI request for this interrupt vector to the MPIC. If the interrupt source was the 8259 and any of the nested interrupt modes of the 8259 are enabled, the interrupt handler issues an EOI request to the 8259.

Normally, interrupts from ISA devices are connected to the 8259 interrupt controller. ISA devices typically rely on the 8259 Interrupt Acknowledge to flush buffers between the ISA device and system memory. If interrupts from ISA devices are directly connected to the MPIC (bypassing the 8259), the device driver interrupt service routine must read status from the ISA device to ensure buffers between the device and system memory are flushed.

Reset State

After power on reset the MPIC state is:

- Current task priority for all CPU's set to 15.
- All interrupt source priorities set to zero.
- All interrupt source mask bits set to a one.
- All interrupt source activity bits cleared.
- Processor Init Register is cleared.
- All counters stopped and interrupts disabled.
- Controller mode set to 8259 pass-through.

2

Operation

The following subsections describe the operational characteristics of the MPIC.

Interprocessor Interrupts

Four interprocessor interrupt (IPI) channels are provided for use by all processors. During system initialization the IPI vector/priority registers for each channel should be programmed to set the priority and vector returned for each IPI event. During system operation a processor may generate an IPI by writing a destination mask to one of the IPI dispatch registers. Note that each IPI dispatch register is shared by both processors. Each IPI dispatch register has two addresses but they are shared by both processors. That is there is a total of four IPI dispatch registers in the MPIC.

The IPI mechanism may be used for self interrupts by programming the dispatch register with the bit mask for the originating processor.

Dynamically Changing I/O Interrupt Configuration

The interrupt controller provides a mechanism for safely changing the vector, priority, or destination of I/O interrupt sources. This is provided to support systems which allow dynamic configuration of I/O devices. In order to change the vector, priority, or destination of an active interrupt source, the following sequence should be performed:

- Mask the source using the MASK bit in the vector/priority register.
- Wait for the activity bit (ACT) for that source to be cleared.
- Make the desired changes.
- Unmask the source.

This sequence ensures that the vector, priority, destination, and mask information remain valid until all processing of pending interrupts is complete.

EOI Register

Each processor has a private EOI register which is used to signal the end of processing for a particular interrupt event. If multiple nested interrupts are in service, the EOI command terminates the interrupt service of the highest priority source. Once an interrupt is acknowledged, only sources of higher priority will be allowed to interrupt the processor until the EOI command is received. This register should always be written with a value of zero which is the nonspecific EOI command.

Interrupt Acknowledge Register

Upon receipt of an interrupt signal, the processor may read this register to retrieve the vector of the interrupt source which caused the interrupt.

8259 Mode

The 8259 mode bits control the use of an external 8259 pair for PC--AT compatibility. Following reset this mode is set for pass through which essentially disables the advanced controller and passes an 8259 input on external interrupt source 0 directly through to processor zero interrupt pin (IRQ0_). During interrupt controller initialization this channel should be programmed for mixed mode in order to take advantage of the interrupt delivery modes.

Current Task Priority Level

Each processor has a separate Current Task Priority Level register. The system software uses this register to indicate the relative priority of the task running on the corresponding processor. The interrupt controller will not deliver an interrupt to a processor unless it has a priority level which is greater than the current task priority level of that processor. This value is also used in determining the destination for interrupts which are delivered using the distributed delivery mode.

2

I²C Interface

The Inter-Integrated Circuit (I²C) interface provides 2 two-wire *master-only* serial ports to support communication with slave I²C devices such as serial EEPROMs. The I²C interface is compatible with these devices, and the inclusion of a serial EEPROM in the memory subsystem may be desirable. The EEPROM could maintain the configuration information related to the memory subsystem even when the power is removed from the system. Each slave device connected to the I²C bus is software addressable by a unique address. The number of interfaces connected to the I²C bus is solely dependent on the bus capacitance limit of 400pF.

For I²C bus programming, the Harrier is the *only* master on the bus and the serial EEPROM devices are all slaves. The I²C bus supports 7-bit addressing mode and transmits data one byte at a time in a serial fashion with the most significant bit (MSB) being sent out first. Five registers are required to perform the I²C bus data transfer operations. These are the I²C Clock Prescaler (**I2PSx**) Register, I²C Control (**I2COx**) Register, I²C Status (**I2STx**) Register, I²C Transmitter Data (**I2TDx**) Register, and I²C Receiver Data (**I2RDx**) Register.

The I²C serial data (SDAx) is an open-drain bidirectional line on which data can be transferred at a rate up to 100 Kbits/s in the standard mode, or up to 400 kbits/s in the fast mode. The I²C serial clock (SCLx) is programmable via the **I2PSx** Register. The I²C clock frequency is determined by the following formula:

I²C CLOCK = SYSTEM CLOCK / (**I2PSx**+1) / 2

The I²C bus has the ability to perform byte write, page write, current address read, random read, and sequential read operations.

Byte Write

The **I2STx** Register contains the CMP bit which is used to indicate if the I²C master controller is ready to perform an operation. Therefore, the first step in the programming sequence should be to test the CMP bit for the operation-complete status. The next step is to initiate a start sequence by first setting the STA and ENA bits in the **I2COx** Register and then writing

the device address (bits 24-30) and write bit (bit 31=0) to the **I2TDx** Register. The CMP bit will be automatically clear with the write cycle to the **I2TDx** Register. The **I2STx** Register must now be polled to test the CMP and ACKI bits. The CMP bit becomes set when the device address and write bit have been transmitted, and the ACKI bit provides status as to whether or not a slave device acknowledged the device address. With the successful transmission of the device address, the word address will be loaded into the **I2TDx** Register to be transmitted to the slave device. Again, CMP and ACKI bits must be tested for proper response. After the word address is successfully transmitted, the next data loaded into the **I2TDx** Register will be transferred to the address location selected previously within the slave device. After CMP and ACKI bits have been tested for proper response, a stop sequence must be transmitted to the slave device by first setting the STP and ENA bits in the **I2COx** Register and then writing a dummy data (data=don't care) to the **I2TDx** Register. The **I2STx** Register must now be polled to test CMP bit for the operation-complete status. The stop sequence will initiate a programming cycle for the serial EEPROM and also relinquish the Harrier master's possession of the I^2C bus. The following figure shows the suggested software flow diagram for programming the I^2C byte write operation.

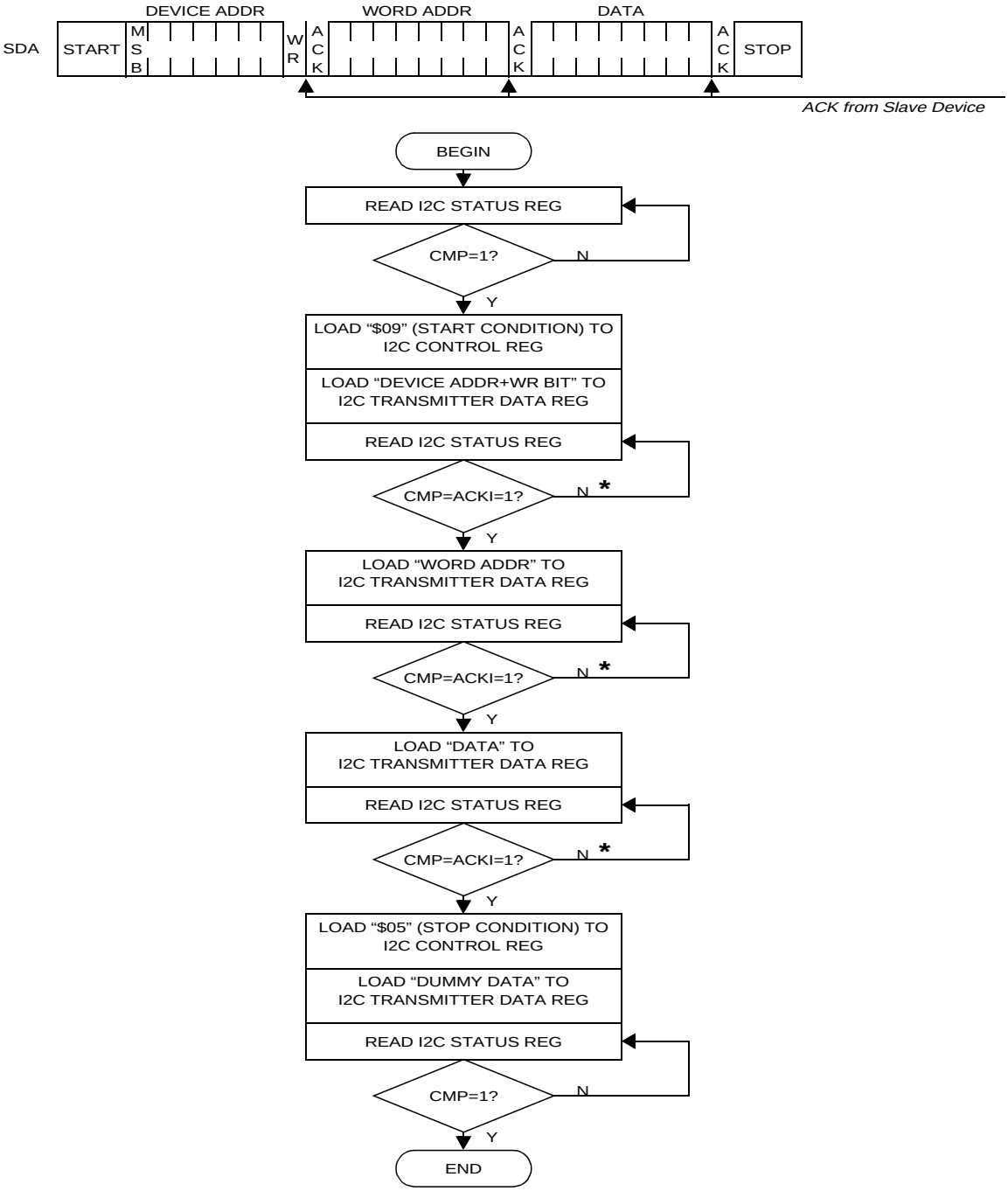
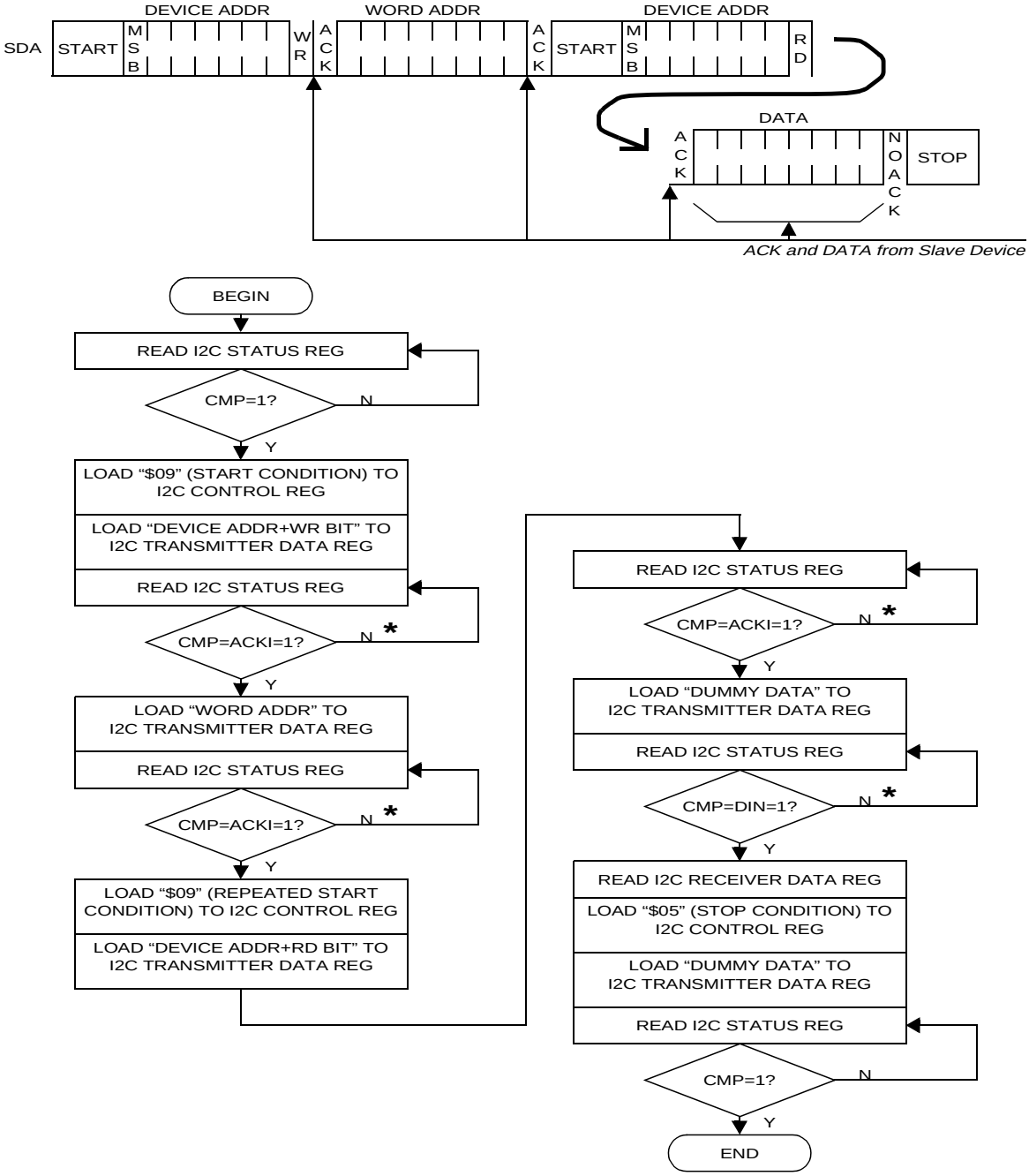


Figure 2-18. Programming Sequence for I²C Byte Write

Random Read

The I^2C random read begins in the same manner as the I^2C byte write. The first step in the programming sequence should be to test the CMP bit for the operation-complete status. The next step is to initiate a start sequence by first setting the STA and ENA bits in the **I2COx** Register and then writing the device address (bits 24-30) and write bit (bit 31=0) to the **I2TDx** Register. The CMP bit will be automatically clear with the write cycle to the **I2TDx** Register. The **I2STx** Register must now be polled to test the CMP and ACKI bits. The CMP bit becomes set when the device address and write bit have been transmitted, and the ACKI bit provides status as to whether or not a slave device acknowledged the device address. With the successful transmission of the device address, the word address will be loaded into the **I2TDx** Register to be transmitted to the slave device. Again, CMP and ACKI bits must be tested for proper response. At this point, the slave device is still in a write mode. Therefore, another start sequence must be sent to the slave to change the mode to read by first setting the STA and ENA bits in the **I2COx** Register and then writing the device address (bits 24-30) and read bit (bit 31=1) to the **I2TDx** Register. After CMP and ACKI bits have been tested for proper response, the I^2C master controller writes a dummy value (data=don't care) to the **I2TDx** Register.This causes the I^2C master controller to initiate a read transmission from the slave device. Again, CMP bit must be tested for proper response. After the I^2C master controller has received a byte of data (indicated by DIN=1 in the **I2STx** Register), the system software may then read the data by polling the **I2RDx** Register. The I^2C master controller does not acknowledge the read data for a ***single*** byte transmission on the I^2C bus, but must complete the transmission by sending a stop sequence to the slave device. This can be accomplished by first setting the STP and ENA bits in the **I2COx** Register and then writing a dummy data (data=don't care) to the **I2TDx** Register. The **I2STx** Register must now be polled to test CMP bit for the operation-complete status. The stop sequence will relinquish the Harrier's master possession of the I^2C bus. The following figure shows the suggested software flow diagram for programming the I^2C random read operation.

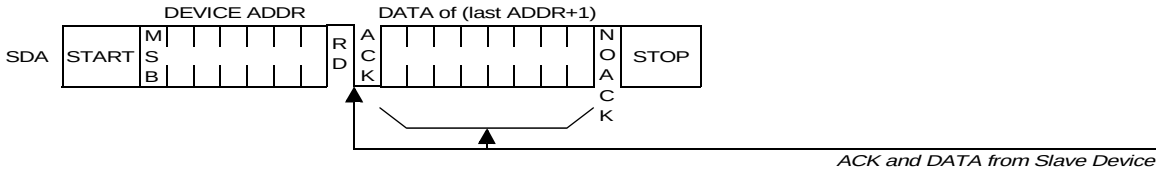


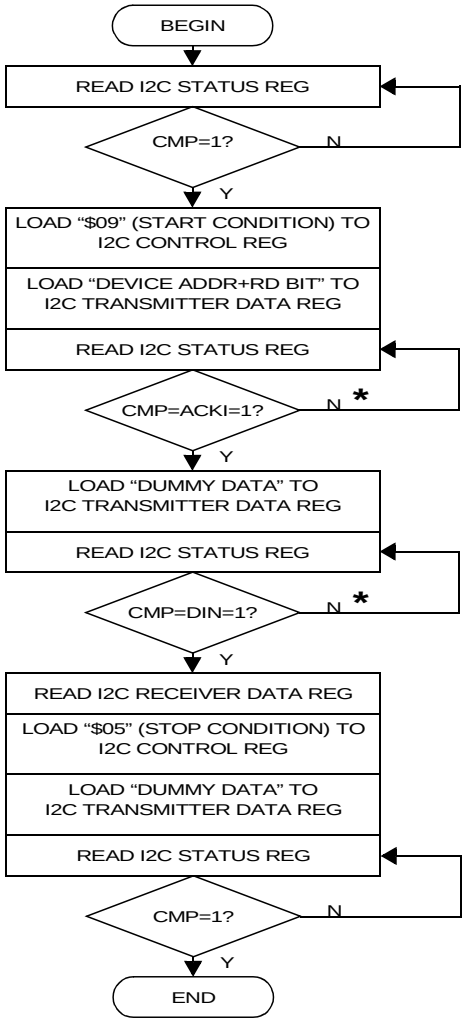
***)**: *Stop condition should be generated to abort the transfer after a software wait loop* (~100µs @100KHz SCL) *has been expired*

Figure 2-19. Programming Sequence for I²C Random Read

Current Address Read

The I^2C slave device should maintain the last address accessed during the last I^2C read or write operation, incremented by one. The first step in the programming sequence should be to test the CMP bit for the operation-complete status. The next step is to initiate a start sequence by first setting the STA and ENA bits in the **I2COx** Register and then writing the device address (bits 24-30) and read bit (bit 31=1) to the **I2TDx** Register. The CMP bit will be automatically clear with the write cycle to the **I2TDx** Register. The **I2STx** Register must now be polled to test the CMP and ACKI bits. The CMP bit becomes set when the device address and read bit have been transmitted, and the ACKI bit provides status as to whether or not a slave device acknowledged the device address. With the successful transmission of the device address, the I^2C master controller writes a dummy value (data=don't care) to the **I2TDx** Register.This causes the I^2C master controller to initiate a read transmission from the slave device. Again, CMP bit must be tested for proper response. After the I^2C master controller has received a byte of data (indicated by DIN=1 in the **I2STx** Register), the system software may then read the data by polling the **I2RDx** Register. The I^2C master controller does not acknowledge the read data for a ***single*** byte transmission on the I^2C bus, but must complete the transmission by sending a stop sequence to the slave device. This can be accomplished by first setting the STP and ENA bits in the **I2COx** Register and then writing a dummy data (data=don't care) to the **I2TDx** Register. The **I2STx** Register must now be polled to test CMP bit for the operation-complete status. The stop sequence will relinquish the Harrier master's possession of the I^2C bus. The following figure shows the suggested software flow diagram for programming the I^2C current address read operation.



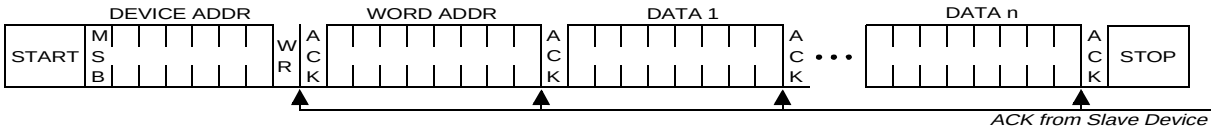


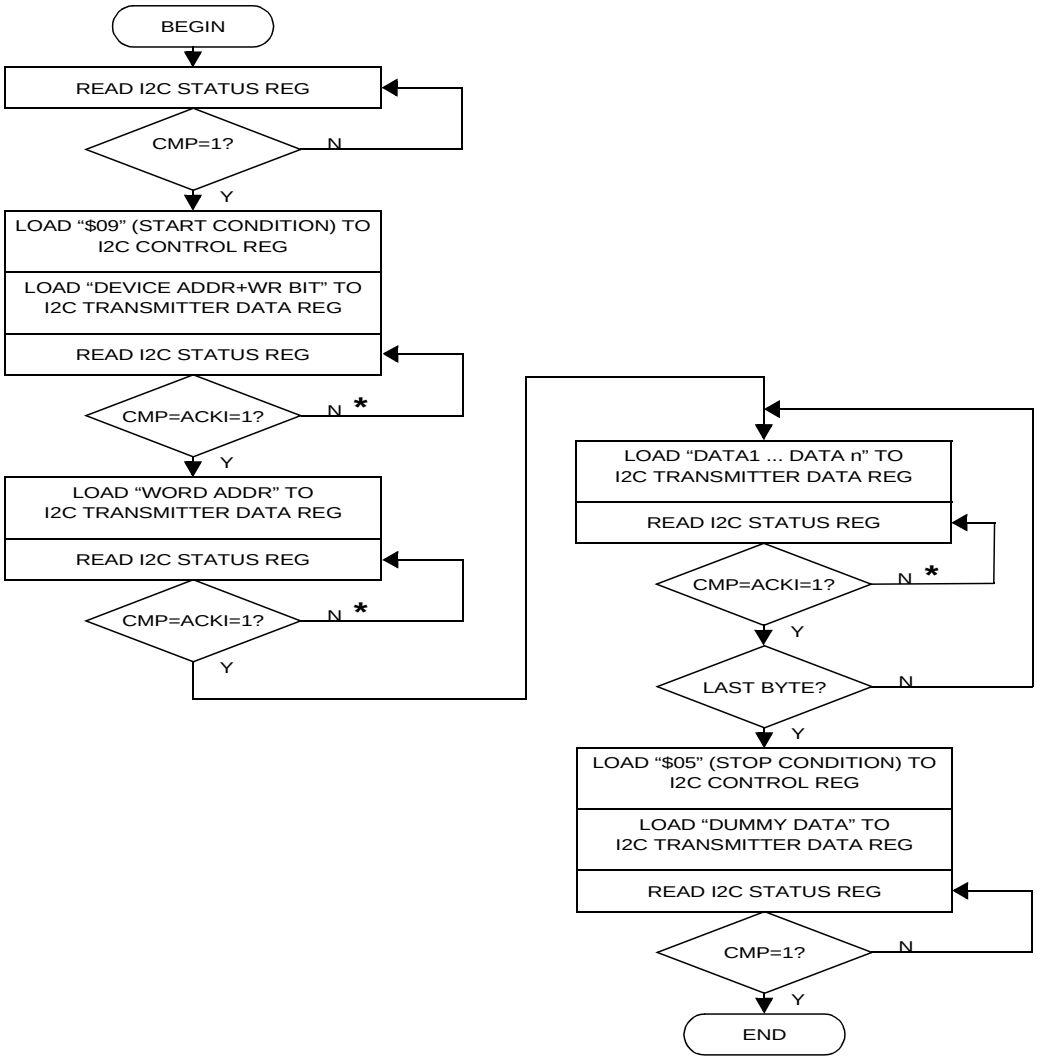
(*): *Stop condition should be generated to abort the transfer after a software wait loop* (~100µs @100KHz SCL) *has been expired*

Figure 2-20. Programming Sequence for I²C Current Address Read

Page Write

The I^2C page write is initiated the same as the I^2C byte write, but instead of sending a stop sequence after the first data word, the I^2C master controller will transmit more data words before a stop sequence is generated. The first step in the programming sequence should be to test the CMP bit for the operation-complete status. The next step is to initiate a start sequence by first setting the STA and ENA bits in the **I2COx** Register and then writing the device address (bits 24-30) and write bit (bit 31=0) to the **I2TDx** Register. The CMP bit is automatically cleard with the write cycle to the **I2TDx** Register. The **I2STx** Register must now be polled to test the CMP and ACKI bits. The CMP bit becomes set when the device address and write bit have been transmitted, and the ACKI bit provides status as to whether or not a slave device acknowledged the device address. With the successful transmission of the device address, the initial word address will be loaded into the **I2TDx** Register to be transmitted to the slave device. Again, CMP and ACKI bits must be tested for proper response. After the initial word address is successfully transmitted, the first data word loaded into the **I2TDx** Register will be transferred to the initial address location of the slave device. After CMP and ACKI bits have been tested for proper response, the next data word loaded into the **I2TDx** Register will be transferred to the next address location of the slave device, and so on, until the block transfer is complete. A stop sequence then must be transmitted to the slave device by first setting the STP and ENA bits in the **I2COx** Register and then writing a dummy data (data=don't care) to the **I2TDx** Register. The **I2STx** Register must now be polled to test CMP bit for the operation-complete status. The stop sequence will initiate a programming cycle for the serial EEPROM and also relinquish the Harrier master's possession of the I^2C bus. The following figure shows the suggested software flow diagram for programming the I^2C page write operation.





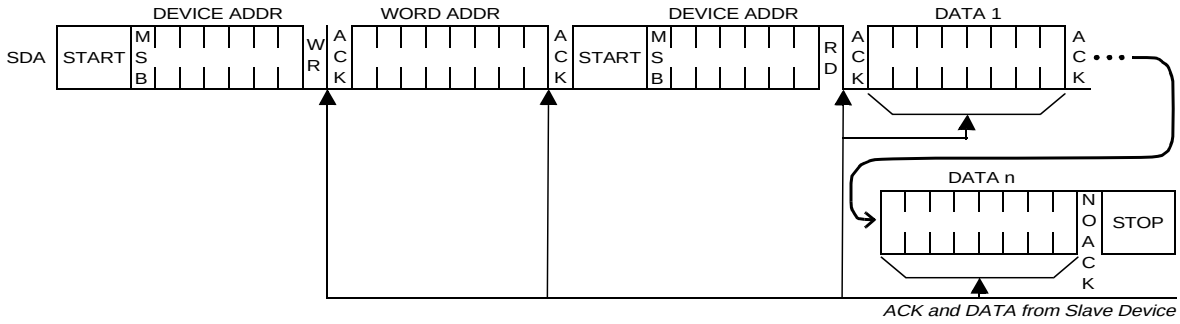
(*): *Stop condition should be generated to abort the transfer after a software wait loop* (~100µs @100KHz SCL) *has been expired*

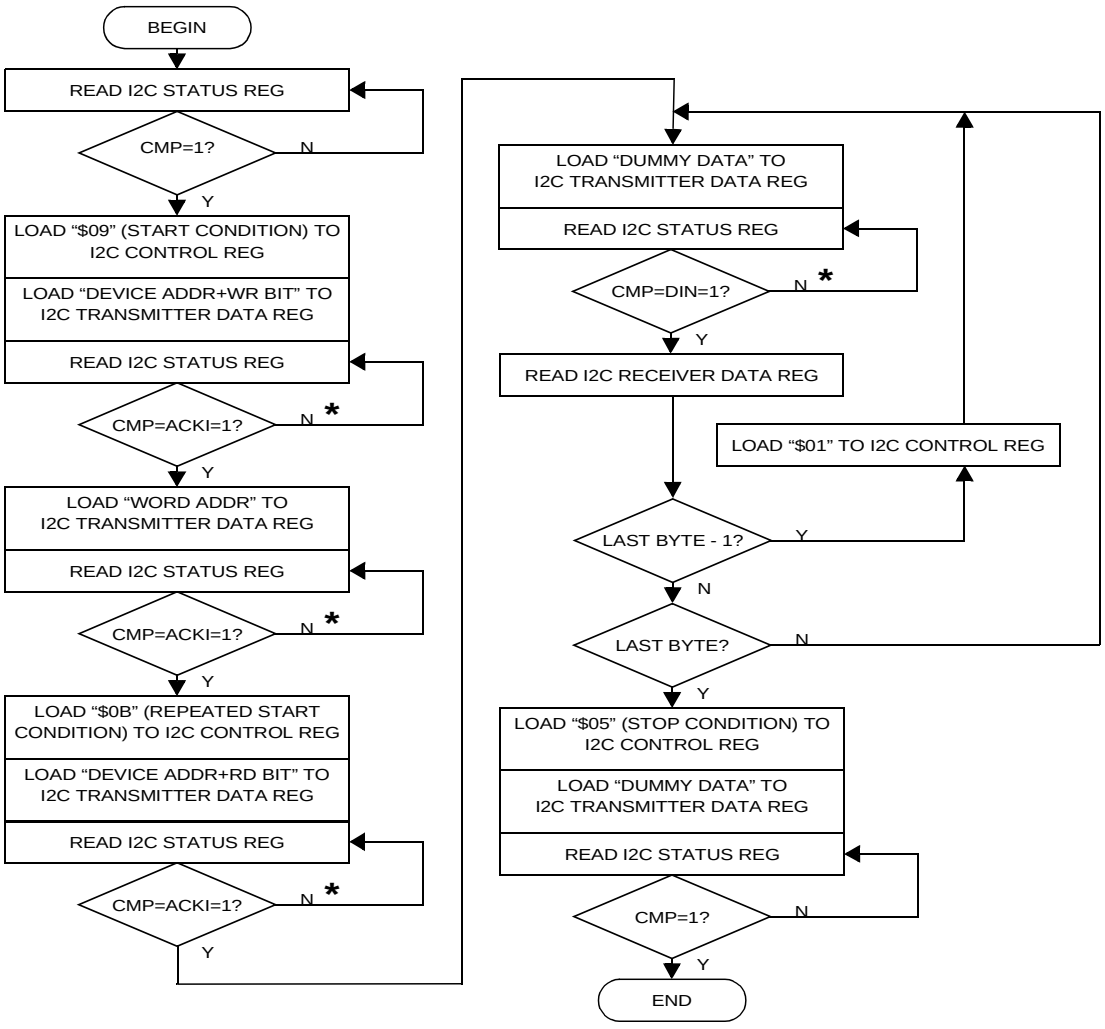
Figure 2-21. Programming Sequence for I²C Page Write

Sequential Read

Note that the I^2C sequential read can be initiated by either an I^2C random read (described here) or an I^2C current address read. With an I^2C random read initiation, the first step in the programming sequence should be to test the CMP bit for the operation-complete status. The next step is to initiate a start sequence by first setting the STA and ENA bits in the **I2COx** Register and then writing the device address (bits 24-30) and write bit (bit 31=0) to the **I2TDx** Register. The CMP bit will be automatically clear with the write cycle to the **I2TDx** Register. The **I2STx** Register must now be polled to test the CMP and ACKI bits. The CMP bit becomes set when the device address and write bit have been transmitted, and the ACKI bit provides status as to whether or not a slave device acknowledged the device address. With the successful transmission of the device address, the initial word address will be loaded into the **I2TDx** Register to be transmitted to the slave device. Again, CMP and ACKI bits must be tested for proper response. At this point, the slave device is still in a write mode. Therefore, another start sequence must be sent to the slave to change the mode to read by first setting the STA, ACKO, and ENA bits in the **I2COx** Register and then writing the device address (bits 24-30) and read bit (bit 31=1) to the **I2TDx** Register. After CMP and ACKI bits have been tested for proper response, the I^2C master controller writes a dummy value (data=don't care) to the **I2TDx** Register.This causes the I^2C master controller to initiate a read transmission from the slave device. After the I^2C master controller has received a byte of data (indicated by DIN=1 in the **I2STx** Register) and the CMP bit has also been tested for proper status, the I^2C master controller will respond with an acknowledge and the system software may then read the data by polling the **I2RDx** Register. As long as the slave device receives an acknowledge, it will continue to increment the word address and serially clock out sequential data words. The I^2C sequential read operation is terminated when the I^2C master controller does not respond with an acknowledge. This can be accomplished by setting *only* the ENA bit in the **I2COx** Register before receiving the last data word. A stop sequence then must be transmitted to the slave device by first setting the STP and ENA bits in the **I2COx** Register and then writing a dummy data (data=don't care) to the **I2TDx** Register. The **I2STx** Register must now be polled to test CMP bit for the operation-complete

status. The stop sequence will relinquish the Harrier's master's possession of the I^2C bus. The following figure shows the suggested software flow diagram for programming the I^2C sequential read operation.





* p p ~ u p

Figure 2-22. Programming Sequence for I²C Sequential Read

UART Interface

The Universal Asynchronous Receiver/Transmitter (UART) interface provides two full duplex serial ports to support communication with modems or other serial peripheral devices. This interface is compatible with the standard National Semiconductor 16550 UART. For detailed information on the functionality and programming model, refer to the *IBM Universal Asynchronous Receiver/Transmitter with FIFOs For 5S/5SE/SA-12E* and *National Semiconductor PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs* documents.

The UART control and status registers are described in the section titled *UART Controller* on page 3-136.

XPORT

The Xport is a bridge that interfaces the PowerPC bus to an expansion bus named Xport Bus. Xport Bus is the set of signals the Harrier uses to control devices that have a simple, “static RAM” style interface. Such devices might include flash, ROM, control registers, and FIFO’s.

A PowerPC bus slave and an Xport Bus master constitute the most significant blocks that make up Xport. The following figure shows a simplified block diagram.

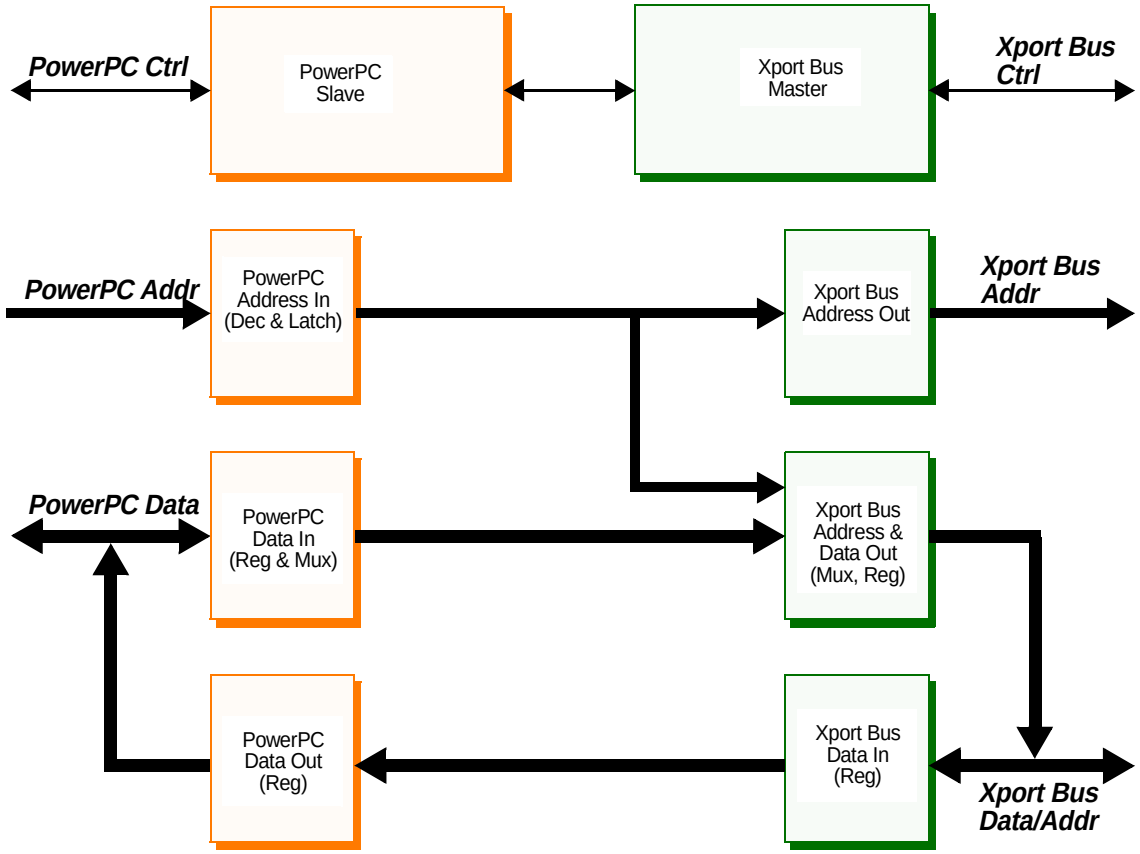


Figure 2-23. Xport Block Diagram

The PowerPC bus slave has four address response ranges. The Xport Bus master has four corresponding chip selects. An address range with its corresponding chip select is referred to as a channel. Each channel employs a combination of control registers and input signal pins to configure its address range and attributes. The section titled *Xport* on page 3-150 details the control registers. The section titled *Hardware Configuration* on page 2-133 explains the use of input signals.

The following two sections detail the PowerPC slave and Xport Bus master.

PowerPC Slave

The PowerPC slave provides an interface between the PowerPC bus and the Xport Bus master. Each of the Xport's four channels is programmable with its own PowerPC bus address range and attributes.

The PowerPC slave uses a "delayed transaction" protocol for its response to PowerPC accesses. For reads, the PowerPC slave issues retry's until data is available. For writes, the PowerPC slave latches and acknowledges data immediately but issues retry's until the data is written. The following paragraph gives more details about Xport Bus-bound reads.

When an Xport Bus-bound read begins, the PowerPC slave responds with AACK_ and ARTRY_ and requests that the Xport Bus master obtain the data. The slave continues to "ARTRY_" Xport Bus-bound accesses until data is obtained and the original PowerPC master comes back to get it. Some CPU PowerPC masters do not always come back to get the original data. If it appears that a CPU has abandoned an Xport bus read, the PowerPC slave and Xport bus master also abandon the read and begin accepting new PowerPC accesses. The PowerPC slave assumes that a CPU has abandoned a read if it begins a non-matching, Xport-bound PowerPC access before completion of the original data tenure, or if the CPU does not begin or request a new PowerPC access within 31 CLK periods of Xport's capturing data from the Xport bus.

Xport Bus Master

The Harrier Xport Bus master furnishes the interface between the PowerPC slave and the Xport Bus. It performs one Xport Bus transaction for each corresponding PowerPC transaction. Xport Bus transactions contain one address phase and one data phase made up of one or more data beats. The number of data beats depends on the amount of data to move, and the active Xport channel's data width configuration.

When performing an Xport Bus transaction, the Xport Bus master adopts the active channel's Xport Bus attributes. Refer to the section titled *Xport* on page 3-150. These attributes include basic mode enable, data port width, burst enable, burst length, access delay, and burst access delay. The following paragraphs describe these attributes.

The Harrier uses data port width to determine which data to use and how many data beats to issue. If the port width is 8 bits, the Harrier uses data signals D31-D24 and issues one beat per data byte. If the port width is 16 bits, the Harrier uses data signals D31-D16 and issues one beat per 2 aligned data bytes. Finally, if the port width is 32 bits, the Harrier uses D31-D0 and issues one beat per 4 aligned data bytes.

The Harrier uses the other attributes to determine which access time to use for each data beat. When bursting is disabled the Harrier uses the access delay (***XCSR*.XPAT**.AD) for every data beat within a transaction. When bursting is enabled the Harrier uses ***XCSR*.XPAT**.AD as the access delay for only the first beat of each burst. The Harrier uses the burst access delay (***XCSR*.XPAT**.BRD/***XCSR*.XPAT**.BWD) for the remaining beats of each burst. In all cases, the access time can be extended beyond the programmed time by using the XWAIT_ input signal.

If the ***XCSR*.XPAT**.BAM bit is set, the Harrier accesses the Xport Bus in the “basic timing mode”. In the basic timing mode, the Harrier uses only non-burst accesses and lengthens signal timings for compatibility with less flexible Xport Bus devices.

Xport Bus Transaction Examples

Xport Bus accesses begin with the address phase, in which the master presents address and attributes. The accesses continue with the data phase, in which the master transfers 1 to 32 beats of data. Depending on programmed configuration, the beats of data may involve bursting.

The following discussion shows different examples. Note that the signals named XA represent all XADR signals and any XAD signals that are configured as address signals only.

The first few examples are for the simplest case in which the Xport Bus master requires only one beat to transfer the requested data. The following figure shows a one-beat read with ***XCSR*.XPAT**.BAM cleared.

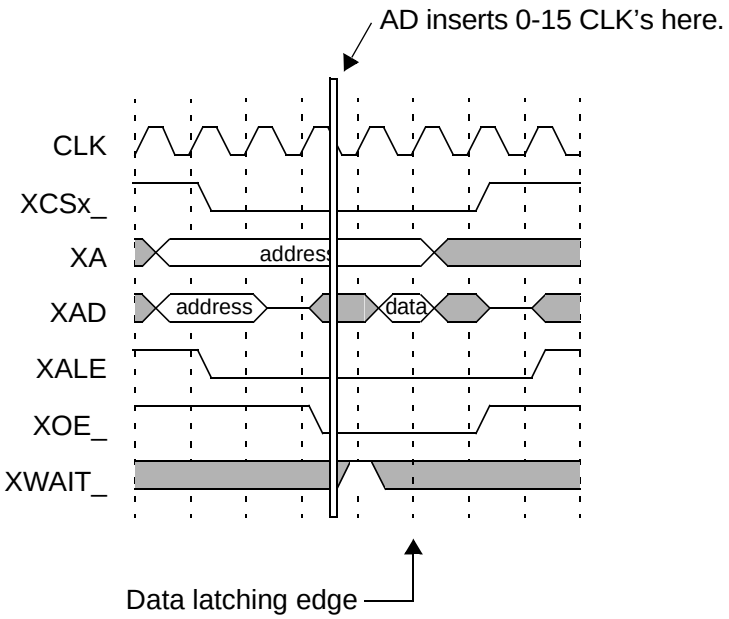


Figure 2-24. Xport Bus One-Beat Read Transaction

The following figure shows two back-to-back one-beat write transactions with ***XCSR*.XPAT**.BAM cleared.

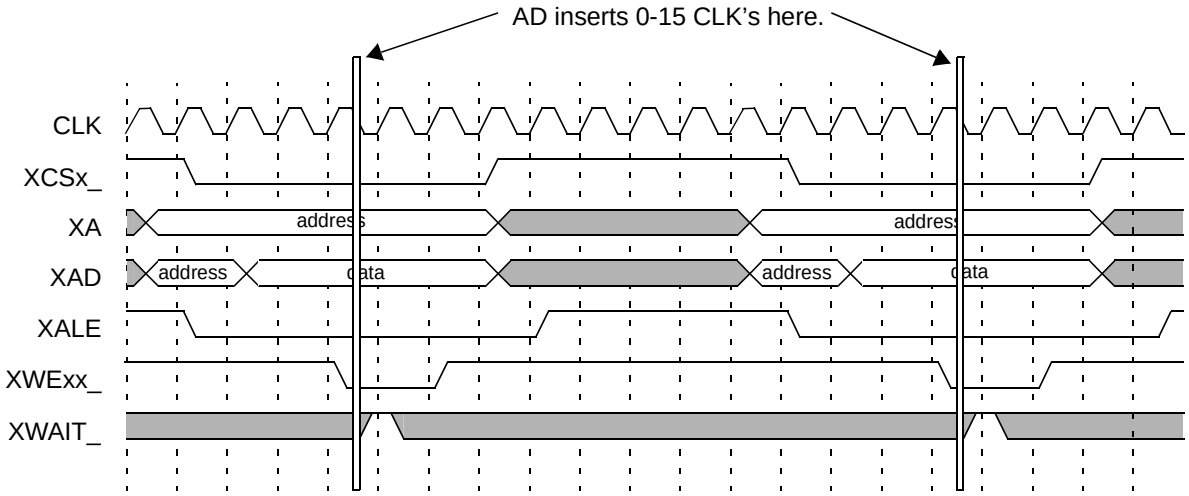


Figure 2-25. Xport Bus Two One-beat Write Transactions

When multiple data beats need to transfer during a single transaction, the Xport Bus master increments the address after each beat. If writing, the master also regenerates WE_ for each beat. If bursting is off, the master uses ***XCSR*.XPAT**.AD as the access time for all data beats, If bursting is on, the master uses ***XCSR*.XPAT**.BRD/***XCSR*.XPAT**.BWD for each beat after the first in the burst.

The following figure shows a two-beat read transaction with no bursting and ***XCSR*.XPAT**.BAM.

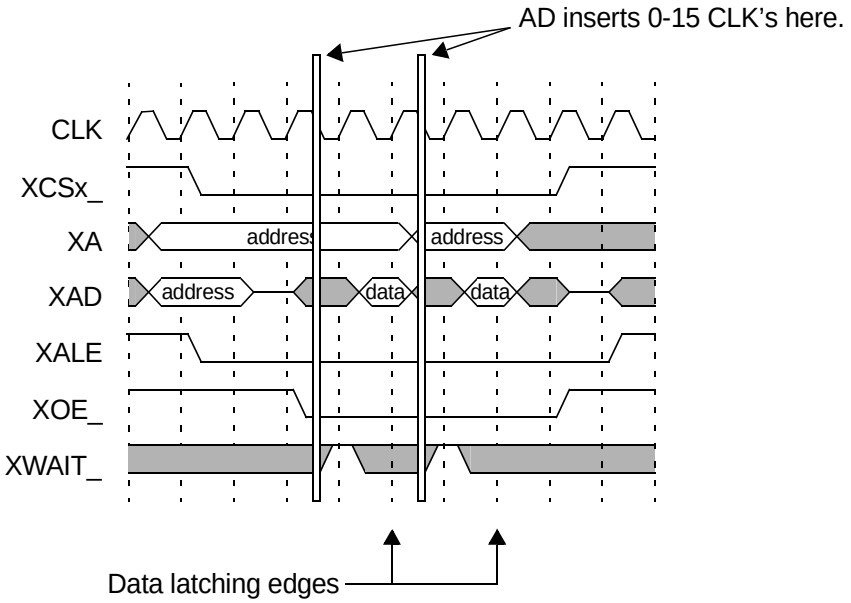


Figure 2-26. Xport Bus Two-Beat Read Transaction (No Bursting)

The following figure shows a multi-beat read transaction with bursting on and ***XCSR*.XPAT**.BAM cleared. The burst size matches the number of beats to transfer. Note the use of XWAIT_ to extend the last data beat.

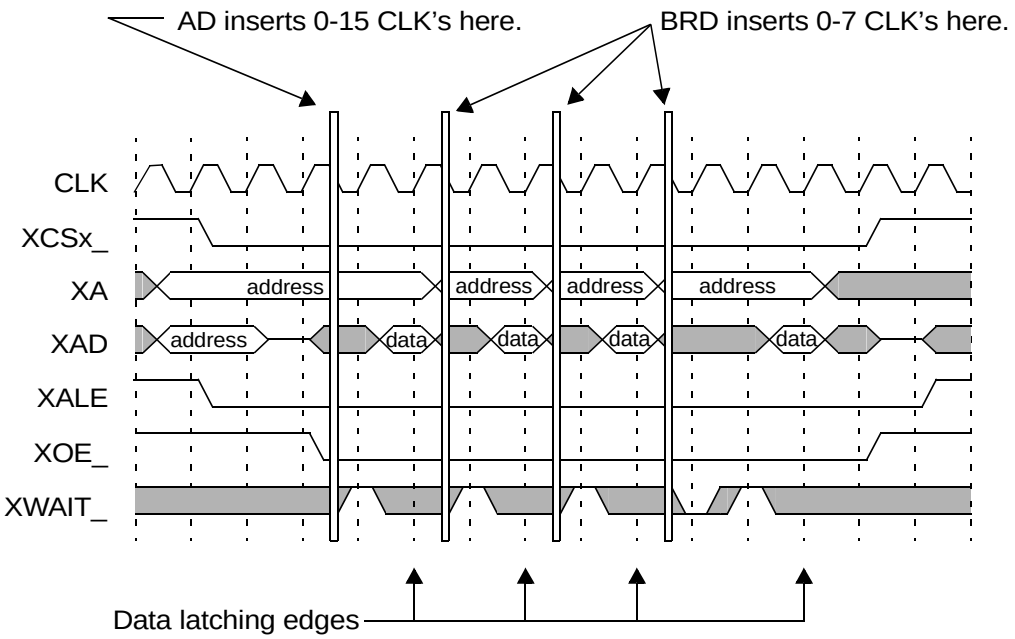


Figure 2-27. Xport Bus 4-beat Read Transaction with Burst Size of 4

The following figure shows a multi-beat write transaction with bursting turned on and ***XCSR*.XPAT**.BAM cleared. The burst size is greater than the number of beats to transfer. XWAIT_ delays one data.

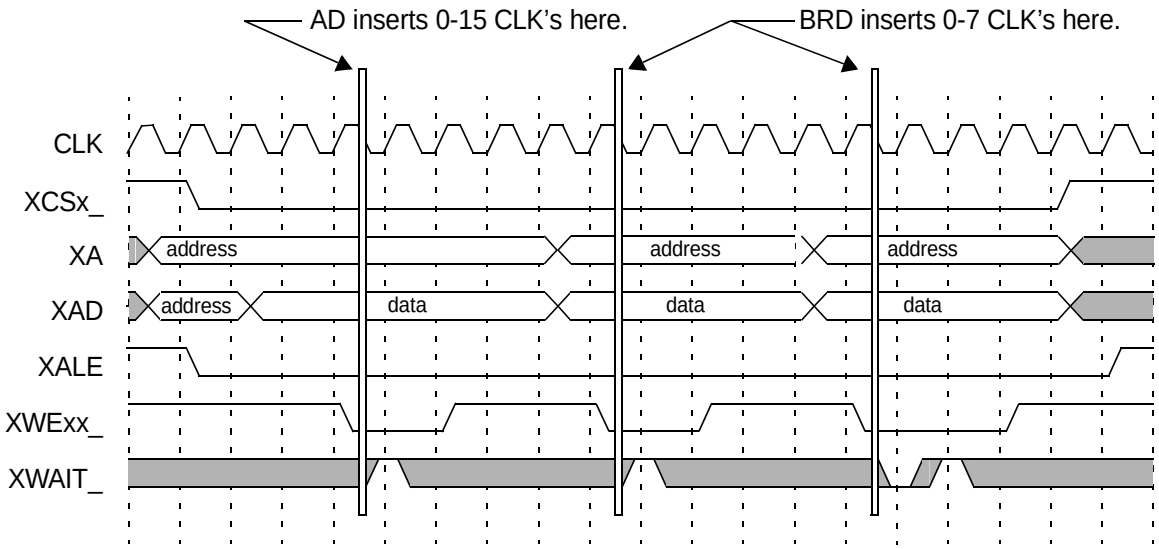


Figure 2-28. Xport Bus 3-beat Write Transaction with Burst Size of 4

The following figure shows a multi-beat read with bursting turned on and ***XCSR*.XPAT**.BAM cleared. Multiple bursts are employed because the number of beats to transfer exceeds the burst size.

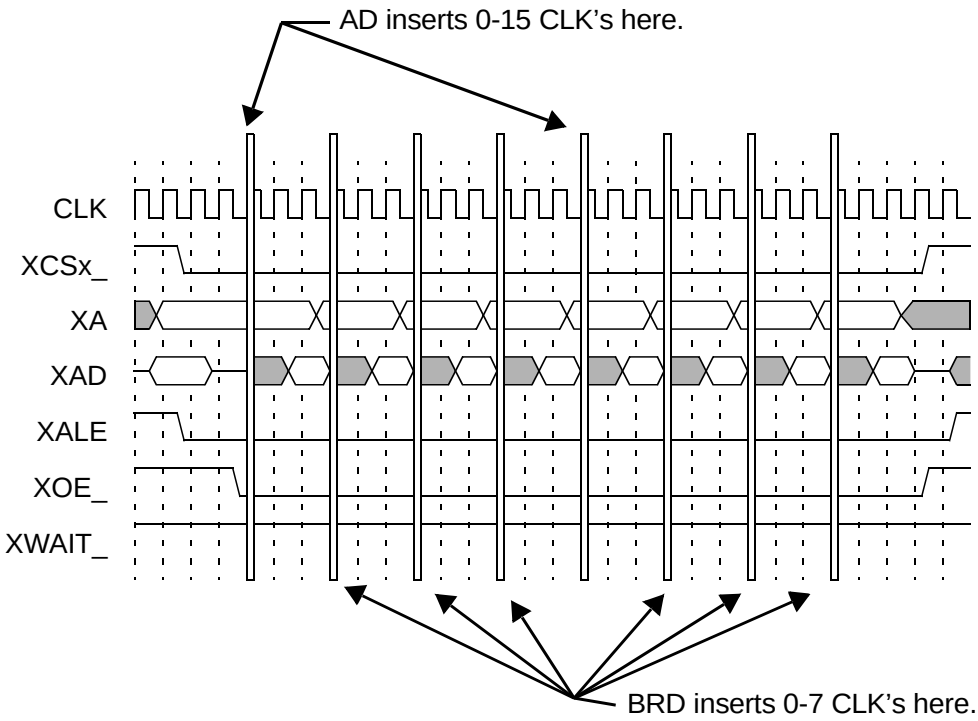


Figure 2-29. Xport Bus, 8-beat Read Transaction with Burst Size of 4

In all the preceding timing diagrams, the ***XCSR*.XPAT**.BAM (Basic Mode) control bit is cleared. In the next two, it is set. When ***XCSR*.XPAT**.BAM is set, all Xport transactions are broken into single-beats on the Xport Bus and cycle timing slows to allow longer setup and hold times.

The following figure shows an Xport Bus read when ***XCSR*.XPAT**.BAM is set.

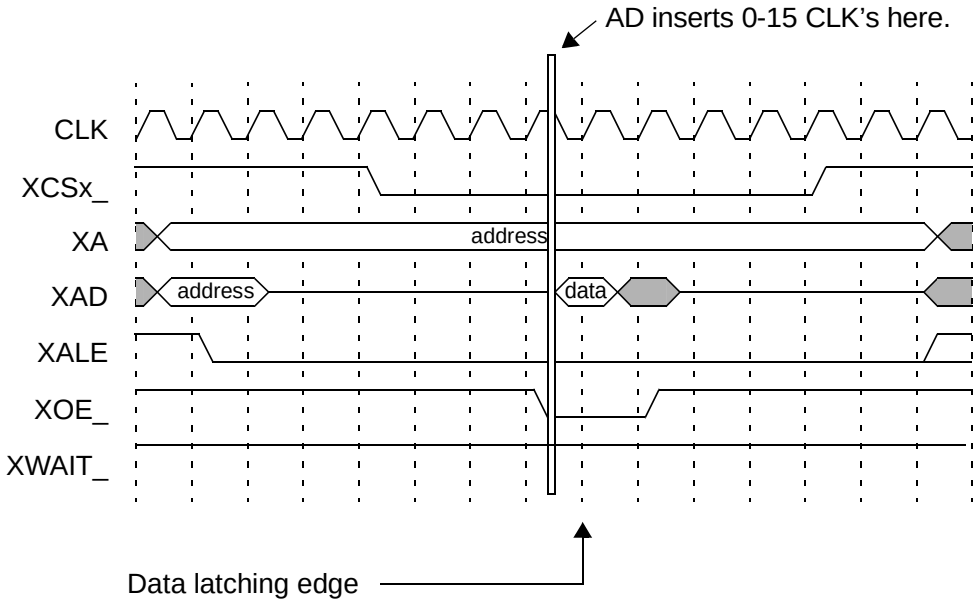


Figure 2-30. Xport Bus One-Beat Read Transaction in Basic Mode

The following figure shows an Xport Bus write when ***XCSR*.XPAT**.BAM is set.

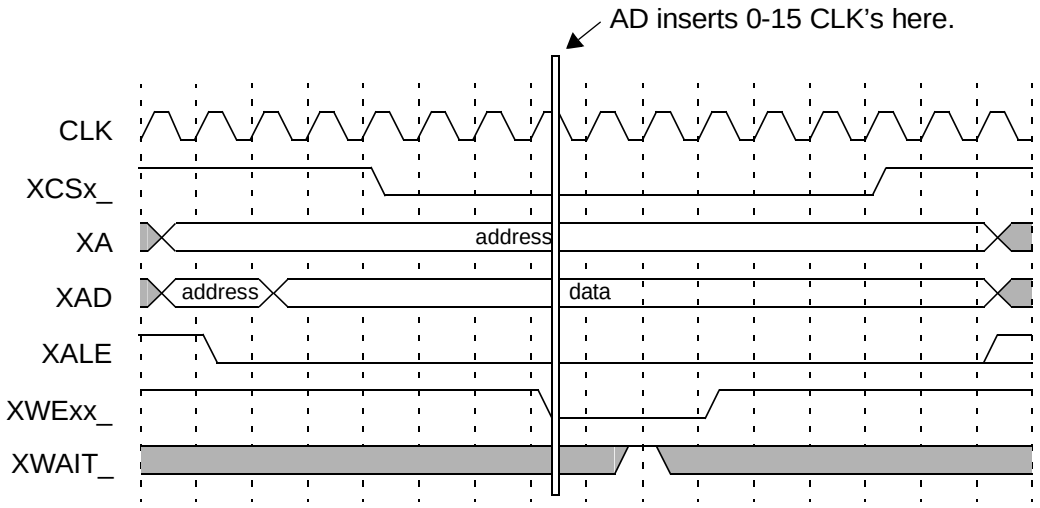


Figure 2-31. Xport Bus One-Beat Write Transaction in Basic Mode

Xport's Xport Bus master has a special Hawk compatibility mode that is enabled when the ***XCSR*.XPAT**.DW bits are set to 1,1. When in this mode, the channel width is fixed at 16 bits, XALE's polarity reverses, and the ***XCSR*.XPAT**.BAM control bit has the same effect on timing whether in Hawk compatibility mode or not.

The following figure shows an Xport Bus single read transaction in a Hawk compatibility mode.

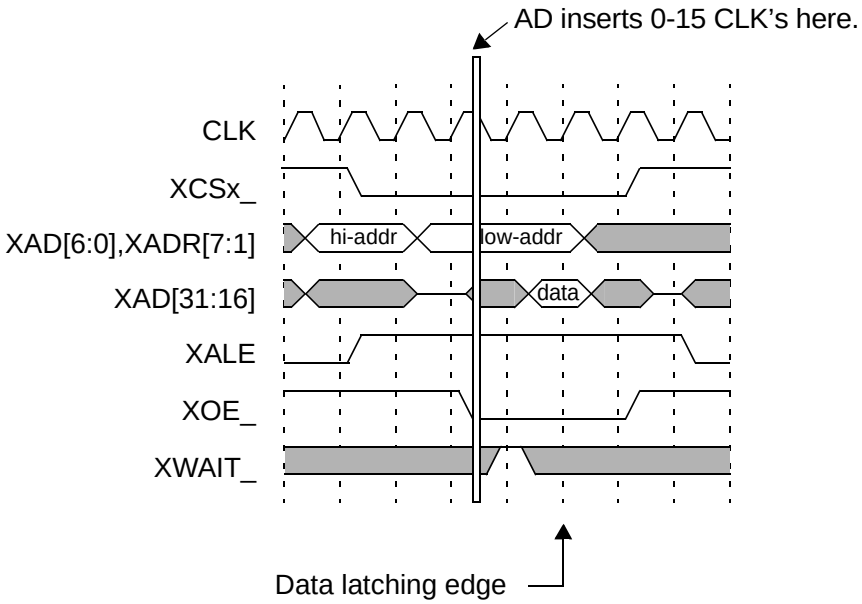


Figure 2-32. Xport Bus One-Beat Read Transaction in Hawk Compatibility Mode

The following figure shows an Xport Bus single write transaction in a Hawk compatibility mode.

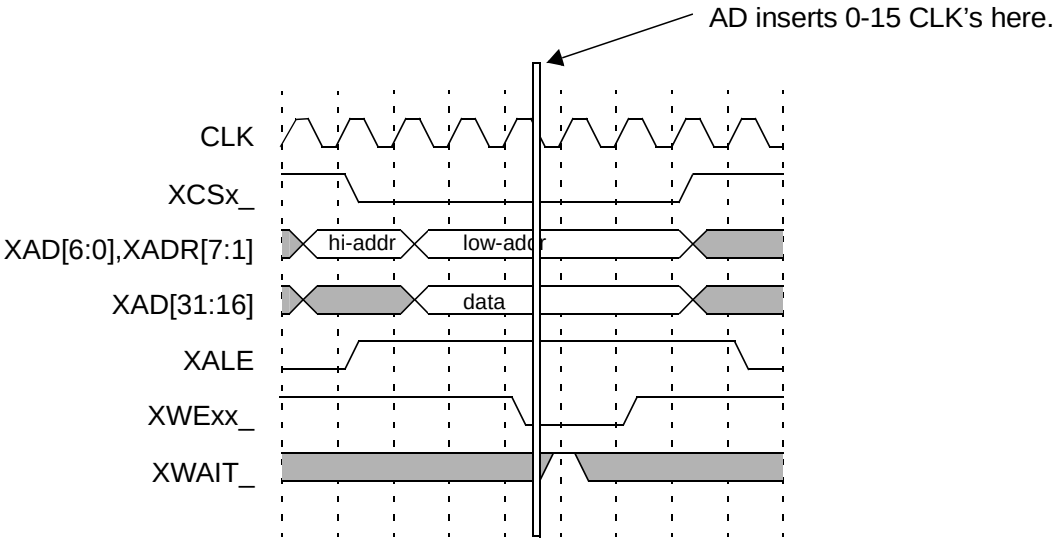


Figure 2-33. Xport Bus One-Beat Write Transaction in Hawk Compatibility Mode

Xport Bus Address Mapping

Xport Bus has a 10-bit dedicated address bus (XADR 25, 24 and 7-0) and a 32-bit multiplexed address and data bus (XAD31:0). The highest bit number is the most significant bit and bit zero is the least significant bit. The number of address lines available depends on the width of the device and whether an external latch is used. The following table shows how 8, 16 and 32-bit wide devices are connected to Xport Bus. These tables assume the Xport Bus width is configured to match the device width. A latch is not required for 8-bit devices. A latch is not required for 16-bit devices unless more than 128MB of address space is required. A latch is required for 32-bit devices if more than 256-bytes of address space is required. Address lines prefixed by an L (for example, LXADR0) indicate a latched version of the address line. A 373 type transparent latch should be used to latch the address lines. If a latch is used, 8 and 16-bit devices may be connected to either the latched or unlatched version of the address line.

.

Table 2-9. Xport Bus Address Mapping

MPU Address	8-bit Devices		16-bit Devices			32-bit Devices		
	Device Address	Xport Bus Address (4GB)	Device Address	Xport Bus Address (128MB)	Xport Bus Address 16-bit Latch (4GB)	Device Address	Xport Bus Address 16-bit Latch (256MB)	Xport Bus Address 24-bit Latch (4GB)
A31	A0	XADR0						
A30	A1	XADR1	A0	XADR1	XADR1			
A29	A2	XADR2	A1	XADR2	XADR2	A0	XADR2	XADR2
A28	A3	XADR3	A2	XADR3	XADR3	A1	XADR3	XADR3
A27	A4	XADR4	A3	XADR4	XADR4	A2	XADR4	XADR4
A26	A5	XADR5	A4	XADR5	XADR5	A3	XADR5	XADR5
A25	A6	XADR6	A5	XADR6	XADR6	A4	XADR6	XADR6
A24	A7	XADR7	A6	XADR7	XADR7	A5	XADR7	XADR7
A23	A8	XAD0	A7	XAD0	XAD0	A6	LXAD0	LXAD0
A22	A9	XAD1	A8	XAD1	XAD1	A7	LXAD1	LXAD1
A21	A10	XAD2	A9	XAD2	XAD2	A8	LXAD2	LXAD2
A20	A11	XAD3	A10	XAD3	XAD3	A9	LXAD3	LXAD3
A19	A12	XAD4	A11	XAD4	XAD4	A10	LXAD4	LXAD4
A18	A13	XAD5	A12	XAD5	XAD5	A11	LXAD5	LXAD5
A17	A14	XAD6	A13	XAD6	XAD6	A12	LXAD6	LXAD6
A16	A15	XAD7	A14	XAD7	XAD7	A13	LXAD7	LXAD7
A15	A16	XAD8	A15	XAD8	XAD8	A14	LXAD8	LXAD8
A14	A17	XAD9	A16	XAD9	XAD9	A15	LXAD9	LXAD9
A13	A18	XAD10	A17	XAD10	XAD10	A16	LXAD10	LXAD10
A12	A19	XAD11	A18	XAD11	XAD11	A17	LXAD11	LXAD11
A11	A20	XAD12	A19	XAD12	XAD12	A18	LXAD12	LXAD12
A10	A21	XAD13	A20	XAD13	XAD13	A19	LXAD13	LXAD13
A9	A22	XAD14	A21	XAD14	XAD14	A20	LXAD14	LXAD14
A8	A23	XAD15	A22	XAD15	XAD15	A21	LXAD15	LXAD15
A7	A24	XAD16	A23	XADR24	LXAD16	A22	XADR24	LXAD16
A6	A25	XAD17	A24	XADR25	LXAD17	A23	XADR25	LXAD17

Table 2-9. Xport Bus Address Mapping (Continued)

MPU Address	8-bit Devices		16-bit Devices			32-bit Devices		
	Device Address	Xport Bus Address (4GB)	Device Address	Xport Bus Address (128MB)	Xport Bus Address 16-bit Latch (4GB)	Device Address	Xport Bus Address 16-bit Latch (256MB)	Xport Bus Address 24-bit Latch (4GB)
A5	A26	XAD18	A25	XADR0	LXAD18	A24	XADR0	LXAD18
A4	A27	XAD19	A26		LXAD19	A25	XADR1	LXAD19
A3	A28	XAD20	A27		LXAD20	A26		LXAD20
A2	A29	XAD21	A28		LXAD21	A27		LXAD21
A1	A30	XAD22	A29		LXAD22	A28		LXAD22
A0	A31	XAD23	A30		LXAD23	A29		LXAD23

Xport Bus XAD Mapping

The table below shows the function of the XAD lines during the address phase and data phase. During the address phase, XAD(23-0) are address lines. During the data phase, the width of the device determines which XAD signals are address lines and which are data lines. The data lines of 8-bit devices are connected to XAD(31-24). The data lines of 16-bit devices are connected to XAD(31-16). The data lines of 32-bit devices are connected to XAD(31-0).

Table 2-10. Xport Bus XAD Mapping

XAD	Address Phase	Data Phase 8-bit	Data Phase 16-bit	Data Phase 32-bit
XAD7-XAD0	60x A24-A31	60x A24-A31	60x A24-A31	Device D7-D0
XAD15-XAD8	60x A16-A23	60x A16-A23	60x A16-A23	Device D15-D8
XAD23-XAD16	60x A8-A15	60x A8-A15	Device D7-D0	Device D23-D16
XAD29-XAD24	Byte Count	Device D5-D0	Device D13-D8	Device D29-D24
XAD30	Reserved	Device D6	Device D14	Device D30
XAD31	Read/Write	Device D7	Device D15	Device D31

Hawk Compatibility Mode

In a previous design, Hawk's Flash interface was routed through a connector to Flash devices located on another board. To allow the Harrier to provide the same interface, a Hawk compatibility mode was added. This mode should only be used to provide backward compatibility with hardware developed for the Hawk ASIC. The following table shows the address mapping when operating in the Hawk compatibility mode. A 374 type edge triggered latch should be used to latch the address lines

Table 2-11. Hawk Compatible Address Mapping

MPU Address	16-bit Device Hawk Compatibility Mode		
	Device Address	Xport Bus Address 14-bit Latch (512MB)	Hawk Address
A31			
A30	A0	XADR1	RA0
A29	A1	XADR2	RA1
A28	A2	XADR3	RA2
A27	A3	XADR4	RA3
A26	A4	XADR5	RA4
A25	A5	XADR6	RA5
A24	A6	XADR7	RA6
A23	A7	XAD0	RA7
A22	A8	XAD1	RA8
A21	A9	XAD2	RA9
A20	A10	XAD3	RA10
A19	A11	XAD4	RA11
A18	A12	XAD5	BA0
A17	A13	XAD6	BA1
A16	A14	LXADR1	LRA0
A15	A15	LXADR2	LRA1

Table 2-11. Hawk Compatible Address Mapping (Continued)

MPU Address	16-bit Device Hawk Compatibility Mode		
	Device Address	Xport Bus Address 14-bit Latch (512MB)	Hawk Address
A14	A16	LXADR3	LRA2
A13	A17	LXADR4	LRA3
A12	A18	LXADR5	LRA4
A11	A19	LXADR6	LRA5
A10	A20	LXADR7	LRA6
A9	A21	LXAD0	LRA7
A8	A22	LXAD1	LRA8
A7	A23	LXAD2	LRA9
A6	A24	LXAD3	LRA10
A5	A25	LXAD4	LRA11
A4	A26	LXAD5	LBA0
A3	A27	LXAD6	LBA1

Xport Bus Byte Mapping

The following tables describe how the data bytes are mapped to 8, 16 and 32-bit devices. When the Hawk compatibility mode is used, there are two data mapping options. When the Hawk data compatibility mode is not used the data is mapped as shown in the second table in this section. When the Hawk data compatibility mode is used, the data is mapped as shown in the third table in this section. Refer to the section titled *Hardware Configuration* on page 2-133 for information on configuring the Xport modes.

Table 2-12. 8-bit Device Byte Lane Mapping

PowerPC Bus Address	8-bit Device Address	Byte XAD(31-24)
$00000000	$00000000	0
$00000001	$00000001	1
$00000002	$00000002	2
$00000003	$00000003	3

Table 2-13. 16-bit Device Byte Lane Mapping

PowerPC Bus Address	16-bit Device Address	Byte XAD(31-24)	Byte XAD(23-16)
$00000000	$00000000	0	1
$00000002	$00000002	2	3
$00000004	$00000004	4	5
$00000006	$00000006	6	7

Table 2-14. Hawk Data Compatibility Byte Lane Mapping

PowerPC Bus Address	16-bit Device Address	Byte XAD(31-24)	Byte XAD(23-16)
$00000000	$00000000	0	
$00000001	$00000001	1	
$00000002	$00000002	2	
$00000003	$00000003	3	
$00000004	$00000004		4
$00000005	$00000005		5
$00000006	$00000006		6
$00000007	$00000007		7

Table 2-14. Hawk Data Compatibility Byte Lane Mapping (Continued)

PowerPC Bus Address	16-bit Device Address	Byte XAD(31-24)	Byte XAD(23-16)
$00000008	$00000008	8	
$00000009	$00000009	9	
$0000000A	$0000000A	A	
$0000000B	$0000000B	B	
$0000000C	$0000000C		C
$0000000D	$0000000D		D
$0000000E	$0000000E		E
$0000000F	$0000000F		F

Table 2-15. 32-bit Device Byte Lane Mapping

PowerPC Bus Address	32-bit Device Address	Byte XAD(31-24)	Byte XAD(23-16)	Byte XAD(15-8)	Byte XAD(7-0)
$00000000	$00000000	0	1	2	3
$00000004	$00000004	4	5	6	7
$00000008	$00000008	8	9	A	B
$0000000C	$0000000C	C	D	E	F

Arbiters

The Harrier has an internal PowerPC bus arbiter. The use of this arbiter is optional. If the internal arbiter is disabled, the Harrier must be allowed to participate in an externally implemented PowerPC arbitration mechanism. The selection of either internal or external PowerPC arbitration mode is made by sampling an XAD line at the release of reset. Please see the section titled *Hardware Configuration* on page 2-133 for more information.

The Harrier is designed to accommodate up to four PowerPC bus masters, including itself (HARR), two processors (CPU0/CPU1) and an external PowerPC master (EXTL). EXTL can be a second bridge chip, etc. The PPC Arbiter can optionally support a three bridge system, in which case the third bridge device would be connected to the CPU1 request/grant pair.

When the PPC Arbiter is disabled, the Harrier generates an external request and listens for an external grant for itself. It also listens to the other external grants to determine the PowerPC master identification field (MID) within the **GCSR** register. When the PPC Arbiter is enabled, the Harrier will receive requests and issue grants for itself and for the other three bus masters. The MID field is determined by the PPC Arbiter.

The PowerPC arbitration signals and their functions are summarized in the table below.

.

Table 2-16. PPC Arbiter Pin Assignments

Pin Name	Pin Type	Reset	Internal Arbiter		External Arbiter	
			Direction	Function	Direction	Function
XARB0	BiDir	Tristate	Output	CPU0 Grant_	Input	CPU0 Grant_
XARB1	BiDir	Tristate	Output	CPU1 Grant_	Input	CPU1 Grant_
XARB2	Output	- -	Output	EXTL Grant_	Output	HARR Request_

Table 2-16. PPC Arbiter Pin Assignments (Continued)

Pin Name	Pin Type	Reset	Internal Arbiter		External Arbiter	
			Direction	Function	Direction	Function
XARB3	Input	- -	Input	CPU0 Request_	Input	CPU0 Request_
XARB4	Input	- -	Input	CPU1 Request_	Input	CPU1 Request_
XARB5	Input	- -	Input	EXTL Request_	Input	HARR Grant_

While RST_ is asserted, XARB0 and XARB1 are held in tri-state. If the internal arbiter mode is selected, then XARB0 and XARB1 are driven to an active state no more than ten clock periods after the Harrier has detected a rising edge on RST_.

The PPC Arbiter implements a fully rotational prioritization scheme. Each master is guaranteed at least one access on the PowerPC bus. The timing of a priority switch is controlled by a bus master. Once a master obtains ownership of the bus, the priority remains in the favor of that master until the master momentarily releases it's request. This release is a signal to the PPC Arbiter to switch to the next rotated priority. The end effect is that each master has complete control of how much PowerPC bandwidth it consumes. It is the responsibility of each PowerPC master to control it's bandwidth consumption according to system level requirements. The **GCSR**.XBS field controls how much bandwidth the Harrier consumes.

The relationship between the retention of priority and leaving request asserted is called *Latching Request.* This function is intended to give bridge devices a way to guarantee a portion of the PowerPC bandwidth. It is not intended to be used by processor devices. The PPC Arbiter always implements the latching request function for the HARR and EXTL bus masters. Latching Request is optionally implemented for the CPU1 bus master, and is never implemented for the CPU0 bus master.

The PPC Arbiter supports four parking modes. Parking is implemented only on the CPUs and is not implemented on either HARR or EXTL. The parking options include parking on CPU0, parking on CPU1, parking on the last CPU, or parking disabled.

There are various system level debug functions provided by the PPC Arbiter. The PPC Arbiter has the optional ability to flatten the PowerPC bus pipeline. Flattening can be imposed uniquely on single beat reads, single beat writes, burst reads, and burst writes. It is possible to further qualify the ability to flatten based on whether there is a switch in masters or whether to flatten unconditionally for each transfer type. This is a debug function only and is not intended for normal operation.

PCI Arbiter

The Harrier has an optional internal PCI Arbiter. The arbiter can support up to 8 PCI masters, including the Harrier and 7 other external PCI masters. The arbiter can be configured to be enabled or disabled at reset time by strapping the XAD[18] bit either high for enabled or low for disabled. The table below describes the pins and its function for both modes

Table 2-17. PCI Arbiter Pin Description

Pin Name	Pin Type	Reset	Internal Arbiter		External Arbiter	
			Direction	Function	Direction	Function
PARBI0	Input	- -	Input	ext req0_	input	HARR gnt_
PARBI1	Input	- -	Input	ext req1_	Input	NA
PARBI2	Input	- -	Input	ext req2_	Input	NA
PARBI3	Input	- -	Input	ext_req3_	Input	NA
PARBI4	Input	- -	Input	ext_req4_	Input	NA
PARBI5	Input	- -	Input	ext req5_	Input	NA
PARBI6	Input	- -	Input	ext req6_	Input	NA
PARBO0	Output	Tristate	Output	ext gnt0_	Output	HARR req_
PARBO1	Output	Tristate	Output	ext gnt1_	Output	NA
PARBO2	Output	Tristate	Output	ext gnt2_	Output	NA
PARBO3	Output	Tristate	Output	ext gnt3_	Output	NA
PARBO4	Output	Tristate	Output	ext gnt4_	Output	NA
PARBO5	Output	Tristate	Output	ext gnt5_	Output	NA
PARBO6	Output	Tristate	Output	ext gnt6_	Output	NA

The PCI Arbiter is controlled by the **PARB** register within the *XCSR* Register Group.

The PCI Arbiter supports three different priority schemes: fixed, round robin and mixed mode. It also provides different prioritization options within either the fixed and mixed mode. Parking can be disabled, given to any one requester, or given to the last requester. A special bit is added to hold grant asserted for an agent that initiates a lock cycle. Once a lock cycle is detected, the grant is held asserted until the PCI LOCK_ pin is released. This feature works only when the POL feature is enabled.

The priority scheme can be programmed by writing the PRI field in the **PARB** register. The Fixed mode holds each requestor at a fixed level in its hierarchy. The levels of priority for each requestor are programmable by writing the HEIR field in the **PARB** register. Table 2-18 on page 2-117 describes all available settings for the HEIR field in fixed mode.

When the PCI Arbiter is programmed for round robin priority mode, the arbiter maintains fairness and provides equal opportunity to the requestors by rotating its grants. The contents of HEIR are “don’t cares” when operated in this mode. Round Robin mode is the default priority scheme on power up.

When the PCI Arbiter is programmed for mixed mode, the 8 requestors are separated into 4 groups. Each group has 2 requestors. PARB6 and PARB5 are defined in group 1; PARB4 and PARB3 are defined in group 2; PARB2 and PARB1 are defined in group 3; PARB0 and HARR are defined in group 4. Arbitration is set for round robin mode between the 2 requestors within each group and set for fixed mode between the 4 groups. The levels of priority for each group is programmable by writing the HEIR field in the **PARB** register. Table 2-19 on page 2-117 describes all available settings for the HEIR field in mixed mode.

Table 2-18. HEIR Encoding for Fixed Mode Priority

HEIR	Priority Levels							
	Highest							**Lowest**
000	PARB6	PARB5	PARB4	PARB3	PARB2	PARB1	PARB0	HARR
001	HARR	PARB6	PARB5	PARB4	PARB3	PARB2	PARB1	PARB0
010	PARB0	HARR	PARB6	PARB5	PARB4	PARB3	PARB2	PARB1
011	PARB1	PARB0	HARR	PARB6	PARB5	PARB4	PARB3	PARB2
100	PARB2	PARB1	PARB0	HARR	PARB6	PARB5	PARB4	PARB3
101	PARB3	PARB2	PARB1	PARB0	HARR	PARB6	PARB5	PARB4
110	PARB4	PARB3	PARB2	PARB1	PARB0	HARR	PARB6	PARB5
111	PARB5	PARB4	PARB3	PARB2	PARB1	PARB0	HARR	PARB6

Notes

1. "000" is the default setting in fixed mode.
2. HEIR only covers a small subset of all possible combinations. It is the responsibility of the system designer to connect the request/grant pair in a manner most beneficial

.

Table 2-19. HEIR Encoding for Mixed Mode Priority

HEIR	PRIORITY Levels			
	Highest			**Lowest**
000	group 1	group 2	group 3	group 4
	PARB 6 & 5	PARB 4 & 3	PARB 2 & 1	PARB 0 & HARR
001	group 4	group 1	group 2	group 3
	PARB 0 & HARR	PARB 6 & 5	PARB 4 & 3	PARB 2 & 1

Table 2-19. HEIR Encoding for Mixed Mode Priority (Continued)

HEIR	PRIORITY Levels			
	Highest			**Lowest**
010	group 3	group 4	group 1	group 2
	PARB 2 & 1	PARB 0 & HARR	PARB 6 & 5	PARB 4 & 3
011	group 2	group 3	group 4	group 1
	PARB 4 & 3	PARB 2 & 1	PARB 0 & HARR	PARB 6 & 5

Notes

1. "000" is the default setting in mixed mode.
2. HEIR only covers a small subset of all possible combinations and the requestors within each group is fixed and cannot be interchanged with other groups. It is the responsibility of the system designer to connect the request/grant pair in a manner that is most beneficial to his or her design goals.
3. All combinations of HEIR not specified in the table are invalid and should not be used.

Arbitration parking is programmable by writing to the PRK field of the **PARB** register. Parking can be programmed for any of the requestors, last requestor or none. The default setting for parking is “park on the Harrier”. The following table describes all available settings for the PRK field.

Table 2-20. PRK Encoding

PRK	Function
0000	Park on last requestor
0001	Park on PARB6
0010	Park on PARB5
0011	Park on PARB4
0100	Park on PARB3

Table 2-20. PRK Encoding (Continued)

PRK	Function
0101	Park on PARB2
0110	Park on PARB1
0111	Park on PARB0
1000	Park on HARR
1111	Parking disabled

Notes

1. "1000" is the default setting.
2. Parking disabled is a testmode only and should not be used, since nothing will drive the PCI bus when it is in the idle state.
3. All combinations of PRK not specified in the table are invalid and should not be used.

A special function is added to the PCI Arbiter to hold the grant asserted through a lock cycle. When the POL field in the **PARB** register is set, the grant associated with the agent initiating the lock cycle is held asserted until the lock cycle is complete. If this field is clear, the PCI Arbiter does not distinguish between lock and non-lock cycle.

2

Watchdog Timers

The Harrier features two watchdog timers called Watchdog Timer 0 (WT0) and Watchdog Timer 1 (WT1). Although both timers are functionally equivalent, each timer operates completely independent of each other. WT0 and WT1 are initialized at reset to a count value of 8 seconds and 16 seconds respectively. The timers are designed to be reloaded by software at any time. When not being loaded, the timer continuously decrements itself until either reloaded by software or a count of zero is reached. If a timer reaches a count of zero, an output signal will be asserted and the count will remain at zero until reloaded by software or the Harrier's reset is asserted. External logic can use the output signals of the timers to generate interrupts, machine checks, etc.

Each timer is composed of a prescaler and a counter. The prescaler determines the resolution of the timer, and is programmable to any binary value between 1 µs and 32,768 µs. The counter counts in the units provided by the prescaler. For example, the watchdog timer would reach a count of zero within 24 µs if the prescaler was programmed to 2 µs and the counter was programmed to 12.

The watchdog timers are controlled by registers mapped within the *XCSR* Register Group. Each timer has a **WTxC** register and a **WTxS** register. The **WTxC** register can be used to start or stop the timer, write a new reload value into the timer, or cause the timer to initialize itself to a previously written reload value. The **WTxS** register is used to read the instantaneous count value of the watchdog timer.

Programming the Watchdog Timers is a two step process:

1. First, 'arm' the **WTxC** register by writing PATTERN_1 into the KEY field. Only the KEY byte lane may be selected during this process. The **WTxC** register will not arm itself if any of the other byte lanes are selected or the KEY field is written with any other value than PATTERN_1. The operation of the timer itself remains unaffected by this write.

2. Next, write the new programming information to the **WTxC** register. The KEY field byte lane must be selected and must be written with PATTERN_2 for the write to take affect. The effects on

the **WTxC** register depend on the byte lanes that are written to during step 2 and are shown in the following table..

Table 2-21. WTxC Programming

Byte Lane Selection				Results			
KEY	**ENAB /RES**	**RELOAD**		**WTx**		**WTxC Register**	
0:7	**8:15**	**16:23**	**24:31**	**Prescaler/ Enable**	**Counter**	**RES/ENAB**	**RELOAD**
No	x	x	x	No Change	No Change	No Change	No Change
Yes	No	No	x	Update from RES/ENAB	Update from RELOAD	No Change	No Change
Yes	No	x	No	Update from RES/ENAB	Update from RELOAD	No Change	No Change
Yes	Yes	No	x	Update from data bus	Update from RELOAD	Update from data bus	No Change
Yes	Yes	x	No	Update from data bus	Update from RELOAD	Update from data bus	No Change
Yes	Yes	Yes	Yes	Update from data bus	Update from data bus	Update from data bus	Update from data bus

The **WTxC** register will always become unarmed after the second write regardless of byte lane selection. Reads may be performed at any time from the **WTxC** register and will not affect the write arming sequence.

2

Exceptions

There are two categories of exceptions that may be generated internal to the Harrier. One category of exceptions is called Error Exceptions, and the other is called Functional Exceptions. The Error Exceptions are associated with critical events that represent a possible compromise in the integrity of the system. The Functional Exceptions are associated with non-critical events pertaining to the normal flow of information within the system.

The Harrier's internal error exceptions are grouped together and routed as a singular interrupt to a dedicated channel within the MPIC. Similarly, the Harrier's internal functional exceptions are grouped together and routed to another dedicated channel within the MPIC. Please refer to the section titled "Multiprocessor Interrupt Controller" in this chapter for more information on how the two interrupts are handled within the MPIC. Error Exceptions may also be programmed to generate a machine check to processor 0, processor 1 or both.

Each error exception has an enable bit, a status bit, a clear bit, an interrupt enable bit, a machine check 0 enable bit and a machine check 1 enable bit. When the enable bit is set, the exception is enabled and the status bit reflects the status of the error exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero. If the interrupt enable bit is set, an interrupt will be generated whenever the corresponding status bit is set. If the machine check 0 enable bit is set, a machine check will be generated to processor 0 and if the machine check 1 enable bit is set, a machine check will be generated to processor 1 whenever the corresponding status bit is set. If the status bit is set, writing a one to clear bit will clear the status bit and the associated exception.

Each functional exception has an enable bit, a status bit and a mask bit. When the enable bit is set, the exception is enabled and the status bit reflects the status of the functional exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero. If the mask bit cleared, an interrupt will be generated whenever the corresponding status bit is set. If the mask bit is set, an interrupt will not be generated. In the case of DMA, MDB, MM0, MM1 and ABT functional exceptions if the status bit is set, writing a one to clear bit will clear the status bit and the associated exception. Note that the clear bits are in **FECL**

register and **MGID** register. In the case of MIP functional exception the status bit and the exception will be cleared when the processor adjusts the inbound post list head and tail pointers to match. For the UA0 and UA1 functional exceptions UART service routines will clear the status bit and the exception.

The following table summarizes the Harrier generated exceptions..

Table 2-22. Exception Summary

Category	Exception	Description	Primary Status	Additional Status	Clear	Edge/ Level
Error	PowerPC Address Bus Time-out	Any unclaimed PowerPC address tenure originating from any PowerPC bus master.	**EEST**.XBT	**EXAD/EXAT**	**EECL**.XBT (***XCSR***)	Edge
Error	PowerPC Address Parity Error	Detection of an address parity error during any address tenure involving any PowerPC bus master.	**EEST**.XAP	**EXAD/EXAT**	**EECL**.XAP (***XCSR***)	Edge
Error	PowerPC Data Parity Error	Detection of a data parity error during any data tenure involving any PowerPC bus master and any PowerPC bus slave.	**EEST**.XDP	**EXAD/EXAT**	**EECL**.XDP (***XCSR***)	Edge
Error	PowerPC Delayed Transaction Time-out	Any unclaimed PowerPC delayed transaction originating from any PowerPC bus master.	**EEST.**XDT	**EXAD/EXAT**	**EECL**.XDT (***XCSR***)	Edge
Error	SDRAM Memory Interface Single BIt Error	Detection and correction of a single bit error.	**EEST**.SSE	**SDSES/ SDSEA**	**EECL**.SSE (***XCSR***)	Edge

Table 2-22. Exception Summary (Continued)

Category	Exception	Description	Primary Status	Additional Status	Clear	Edge/ Level
Error	SDRAM Memory Interface Single Bit Error Overflow	Detection of a single bit error count overflow	**EEST**.SSC	**SDSES**	**EECL**.SSC (*XCSR*)	Edge
Error	SDRAM Memory Interface Multi Bit Error on PowerPC Access	Detection of a multi bit error during any PowerPC access.	**EEST**.SMX	**SDMES/ SDMEA**	**EECL**.SMX (*XCSR*)	Edge
Error	SDRAM Memory Interface Multi Bit Error on Scrub	Detection of a multi bit error on a scrub.	**EEST**.SMS	**SDMES/ SDMEA**	**EECL**.SMS (*XCSR*)	Edge
Error	PCI Master Abort	Detection of a Master Abort with the Harrier as a PCI Bus Master. Can be either a bridge or a DMA transaction.	**EEST**.PMA	**EPAD/EPAT**	EECL.PMA (*XCSR*)	Edge
Error	PCI Target Abort	Detection of a Target Abort with the Harrier as a PCI Bus Master. Can be either a bridge or a DMA transaction.	**EEST**.PTA	**EPAD/EPAT**	EECL.PTA (*XCSR*)	Edge
Error	PCI Address Parity Error	Detection of a parity error during the address phase of any PCI transfer involving any PCI bus master and target.	**EEST**.PAP	**EPAD/EPAT**	EECL.PAP (*XCSR*)	Edge

Table 2-22. Exception Summary (Continued)

Category	Exception	Description	Primary Status	Additional Status	Clear	Edge/ Level
Error	PCI Data Parity Error	Detection of a parity error during the data phase of any PCI transfer involving any PCI bus master and target.	**EEST**.PDP	**EPAD/EPAT**	EECL.PDP (*XCSR*)	Edge
Error	PCI SERR	Assertion of SERR. Note that this may or may not be as a result of the Harrier detecting an address parity error.	**EEST**.PSE	None	**EECL**.PSE (*XCSR*)	Edge
Error	PCI PERR	Assertion of PERR. Note that this may or may not be as a result of the Harrier detecting a data parity error.	**EEST**.PPE	None	**EECL**.PPE (*XCSR*)	Edge
Error	PCI Delayed Trans-action Time-out	Any unclaimed PCI delayed transaction originating from any PCI bus master.	**EEST.**PDT	**EXAD/EXAT**	**EECL**.PDT (*XCSR*)	Edge
Error	PCI Master Retry Error	Bridge or DMA as a PCI master has exceeded the maximum number of sequential retries.	**EEST**.PMR	**EPAD/EPAT**	EECL.PMR (*XCSR*)	Edge
Functional	DMA Controller	Completion of a Direct Mode transfer or a Linked-List Mode transaction. May be accompanied by an error condition.	**FEST**.DMA	**DSTA**	**FECL**.DMA (*XCSR*)	Edge
Functional	MP Generic Doorbell	Assertion of any Inbound Doorbell bit.	**FEST**.MDB	**MGID**	MGID.DBIx (*XCSR*)	Edge

Table 2-22. Exception Summary (Continued)

Category	Exception	Description	Primary Status	Additional Status	Clear	Edge/ Level
Functional	MP Generic Message #0	New message written to Message Passing Register 0.	**FEST**.MM0	**MGIM0**	**FECL**.MM0 (*XCSR*)	Edge
Functional	MP Generic Message #1	New message written to Message Passing Register 1.	**FEST**.MM1	**MGIM1**	**FECL**.MM1 (*XCSR*)	Edge
Functional	MP I2O Inbound post_list	New entry written into Inbound Post_List queue. (**MIIPH** != **MIIPT**)	**FEST**.MIP	**MIIPH/ MIIPT**	MIIPH = MIIPT	Level
Functional	UART #0	UART0 Received data available interrupt, time-out interrupt in the FIFO mode, Transmitter holding register empty interrupt, Receiver line status interrupt or MODEM status interrupt.	**FEST**.UA0	**IDFC0**	UART0 service routine	Level
Functional	UART #1	UART1 Received data available interrupt, time-out interrupt in the FIFO mode, Transmitter holding register empty interrupt, Receiver line status interrupt or MODEM status interrupt.	**FEST**.UA1	**IDFC1**	UART1 service routine	Level
Functional	Abort	Assertion of ABTSW_ pin for a short period.	**FEST**.ABT	None	**FECL**.ABT (*XCSR*)	Edge

Error Diagnostics

The Harrier provides a set of registers to efficiently diagnose error exceptions whenever a transaction terminates abnormally. These registers are contained within the *XCSR* Register Group.

In the event of a PowerPC bus time-out, PowerPC delayed transaction time-out, PowerPC address parity error or PowerPC data parity error, the corresponding PowerPC error status bit will be set in the **EEST** register if the corresponding error is enabled (**EEEN** register) any other PowerPC error status bit is not set. In addition, the Harrier will capture the address and attributes corresponding to the erroneous transaction in the **EXAD** register and **EXAT** register respectively. If any PowerPC error is detected while any other PowerPC error status bit is already set, the PowerPC overflow (**EEST**.XOF) bit will be set in the **EEST** register and further logging of PowerPC errors is suspended until software clears the first error and the PowerPC overflow bit. The PowerPC overflow bit (in the **EEST** register) is set to prevent the Harrier from updating **EXAD** and **EXAT** registers in the event of additional errors. Thus, the information related to the first erroneous transaction is latched until the error status bit is cleared by writing to **EECL** register.

Similarly, in the event of a PCI master abort, PCI target abort, PCI address parity error, PCI data parity error or PCI delayed transaction time-out the corresponding PCI error status bit will be set in the **EEST** register if the corresponding error is enabled (**EEEN** register) any other PCI error status bit is not set. In addition, the Harrier will capture the PCI address and attributes corresponding to the erroneous transaction in the **EPAD** register and **EPAT** register respectively. If any PCI error is detected while any other PCI error status bit is already set, the PCI overflow (**EEST**.POF) bit will be set in the **EEST** register and further logging of PCI errors is suspended until software clears the first error and the PCI overflow bit. The PCI overflow bit (in the **EEST** register) is set to prevent the Harrier from updating **EPAD** and **EPAT** registers in the event of additional errors. Thus, the information related to the first erroneous transaction is latched until the error status bit is cleared by writing to **EECL** register.

The Harrier memory controller supports ECC for correcting single bit errors and detecting double bit errors. In the event of a memory controller single bit error or single bit error count overflow, the corresponding SDRAM error status bit will be set in the **EEST** register if the corresponding error is enabled (**EEEN** register) any other SDRAM single bit error status bit is not set. In addition, the Harrier will update **SDSES** and **SDSEA** registers in the case of an SDRAM single bit error. In the case any SDRAM single bit error detected while any other SDRAM single bit error status bit is already set, the SDRAM single bit overflow (**EEST**.SSOF) bit will be set in the **EEST** register and further logging of SDRAM single bit errors is suspended until software clears the first error and the **EEST**.SSOF bit. In the event of a memory controller multi bit error on PowerPC access or multi bit error on a scrub, the corresponding SDRAM error status bit will be set in the **EEST** register if the corresponding error is enabled (**EEEN** register) any other SDRAM multi bit error status bit is not set. In addition, the Harrier will update **SDMES** and **SDMEA** registers in the case of an SDRAM multi bit scrub error and **SDMEA** register in the case of an SDRAM multi bit error due to a PowerPC access. In the case any SDRAM multi bit error detected while any other SDRAM multi bit error status bit is already set, the SDRAM multi bit overflow (**EEST**.SMOF) bit will be set in the **EEST** register and further logging of SDRAM multi bit errors is suspended until software clears the first error and the **EEST**.SMOF bit.

The following table shows how the Harrier captures address and attribute information for various error exceptions.

Table 2-23. Error Exceptions and Address/Attribute Capture

Error Status	Error Address and Attributes
XBT	From PowerPC bus
XAP	From PowerPC bus
XDP	From PowerPC bus
XDT	From PowerPC bus
PMA	From PCI bus
PTA	From PCI bus
PAP	From PCI bus
PDP	From PCI bus

Table 2-23. Error Exceptions and Address/Attribute Capture (Continued)

Error Status	Error Address and Attributes
PMR	From PCI bus
PDT	From PCI bus
PSE	Invalid
PPE	Invalid

PowerPC Address Bus Timer

The PPC Timer allows the current bus master to recover from a potential lock-up condition caused when there is no response to a transfer request. The time-out length of the bus timer is determined by the BTO field within the **GCSR** register.

The PPC Timer is designed to handle the case where an address tenure is not closed out by the assertion of AACK_. The PPC Timer will not handle the case where a data tenure is not closed out by the appropriate number of TA_ assertions. The PPC Timer will start timing at the exact moment when the PowerPC bus pipeline has gone flat. In other words, the current address tenure is pending closure, all previous data tenures have completed, and the current pending data tenure awaiting closer is logically associated with the current address tenure.

The time-out function is aborted if AACK_ is asserted anytime before the time-out period has passed. If the time-out period reaches expiration, then the PPC Timer will assert AACK_ to close the faulty address tenure. If the transaction was an address only cycle, then no further action will be taken. If the faulty transaction was a data transfer cycle, then the PPC Timer will assert the appropriate number of TA_'s to close the pending data tenure. Error information related to the faulty transaction will be latched within the **EXAD** and **EXAT** registers, and an interrupt or machine check will be generated depending on the programming of the **EEMA**, **EEDE** and **EEMS** registers.

There is one exception that will dynamically disable the PPC Timer. If the transaction is outbound, then the burden of closing out a transaction is left to the PCI bus. Note that a transaction to the PowerPC Control Register

Group is considered to be outbound since the completion of these types of accesses depends on the ability of the PCI bus to empty outbound write-posted data.

The ERRAT signal is an open collector (wired OR) bi-directional signal that is used to notify other PowerPC devices of an address bus timeout exception. As an output, ERRAT is asserted when the PPC Timer expires. As an input, Harrier uses ERRAT as an indication to the DMA controller that a DMA transaction must be aborted.

PowerPC Parity

The Harrier generates data parity whenever it is sourcing PowerPC data. This happens when performing a write as a bus master, and when servicing a read as a bus slave. Valid data parity will be presented when DBB_ is asserted for write cycles, and when TA_ is asserted for read cycles.

The Harrier checks data parity whenever it is sinking PowerPC data. This happens when performing a read as a bus master, and when servicing a write as a bus slave. Data parity is considered valid anytime TA_ is asserted. If a data parity error is detected, then address and attribute information will be latched within the **EXAD** and **EXAT** registers, and an interrupt or machine check will be generated depending on the programming of the **EEMA**, **EEDE** and **EEMS** registers.

The Harrier has a mechanism to purposely induce data parity errors for testability. The DPE field within the **EPEI** register can be used to purposely inject data parity errors on specific data parity lines. Data parity errors can only be injected during cycles that the Harrier is sourcing PowerPC data.

The Harrier generates address parity whenever it is sourcing a PowerPC address.This happens for all transactions where the Harrier is a bus master. Valid address parity is presented when ABB_ is asserted.

The Harrier checks address parity for all PowerPC bus transactions. If an address parity error is detected, then address and attribute information will be latched within the **EXAD** and **EXAT** registers, and an interrupt or machine check will be generated depending on the programming of the **EEMA**, **EEDE** and **EEMS** registers.

The Harrier has a mechanism to purposely inject address parity errors for testability. The APE field within the **EPEI** register can be used to purposely inject address parity errors on specific address parity lines. Address parity errors can only be injected when the Harrier is sourcing a PowerPC address.

Reset Signals

The Harrier has two reset signals: PURST_ and RST_. The PURST_ signal should be asserted at power up and resets all of the Harrier's internal logic. The PURST_ signal must be asserted for 50 µs after the VDD2V5 and VDD3V3 power supplies are stable, the CLK and PCLK signals are stable and the XAD signals used for configuration are stable. During this time, the Harrier synchronizes its internal clock generation logic. When the PURST_ signal is removed, the Harrier latches the hardware configuration as described in the section titled "Hardware Configuration" further on in this chapter. Whenever the PURST_ signal is asserted, it must be asserted for at least 50 µs.

The RST_ signal resets most of the Harrier's internal logic. The SDRAM refresh logic is not reset by the RST_ signal. This allows the contents of the SDRAM to be preserved when RST_ is asserted. The RST_ signal must be asserted for at least 100 cycles of the CLK signal. The PURST_ and RST_ signals may be asserted or negated in any sequence, however, the Harrier is not functional or does not respond to any PCI or PowerPC bus cycles until both PURST_ and RST_ are negated. The hardware configuration is not changed when RST_ is asserted.

The Harrier also has a reset out signal (RSTO_). The RSTO_ signal is asserted whenever the PURST_ signal is asserted, the RSTSW_ signal is asserted and enabled, AUXRST_ signal is asserted or the reset out bit (***XCSR***.**MCSR**.RSTOUT) is set. The RSTO_ signal does not reset any of

Harrier's internal logic. The RSTO_ signal is normally combined with other on board reset signals to generate the RST_ signal, which is connected to the Harrier. The on board reset logic must not generate any reset loops. When the Harrier is used on a Processor PCI Mezzanine Card (PPMC), the RSTO_ signal may be connected to the RESET_OUT# signal on the PPMC connector. The Harrier always keeps the RSTO_ signal asserted for 100 µs after the reset inputs are negated.

The Harrier provides support for an external reset switch that can be connected to the RSTSW_ input. The RSTSW_ input is debounced and has a three second delay. The RSTSW_ signal must be asserted for three seconds before the RSTO_ signal is generated. The three second delay is included to allow the RSTSW_ and the ABTSW_ (abort switch) signals to be connected together. This allows a single switch to be used for the abort and reset functions. When the switch is pressed, an abort interrupt is generated and if the switch is held for three seconds, the RSTO_ signal is asserted.

The Harrier provides an auxiliary reset input signal AUXRST_. The AUXRST_ input allows other reset sources on the board to be combined with the Harrier's internal reset sources. For example, a watchdog timer output from the Harrier can be connected to the AUXRST_ input.

Software can initiate a reset out signal by setting the reset out bit (***XCSR***.**MCSR**.RSTOUT). The reset out bit will remain asserted until it is reset by the RST_ signal.

When the Harrier receives a reset in signal (RST_), it saves the state of reset out bit, reset switch and auxiliary reset input. Software can determine the source of the reset by reading the bits in the ***XCSR*.GCSR** register.

PPMC Features

The Harrier includes several signals for use in the PPMC environment. These are the SCON_, EREADY, INT[A:D]_ and BRDFL_ signals. The SCON_ signal is an input signal that can be connected to the MONARCH# signal on the PPMC connector. This allows software to read the state of the MONARCH# signal.

The EREADY signal is an open drain bi-directional signal that can be connected to the EREADY signal on the PPMC connector. When PURST_ or RST_ is asserted, the EREADY bit is cleared and the EREADY signal is driven low. This indicates that the board is not ready. After the software has initialized the board, the EREADY bit can be set. When all the boards are ready, the EREADY signal is pulled high. The software can check the status of the EREADY signal by reading the EREADYS bit.

Hardware Configuration

Harrier has the ability to perform custom hardware configuration to accommodate different system requirements. It has several functions that may be optionally enabled or disabled using passive hardware external to the Harrier itself. Normally, pullup and pulldown resistors are used to configure the state of the XAD bus while PURST_ is asserted. The selection process occurs when PURST_ is asserted and the configuration values are latched when PURST_ is negated. All of the sampled pins are cascaded with several layers of registers to eliminate problems with hold time. The following table summarizes the Harrier's hardware configuration options. The power up reset timing requirements are shown in the figure following this table.

Table 2-24. Harrier Hardware Configuration

Function	Sample Pin(s)	Sampled State	Description	Timing Group
PCI 64-bit Enable	REQ64_	0	64-bit PCI Bus	PCI Spec
		1	32-bit PCI Bus	
Xport Hawk Data Compatibility Mode, ***XCSR*.XPGC**.HDM	XAD[30]	0	Disabled	1
		1	Enabled	
UART Clock Select	XAD[29]	0	External	1
		1	Internal	
PCI Slave Configuration Holdoff, ***XCSR*.BPCS**.CSH	XAD[28]	0	Normal access	1
		1	No access, disconnect with retry	

Table 2-24. Harrier Hardware Configuration (Continued)

Function	Sample Pin(s)	Sampled State	Description	Timing Group
PCI Slave Configuration Mask, ***XCSR*.BPCS**.CSM	XAD[27]	0	Unlimited access	1
		1	Access limited to 64 byte header	
PowerPC Processor Holdoff, ***XCSR*.BXCS**.P1H and ***XCSR*.BXCS**.P0H	XAD[26]	0	Processors 0 and 1 not held off	1
		1	Processors 0 and 1 held off	
SDRAM External Register, ***XCSR*.SDTC**.SDER	XAD[25]	0	No external registers in series for BAx, RAx, RAS_, CAS_, WE_	1
		1	External registers in series for BAx, RAx, RAS_, CAS_, WE_	
Response to Unmapped Address-Only Cycles See ***XCSR*.GCSR**.AOAO (See "Global Control and Status Register" in next Chapter 3)	XAD[24]	0	No Response from Harrier	1
		1	Response from Harrier	
Generic Power Up Status ***XCSR*.GCSR**.PUST3	XAD[23]	0	Cleared	1
		1	Set	
Generic Power Up Status ***XCSR*.GCSR**.PUST2	XAD[22]	0	Cleared	1
		1	Set	
Generic Power Up Status ***XCSR*.GCSR**.PUST1	XAD[21]	0	Cleared	1
		1	Set	
Generic Power Up Status ***XCSR*.GCSR**.PUST0 See "Global Control and Status Register" in next Chapter 3.	XAD[20]	0	Cleared	1
		1	Set	
I_2O IOP Agent	XAD[19]	0	False	1
		1	True	

Table 2-24. Harrier Hardware Configuration (Continued)

Function	Sample Pin(s)	Sampled State	Description	Timing Group
Internal PCI Arbiter	XAD[18]	0	Disabled	1
		1	Enabled	
Internal PowerPC Arbiter	XAD[17]	0	Disabled	1
		1	Enabled	
XCSR Register Group Base Address	XAD[16:15]	00	Base Address = $FEFF0000	1
		01	Base Address = $FEFF1000	
		10	Base Address = $FEFF2000	
		11	Base Address = $FEFF3000	
PowerPC:PCI Clock Ratio	XAD[14:12]	000	Reserved	1
		100	1:1	
		010	2:1	
		110	3:1	
		001	3:2	
		101	Reserved	
		011	5:2	
		111	Reserved	
Xport Channel 0 Data Width, ***XCSR*.XPAT0**.DW	XAD[11:10]	00	8-Bit Width	1
		01	16-Bit Width	
		10	32-Bit Width	
		11	16-Bit Width, Hawk compatibility mode	
Xport Channel 0 as Reset Vector Source	XAD[9]	0	Disabled	1
		1	Enabled (Only if Channels 1-3 are Disabled)	

Table 2-24. Harrier Hardware Configuration (Continued)

Function	Sample Pin(s)	Sampled State	Description	Timing Group
Xport Channel 1 Data Width, ***XCSR*.XPAT1**.DW	XAD[8:7]	00	8-Bit Width	1
		01	16-Bit Width	
		10	32-Bit Width	
		11	16-Bit Width, Hawk compatibility mode	
Xport Channel 1 as Reset Vector Source	XAD[6]	0	Disabled	1
		1	Enabled (Only if Channels 2-3 are Disabled)	
Xport Channel 2 Data Width, ***XCSR*.XPAT2**.DW	XAD[5:4]	00	8-Bit Width	1
		01	16-Bit Width	
		10	32-Bit Width	
		11	16-Bit Width, Hawk compatibility mode	
Xport Channel 2 as Reset Vector Source	XAD[3]	0	Disabled	1
		1	Enabled (Only if Channel 3 is Disabled)	
Xport Channel 3 Data Width, ***XCSR*.XPAT3**.DW	XAD[2:1]	00	8-Bit Width	1
		01	16-Bit Width	
		10	32-Bit Width	
		11	16-Bit Width, Hawk compatibility mode	
Xport Channel 3 as Reset Vector Source	XAD[0]	0	Disabled	1
		1	Enabled	

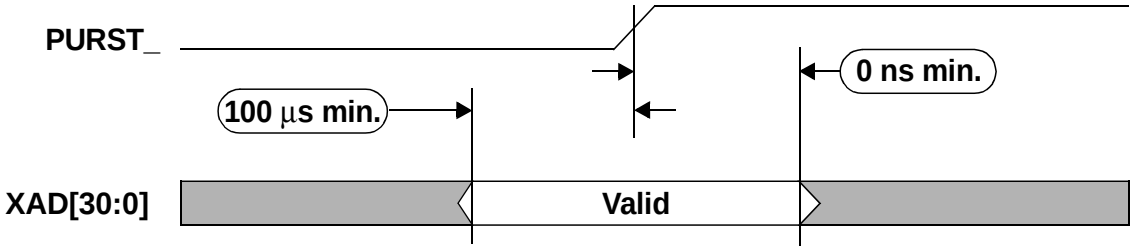


Figure 2-34. Power Up Reset Timing - Timing Group 1

Programming Model 3

Architectural Overview

The Harrier ASIC offers a variety of resources to the PowerPC and PCI address spaces. The placement and sizing of these resources is highly flexible, allowing the Harrier to support an unlimited number of possible resource mapping schemes. The figure below summarizes all of the resources offered by the Harrier.

Table 3-1. Harrier PowerPC and PCI Resources

Address Space	Resource	Size	Relocation		Page
			Options	Mechanism	
PowerPC	Control and Status Registers (*XCSR*)	1 KBytes	One of four selectable base addresses: $FEFF0000 $FEFF1000 $FEFF2000 $FEFF3000	Determined at power-up reset by sampling RD[x:y] pins	3-3
	SDRAM Banks A thru H	Variable from 32MB to 2GB	Any integer multiple of the corresponding bank size.	SDRAM Bank Addressing registers(**SDBAx**) within SDRAM Interface function of *XCSR*	3-30
	PCI Memory or IO Space, Ports 0 thru 3	Variable in multiples of 64K Byte	Any 64K Byte boundary in PowerPC space, translated to any 64K Byte boundary in PCI space	Outbound Translation registers (**OTADx** and **OTOFx**) within PowerPC to PCI Bridge function of *XCSR*	3-19

Table 3-1. Harrier PowerPC and PCI Resources (Continued)

Address Space	Resource	Size	Relocation		Page
			Options	Mechanism	
	PCI Configuration Space (*XCFS*)	Entire Configuration Space	Fixed **CONFIG_ADDR** and **CONFIG_DATA** registers within port 3 to PCI IO space	None	3-19
	MPIC (*XMPI*)	256 KBytes	Any 256K Byte boundary	**MBAR** register within MPIC function of *XCSR*	3-14
	Xport Channels 0 thru 3	Variable in multiples of 64KBytes	Any integer multiple of the corresponding device size (down to 64KBytes	**XPARx**, **XPATx** registers within Xport function of *XCSR*	3-150
PCI	Configuration Space (*PCFS*)	256 Bytes	Any Device Number	IDSEL pin	3-21
	PowerPC Memory Space, Ports 0 thru 3	Variable binary progression from 4K Bytes to 2G Bytes	Any corresponding block size boundary in PCI space, translated to any 64K Byte boundary in PowerPC space	Inbound Translation registers (**ITBAx**, **ITOFx**, and **ITSZx**) within *PCFS*	3-21
	Message Passing Group (*PMEP*)	4K Bytes	Any 4K Byte boundary	**MPBAR** register within *PCFS*	3-21

Register Group Summary

The following subsections describe the Harrier register groups in detail.

PowerPC Control and Status (*XCSR*) Register Group

The PowerPC Control and Status Register Group consumes a 1K byte block and contains control and status registers specific to the Harrier. This group resides within PowerPC address space and is byte, halfword, word, and doubleword accessible.

It is possible to place the base address of the *XCSR* Register Group at either $FEFF0000, $FEFF1000, $FEFF2000, or $FEFF3000. Having multiple choices for a base address allows the system designer to connect multiple Harrier ASICs in one system, and thus multiple PCI busses to one PowerPC bus. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information.

Generally the processor is responsible for maintaining this group of registers. It is possible however to setup an Inbound Translation Function that would place this register group as a PCI memory mapped resource thereby allowing access by external PCI masters.

This Register Group contains a reflection of the Harrier's PCI Configuration Space. A detailed description of each reflected configuration space register within the *XCSR* Register Group may be found within the description of the *PCFS* Register Group.

The Harrier's PowerPC Control and Status Register Group is shown in the following table.

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group

Function		Offset	Bits							
			0	7	8	15	16	23	24	31
Misc	IDs	$000	VENI				DEVI			
		$004			REVI					
		$008								
		$00C								
	Control and Status	$010	GCSR							
		$014	XCFR							
		$018	CT32							
		$01C	MCSR							
	General Purpose Registers	$020	GPRG0							
		$024	GPRG1							
		$028	GPRG2							
		$02C	GPRG3							
		$030	GPRG4							
		$034	GPRG5							
		$038								
		$03C								
Exceptions		$040	FEEN							
		$044	FEST							
		$048	FEMA							
		$04C	FECL							
		$050	EEEN							
		$054	EEST							
		$058	EECL							
		$05C	EEINT							
		$060	EEMCK0							
		$064	EEMCK1							

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function	Offset	Bits			
		0 7	8 15	16 23	24 31
Error Diagnostics	$068				
	$06C	EDEI			
	$070	EXAD			
	$074	EXAT			
	$078	EPAD			
	$07C	EPAT			
Watchdog Timers	$080	WT0C			
	$084	WT0S			
	$088	WT1C			
	$08C	WT1S			
Arbiters	$090	PARB			
	$094	XARB			
	$098				
	...etc...				
	$0BC				
UARTs	$0C0	RTDL0	IEDH0	IDFC0	LCTL0
	$0C4	MCTL0	LSTA0	MSTA0	SCRT0
	$0C8	RTDL1	IEDH1	IDFC1	LCTL1
	$0CC	MCTL1	LSTA1	MSTA1	SCRT1
	$0D0	UCTL			UPS
	$0D4				
	$0D8				
	$0DC				

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
MPIC		$0E0	MBAR			
		$0E4	MCSR		MIRS	
		$0E8				
		...etc...				
		$0FC				
SDRAM Interface	Control	$100	SDGC			
		$104	SDTC			
		$108				
		$10C				
	Address	$110	SDBAA			
		$114	SDBAB			
		$118	SDBAC			
		$11C	SDBAD			
		$120	SDBAE			
		$124	SDBAF			
		$128	SDBAG			
		$12C	SDBAH			
	Scrub	$130	SDSC			
		$134	SDSA			
		$138				
		$13C				
	Error Log	$140	SDSES			
		$144	SDSEA			
		$148		SDMES		
		$14C	SDMEA			

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
Xport	Address & Attributes	$150	XPAR0			
		$154	XPAT0			
		$158	XPAR1			
		$15C	XPAT1			
		$160	XPAR2			
		$164	XPAT2			
		$168	XPAR3			
		$16C	XPAT3			
	General Control	$170	XPGC			
		$174				
		$178				
		$17C				
I2C0		$180			I2PS0	
		$184				I2CO0
		$188				
		$18C				I2TD0
		$190				
		$194				I2ST0
		$198				
		$19C				I2RD0

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function	Offset	Bits			
		0 7	8 15	16 23	24 31
I2C1	$1A0			I2PS1	
	$1A4				I2CO1
	$1A8				
	$1AC				I2TD1
	$1B0				
	$1B4				I2ST1
	$1B8				
	$1BC				I2RD1
Reserved	$1C0				
	...etc...				
	$1FC				

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function		Offset	Bits							
			0	7	8	15	16	23	24	31
PowerPC to PCI Bridge	Control and Status	$200	BPCS							
		$204	BXCS							
		$208								
		$20C								
	Interrupt Ack	$210	PIAC							
		$214								
		$218								
		$21C								
	Address Translation	$220	OTAD0							
		$224	OTOF0				OTAT0			
		$228	OTAD1							
		$22C	OTOF1				OTAT1			
		$230	OTAD2							
		$234	OTOF2				OTAT2			
		$238	OTAD3							
		$23C	OTOF3				OTAT3			
		$240								
		$244								
		$248	PSAD							
		$24C	PSOF				PSSZ		PSAT	

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function	Offset	Bits							
		0	7	8	15	16	23	24	31
DMA	$250	DCTL							
	$254	DSTA							
	$258								
	$25C								
	$260	DSAD							
	$264	DSAT							
	$268	DDAD							
	$26C	DDAT							
	$270	DNLA							
	$274	DCNT							
	$278								
	$27C								
	$280	DCSA							
	$284	DCDA							
	$288	DCLA							
	$28C								

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function		Offset	Bits							
			0	7	8	15	16	23	24	31
Message Passing	Generic	$290	MGOM0							
		$294	MGOM1							
		$298	MGOD							
		$29C								
		$2A0	MGIM0							
		$2A4	MGIM1							
		$2A8	MGID							
		$2AC								
		$2B0	MGIDM							
		$2B4								
		$2B8								
		$2BC								
	I_2O	$2C0	MIOFH							
		$2C4	MIOFT							
		$2C8	MIOPH							
		$2CC	MIOPT							
		$2D0	MIIFH							
		$2D4	MIIFT							
		$2D8	MIIPH							
		$2DC	MIIPT							
		$2E0	MICT							
		$2E4	MIQB							
		$2E8								
		$2EC								

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
Reserved		$2F0				
		$2F4				
		$2F8				
		$2FC				
PowerPC to PCI Bridge	Reflected PCI Config Space	$300	VENI		DEVI	
		$304	CMMD		STAT	
		$308	REVI	CLC		
		$30C	CLS	MLT	HDT	
		$310	MPBAR			
		$314	ITBAR0			
		$318	ITBAR1			
		$31C	ITBAR2			
		$320	ITBAR3			
		$324				
		$328				
		$32C	SUBI		SUBV	
		$330				
		$334				
		$338				
		$33C	INTL	INTP	MNGN	MXLA
		$340				
		$344	MPAT			
		$348	ITSZ0		ITOF0	
		$34C	ITAT0			
		$350	ITSZ1		ITOF1	

Table 3-2. PowerPC Control and Status (*XCSR*) Register Group (Continued)

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
PowerPC to PCI Bridge	Reflected PCI Config Space	$354	ITAT1			
		$358	ITSZ2		ITOF2	
		$35C	ITAT2			
		$360	ITSZ3		ITOF3	
		$364	ITAT3			
		...etc...				
		$380				PSTA
		$384	PGPR			
		...etc...				
		$3FC				
		$380				PSTA
		$384	PGPR			
		...etc...				
		$3FC				
General Purpose Memory		$400				
		$404				
		...etc...				
		$BFC				

PowerPC Multi-Processor Interrupt Controller (*XMPI*) Register Group

The PowerPC Multi-Processor Interrupt Controller (*XMPI*) Register Group is a relocatable 256K Byte block of registers pertaining to the MPIC function. The base address of this group is programmable using the **MBAR** register within the *XCSR* Register Group.

The registers in the Harrier PowerPC Multi-Processor Interrupt Controller Register Group are listed in the following table, along with each register's bit span and offset location.

Table 3-3. PowerPC Multi-Processor Interrupt Controller (*XMPI*) Register Group

Function		Offset	Bits							
			0	7	8	15	16	23	24	31
MPIC	Misc.	$01000	FREP							
		$01010								
		$01020	GLBC							
		$01030								
		...etc...								
		$01070								
		$01080	VENI							
		$01090	PINT							
	IPI	$010A0	IPVP0							
		$010B0	IPVP1							
		$010C0	IPVP2							
		$010D0	IPVP3							
	Spur	$010E0							SPVE	
	Timers	$010F0	TIFR							
		$01100	TICC0							
		$01110	TIBC0							
		$01120	TIVP0							
		$01130	TIDE0							
		$01140	TICC1							
		$01150	TIBC1							
		$01160	TIVP1							
		$01170	TIDE1							
		$01180	TICC2							
		$01190	TIBC2							
		$011A0	TIVP2							
		$011B0	TIDE2							

Table 3-3. PowerPC Multi-Processor Interrupt Controller (*XMPI*) Register Group (Continued)

Function		Offset	Bits							
			0	7	8	15	16	23	24	31
MPIC	Timers	$011C0	TICC3							
		$011D0	TIBC3							
		$011E0	TIVP3							
		$011F0	TIDE3							
	External	$10000	EXVP0							
		$10010	EXDE0							
		$10020	EXVP1							
		$10030	EXDE1							
		$10040	EXVP2							
		$10050	EXDE2							
		$10060	EXVP3							
		$10070	EXDE3							
		$10080	EXVP4							
		$10090	EXDE4							
		$100A0	EXVP5							
		$100B0	EXDE5							
		$100C0	EXVP6							
		$100D0	EXDE6							
		$100E0	EXVP7							
		$100F0	EXDE7							
		$10100	EXVP8							
		$10110	EXDE8							
		$10120	EXVP9							
		$10130	EXDE9							
		$10140	EXVP10							
		$10150	EXDE10							

Table 3-3. PowerPC Multi-Processor Interrupt Controller (*XMPI*) Register Group (Continued)

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
MPIC	External	$10160	EXVP11			
		$10170	EXDE11			
		$10180	EXVP12			
		$10190	EXDE12			
		$101A0	EXVP13			
		$101B0	EXDE13			
		$101C0	EXVP14			
		$101D0	EXDE14			
		$101E0	EXVP15			
		$101F0	EXDE15			
	Internal	$10200	IFEVP			
		$10210	IFEDE			
		$10220	IEEVP			
		$10230	IEEDE			
	Reserved	$10240				
		...etc...				
		$20020				
	CPU 0	$20040	P0IPD0			
		$20050	P0IPD1			
		$20060	P0IPD2			
		$20070	P0IPD3			
		$20080	P0CTP			
		$20090				
		$200A0				P0IAC
		$200B0				P0EOI

Table 3-3. PowerPC Multi-Processor Interrupt Controller (*XMPI*) Register Group (Continued)

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
MPIC	Reserved	$200C0				
		...etc...				
		$21020				
	CPU 1	$21040	P1IPD0			
		$21050	P1IPD1			
		$21060	P1IPD2			
		$21070	P1IPD3			
		$21080	P1CTP			
		$21090				
		$210A0				P1IAC
		$210B0				P1EOI
	Reserved	$210C0				
		...etc...				
		$3FFFF				

PowerPC to PCI Configuration Space (*XCFS*) Register Group

The generation of PCI configuration cycles is supported by the Harrier using Configuration Mechanism #1 as specified in *PCI Local Bus Specification 2.1*. This mechanism is tied to the Outbound Translation Function 3. This translation function only offers PCI I/O space to PowerPC address space mapping. The **CONFIG_ADDRESS** and **CONFIG_DATA** registers are located within this translation space at the first occurrence (i.e. closest to the base address) of the $0CF8 and $0CFC offsets.

The following table shows the location of the **CONFIG_ADDRESS** and **CONFIG_DATA** registers with respect to the PowerPC bus. Note that since these registers are considered PCI resources, Endian conversion rules are applicable. This table represents these registers in Big Endian mode.

Table 3-4. PowerPC to PCI Configuration Space (*XCFS*) Register Group

Function		Offset	Bits			
			0 7	8 15	16 23	24 31
PowerPC to PCI Bridge	PCI I/O Space	$0000				
		...etc...				
		$0CF4				
	Config Mechanism	$0CF8	CONFIG_ADDRESS			
		$0CFC	CONFIG_DATA			
	PCI I/O Space	$0D00				
		...etc...				
		$FFFF				

PCI Configuration Space (*PCFS*) Register Group

The PCI Configuration Space (*PCFS*) Register Group is a PCI compliant single function configuration register group. This 256 byte block contains various control and status registers specific to the Harrier, including base address registers for the Inbound Translation Functions and for the *PMEP* Register Group. The external connection of IDSEL to an AD line establishes the connection between Harrier's device number and it's configuration space.

The PCI Configuration Register Group is compliant with the configuration register set described in the *PCI Local Bus Specification, Revision 2.1*. The Harrier is a single function device, and consequently offers a Single Function Header Type 00 for Function 0.

All write operations to reserved registers are treated as no-ops. That is, the access is completed normally on the bus and the data is discarded. Read accesses to reserved or un-implemented registers are completed normally and a data value of 0 is returned.

A reflection of this Register Group is contained within the *XCSR* Register Group. The description of each register within the *PCFS* Register Group will be accompanied by the appropriate register from the reflected configuration space within the *XCSR* Register Group.

The Harrier PCI Configuration Space Register Group is shown in the following table.

Table 3-5. PCI Configuration Space (*PCFS*) Register Group

Function		Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0	Offset
PowerPC to PCI Bridge	Header	DEVI		VENI		$00
		STAT		CMMD		$04
		CLAS			REVI	$08
			HEAD	MLAT	CLSZ	$0C
		MPBAR				$10
		ITBAR0				$14
		ITBAR1				$18
		ITBAR2				$1C
		ITBAR3				$20
						$24
						$28
		SUBI		SUBV		$2C
						$30
						$34
						$38
		MXLA	MNGN	INTP	INTL	$3C
	Address Translation					$40
			MPAT			$44
		ITOF0			ITSZ0	$48
			ITAT0			$4C
		ITOF1			ITSZ1	$50

Table 3-5. PCI Configuration Space (*PCFS*) Register Group (Continued)

Function		Bits				Offset
		31 24	23 16	15 8	7 0	
			ITAT1			$54
		ITOF2			ITSZ2	$58
			ITAT2			$5C
		ITOF3			ITSZ3	$60
			ITAT3			$64
	Reserved					...etc...
	Status	PSTA				$80
	GP Reg	PGPR				$84
	Reserved					...etc...
						$FF

PCI Message Passing (*PMEP*) Register Group

The Harrier provides a group of PCI Memory mapped resources for Message Passing. This group is a 4K Byte block that contains both I_2O and generic message passing functions. The I_2O components consists of inbound and outbound circular queue ports and interrupt status/control registers. These components are fully compliant with the *Intelligent I/O (I_2O) Architecture Specification.* The generic components are the PCI bus mapped complimentary equivalents to the Message Passing functions found within the *XCSR* Register Group.

The mapping of this group is controlled by the **MPBAR** register within the *PCFS* Register Group.

The Harrier PCI Message Passing Register Group is shown in the following table.

Table 3-6. PCI Message Passing (*PMEP*) Register Group

Function		Bits				Offset
		31 24	23 16	15 8	7 0	
Message Passing	I_2O					\$000
						...etc...
						\$02C
					MIST	\$030
					MIMS	\$034
						\$038
						\$03C
		MIIQ				\$040
		MIOQ				\$044
						\$048
						...etc...
						\$0FC
	Generic	MGOM0				\$100
		MGOM1				\$104
		MGOD				\$108
						\$10C
		MGIM0				\$110
		MGIM1				\$114
		MGID				\$118
						\$11C
					MGST	\$120
					MGMS	\$124
		MGODM				\$128
						\$12C
	Reserved					\$130
						...etc...
						\$FFF

SDRAM Interface

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

For a functional description, refer to the section titled *SDRAM Interface* on page 2-4.

SDRAM General Control Register

Offset	*XCSR* + $100																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDGC																																SDRAM Interface
			MXRR		DREF	DERC	RWCB	ENRV				SWVT																				
Operation	R	R	R/W	R/W	R	R/W	R/W	R/W	R/W	R	R	R	R/W	R	R	R	R								R								
Reset	0	0	0-P	1-P	0	0	1	0	0	0	0	0	0	0	0	0	$00								$00								

The SDRAM General Control Register (**SDGC**) affects functions that apply to all banks of SDRAM. The fields in the **SDGC** register are defined as follows:

MXRR: **Multiply the Refresh Rate**. When set, the Harrier increases the effective refresh rate as shown in the table below. For an example of programming these bits, see the section titled *Optional Method for Sizing SDRAM* on page 5-9.

Table 3-7. MXRR Control of Refresh Rate

MXRR	4-Row Refresh Interval	Effective Refresh Rate
00	62us	1 Row/15.5us
01	31us (default)	1 Row/7.75us
10	15us	1 Row/3.75us
11	7us	1 Row/1.75us

DREF: **Disable Refresh**. When set, the Harrier stops refreshing SDRAM. When cleared, the Harrier resumes refreshing SDRAM. This bit should not be set during normal SDRAM operation.

DERC: **Disable Error Correction**. When set, the Harrier does not correct SDRAM single-bit errors. When cleared, the Harrier does correct single-bit errors.

RWCB: **Read/Write Checkbits**. When set, the Harrier alters SDRAM accesses so that they read/write checkbit data rather than normal data. When cleared, the Harrier does not alter SDRAM accesses. The following table gives an example.

Table 3-8. SDTC RWCB Example

PowerPC Address	Unaltered view of Data (RWCB cleared)	Altered view of Data (RWCB set)	
	Bits 0-63	Bits 0-7	Bits 8-63
$00000000	$00000000	Checkbits for $0000000	Undefined
$00000008	$01234567	Checkbits for $01234567	Undefined
$00000010	$AAAAAAAA	Checkbits for $AAAAAAAA	Undefined
$00000018	$BBBBBBBB	Checkbits for $BBBBBBBB	Undefined
...	...	...	...

ENRV: **Enable Reset Vector**. When set, the Harrier redirects PowerPC accesses in the reset vector address range so that they go to whichever SDRAM bank is mapped at $00000000. When cleared, the Harrier does not redirect accesses in the reset vector address range.

SWVT: **Swap Vector table**. When set, the Harrier selectively swaps the first and second 16KBytes of whichever bank of SDRAM is mapped at $00000000. When cleared, the Harrier does not swap SDRAM. The selection for swapping is that the Harrier swaps when

this bit is set and when the current PowerPC master is identified by the Harrier as CPU 1. See ***XCSR*.GCSR**.MID in the section titled *Global Control and Status Register* on page 3-194.

SDRAM Timing Control Register

Offset	*XCSR* + $104																													Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDTC																																SDRAM Control
				CL3		TRC					TRAS			WDPL	TDP				TRP				TRCD							SDER		
Operation	R	R	R	R/W	R	R/W	R/W	R/W	R	R	R/W	R/W	R	R	R/W	R/W	R	R	R	R/W	R	R	R	R/W	R	R	R	R	R	R	R	R/W	
Reset	0	0	0	1-P	0	0-P	1-P	1-P	0	0	1-P	1-P	0	0	1-P	1-P	0	0	0	1-P	0	0	0	1-P	0	0	0	0	0	0	0	V-P	

The SDRAM Timing Control Register (**SDTC**) affects the timing for all banks of SDRAM. The fields within the **SDTC** register are defined as follows:

CL3: **Cas Latency 3**. When set, the Harrier accesses SDRAM assuming a cas latency of 3. When cleared, the Harrier accesses SDRAM assuming a cas latency of 2. Note that the Harrier performs a “register set” command to SDRAM anytime software changes CL3. Also note that power-up reset is the only kind of reset that affects this bit.

TRC: **SDRAM timing parameter tRC**. TRC determines the minimum number of CLK cycles the Harrier uses to satisfy the SDRAM’s tRC parameter. The following table shows the encoding.

.

Table 3-9. SDTC TRC Encoding

TRC	Time for tRC
000	8 CLK's
001	9 CLK's
010	10 CLK's
011	11 CLK's
100	reserved
101	reserved
110	6 CLK's
111	7 CLK's

Note Power-up reset is the only type of reset that affects this field.

TRAS: **SDRAM timing parameter tRAS**. This field determines the minimum number of CLK cycles the Harrier uses to satisfy the SDRAM's tRAS parameter. The table below shows the encoding. Note that power-up reset is the only kind of reset that affects this field.

Table 3-10. SDTC TRAS Encoding

TRAS	Time for tRAS
00	4 CLK's
01	5 CLK's
10	6 CLK's
11	7 CLK's

WDPL: **Wait on tDPL**. When set, the Harrier always waits at least 4 CLK's after the write command portion of a single-beat write before precharging SDRAM. When cleared, the Harrier does not impose the wait. Software should always set WDPL to 1. Note that power-up reset is the only kind of reset that affect this bit.

TDPL: **SDRAM timing parameter tDPL**. TDPL determines the minimum number of CLK cycles the Harrier uses to satisfy the SDRAM's tDPL parameter. The following table shows the bit encoding. Note that power-up reset is the only kind of reset that affects this bit.

Table 3-11. SDTC TDPL Encoding

TDPL	Time for tDPL
0	1 CLK's
1	2 CLK's

TRP: SDRAM timing parameter tRP. TRP determines the minimum number of CLK cycles the Harrier uses to satisfy the SDRAM's tRP parameter. The table below shows the bit encoding. Note that power-up reset is the only kind of reset that affects this bit.

Table 3-12. SDTC TRP Encoding

TRP	Time for tRP
0	2 CLK's
1	3 CLK's

TRCD: SDRAM timing parameter tRCD. TRCD determines the minimum number of CLK cycles the Harrier uses to satisfy the SDRAM's tRCD parameter. The table below shows the bit encoding. Note that power-up reset is the only kind of reset that affects this bit.

Table 3-13. SDTC TRCD Encodiing

TRCD	Time for tRCD
0	2 CLK's
1	3 CLK's

SDER: **SDRAM External Registers or Buffers**. When SDER is set, the Harrier increases the setup time for SDRAM command/address to CS by one CLK period. Note that SDER initializes at power-up reset to match the values on a certain signal pin. Refer to the section titled *Hardware Configuration* on page 2-133 for more information.

SDRAM Bank (A,B,C,D,E,F,G, and H) Addressing Registers

Offset	SDBA: *XCSR* + $110 SDBB: *XCSR* + $114 SDBC: *XCSR* + $118 SDBD: *XCSR* + $11C SDBE: *XCSR* + $120 SDBF: *XCSR* + $124 SDBG: *XCSR* + $128 SDBH: *XCSR* + $12C	
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	**Function**
Name	SDBA's	**SDRAM Address**
	BASE — SIZE — ENB	
Operation	R/W — R R R R R/W — R R R R R R R R/W — R	
Reset	$00 — 0 0 0 0 $0-P — 0 0 0 0 0 0 0 0 — $00	

The SDRAM Bank Addressing Registers (**SDBAA, SDBAB, SDBAC...**) provide base address, size, and enable for each corresponding SDRAM bank (A,B,C...). The fields within the **SDBAx** registers are defined as follows:

BASE: **Bank Base Address**. This field determines the bank's base address. The Harrier uses only those bits in BASE that correspond to integer multiples of the bank's size. Consequently, larger sized banks do not use BASE's lower significant bits.

SIZE: **Bank Size**. SIZE tells the Harrier the bank's component configuration and size. The table below shows SIZE's encoding. Note that power-up reset is the only kind of reset that affects this field.

Table 3-14. SDBA SIZE Encoding

SIZE	Component configuration	Bank Size	Components per Bank	Technology
0000	-	0MBytes	-	-
0001	4Mx16	32MBytes	5	64Mbit
0010	8Mx8	64MBytes	9	64Mbit
0011	8Mx16	64MBytes	5	128Mbit
0100	16Mx4	128MBytes	18	64Mbit
0101	16Mx8	128MBytes	9	128Mbit
0110	16Mx16	128MBytes	5	256Mbit
0111	32Mx4	256MBytes	18	128Mbit
1000	32Mx8	256MBytes	9	256Mbit
1001	32Mx16	256MBytes	5	512Mbit
1010	64Mx4	512MBytes	18	256Mbit
1011	64Mx8	512MBytes	9	512Mbit
1100	64Mx16	512MBytes	5	1Gbit
1101	128Mx4	1024MBytes	18	512Mbit
1110	128Mx8	1024MBytes	9	1Gbit
1111	256Mx4	2048MBytes	18	1Gbit

ENB: **Bank Enable**. When set, the Harrier allows PowerPC accesses to the bank. When cleared, the Harrier disallows PowerPC accesses to the bank. Note that ENB does not affect scrub accesses to the bank.

SDRAM Scrub Control Register

Offset	*XCSR* + $130																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDSC																																SDRAM Scrubbing
	SCWE						SCCNT		SCPA																								
Operation	R/W	R	R	R	R	R	R	R	R/W								R								R								
Reset	0-P	0	0	0	0	0	0-P	0-P	$00-P								$00								$00								

The SDRAM Scrub Control Register (**SDSC**) controls the scrubber. The fields within the **SDSC** register are defined as follows:

SCWE: **Scrub Write Enable**. When set, the Harrier writes corrected data during scrubs that detect single-bit errors. When cleared, the Harrier does not write during scrubs. Note that power-up reset is the only kind of reset that affects this bit.

SCCNT: **Scrub Counter**. SCCNT increments each time the Harrier completes a scrub of the entire SDRAM array. When SCCNT reaches $3, it rolls over to $0 and continues. Note that power-up reset is the only kind of reset that affects this field.

SCPA: **Scrub Prescaler Adjust**. SCPA determines the Harrier's scrub rate by setting the roll-over count for the scrub prescale counter. Each time the Harrier performs a refresh burst, the scrub prescale counter increments by one. When the scrub prescale counter reaches the value stored in SCPA, it clears and resumes counting from 0. Note that when SCPA is all 0's, the scrub Prescale counter does not increment, which disables scrubs from occurring. Since SCPA clears to 0's at power-up reset, scrubbing comes up disabled until software programs a non-zero value into SCPA. Power-up reset is the only kind of reset that affects this field.

SDRAM Scrub Address Counter

Offset	*XCSR* + $134																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDSA																																SDRAM Scrubbing
Operation	R	R/W																												R	R	R	
Reset	0	$00000000-P																												0	0	0	

The SDRAM Scrub Address Counter (**SDSA**) provides the scrub address for all banks of SDRAM. **SDSA**'s one and only field is defined as follows:

SDSA: SDRAM Scrub Address. This field is the scrub address counter. SDSA increments by eight each time the Harrier completes one scrub to all banks. Bits 0, and 29-31 are always 0's. When SDSA reaches all $7fff fff8, it rolls over to 0 and continues counting. SDSA's most significant bits are meaningful only for those banks whose SDRAM devices are large enough to require them. Note that power-up reset is the only kind of reset that affects SDSA.

SDRAM Single-bit Error Status

Offset	*XCSR* + $140																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDSES																																SDRAM Error Logging
	ESYN												EOS	ESB			SECNT																
Operation	R								R	R	R	R	R	R	R	R	R/W								R								
Reset	$00-P								0	0	0	0	0-P	0-P	0-P	0-P	$00-P								$00								

The SDRAM Single-bit Error Status Register (**SDSES**) provides status for logged SDRAM single-bit errors. It also provides a single-bit-error counter. The fields within the **SDSES** register are defined as follows:

ESYN: **Error Syndrome**. ESYN reflects the syndrome value at the last logging of a single-bit error (refer to the "Error Exception Enable Register" section further on in this chapter). Software can use ESYN together with syndrome codes as listed in the "Data" section of the following chapter to determine which bit was in error. Note that power-up reset is the only kind of reset that affects this field.

EOS: **Error On Scrub**. EOS set indicates that the Harrier was scrubbing at the last single-bit error logging. EOS cleared indicated that the Harrier was not scrubbing.

ESB: **Error Scrub Bank**. ESB indicates which bank the scrubber was accessing when/if the Harrier logged a scrub single-bit error. The following table shows the encoding. Note that power-up reset is the only kind of reset that affects this field.

Table 3-15. SDSES.ESB Encoding

ESB	Bank
000	A
001	B
010	C
011	D
100	E
101	F
110	G
111	H

SECNT: **Single-bit Error Count**. This field is the single-bit error counter. Each time the Harrier detects a single-bit error SECNT increments by one. SECNt always increments on detected single-bit errors regardless of whether or not error logging is enabled. SECNT’s rolling over from $FF to $00 generates a single-bit error counter

exception if so enabled (Refer to the section titled *Error Exception Enable Register* on page 3-170). Note that power-up reset is the only kind of reset that affects this field.

Note: When the Harrier reports a single bit error, software can use the syndrome that was logged by Harrier to determine which bit was in error. The following table shows the syndrome for each possible single bit error.

Table 3-16. Syndrome Code Ordered by Bit in Error

Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome
rd0	$4A	rd16	$92	rd32	$A4	rd48	$29	ckd0	$01
rd1	$4C	rd17	$13	rd33	$C4	rd49	$31	ckd1	$02
rd2	$2C	rd18	$0B	rd34	$C2	rd50	$B0	ckd2	$04
rd3	$2A	rd19	$8A	rd35	$A2	rd51	$A8	ckd3	$08
rd4	$E9	rd20	$7A	rd36	$9E	rd52	$A7	ckd4	$10
rd5	$1C	rd21	$07	rd37	$C1	rd53	$70	ckd5	$20
rd6	$1A	rd22	$86	rd38	$A1	rd54	$68	ckd6	$40
rd7	$19	rd23	$46	rd39	$91	rd55	$64	ckd7	$80
rd8	$25	rd24	$49	rd40	$52	rd56	$94		
rd9	$26	rd25	$89	rd41	$62	rd57	$98		
rd10	$16	rd26	$85	rd42	$61	rd58	$58		
rd11	$15	rd27	$45	rd43	$51	rd59	$54		
rd12	$F4	rd28	$3D	rd44	$4F	rd60	$D3		
rd13	$0E	rd29	$83	rd45	$E0	rd61	$38		
rd14	$0D	rd30	$43	rd46	$D0	rd62	$34		
rd15	$8C	rd31	$23	rd47	$C8	rd63	$32		

SDRAM Single-bit Error Address Register

Offset	*XCSR* + $144																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDSEA																																SDRAM Error Logging
Operation	R																													R	R	R	
Reset	$00000000 P																													0	0	0	

The SDRAM Single-bit Error Status Register (**SDSEA**) reflects the address present when the Harrier last logged a single-bit error. The **SDSEA**'s one and only field is defined as follows:

SDSEA: **SDRAM Single-bit Error Address**. This field contains the address of the last single-bit error logged by the Harrier (refer to the section titled *Error Exception Enable Register* on page 3-170 for more information). If the error was due to a PowerPC access, SDSEA matches the value that was on PowerPC address signals 0-28. If the error was due to a scrub, SDSEA matches the value that was in the scrub address counter. Bits 29-31 are always 0's. Note that power-up reset is the only kind of reset that affects SDSEA.

SDRAM Multi-bit Error Status

Offset	*XCSR* + $148																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name									SDMES																								SDRAM Error Logging
															ESB																		
Operation	R								R	R	R	R	R	R	R	R	R								R								
Reset	$00								0	0	0	0	0	0-P	0-P	0-P	$00								$00								

The SDRAM Multi-bit Error Status Register (**SDMES**) provides status for logged SDRAM multi-bit errors. **SDMES's** one and only field is defined as follows:

ESB: **Error Scrub Bank**. ESB indicates which bank the scrubber was accessing when/if the Harrier logged a scrub multi-bit error (refer to the section titled *Error Exception Enable Register* on page 3-170). The table on the previous page (SDSES.ESB Encoding) shows the encoding. Note that power-up reset is the only kind of reset that affects this field.

SDRAM Multi-bit Error Address Register

Offset	*XCSR* + $14C																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SDMEA																																SDRAM Error Logging
Operation	R																													R	R	R	
Reset	$00000000 P																													0	0	0	

The SDRAM Multi-bit Error Address Register (**SDMEA**) reflects the address present when the Harrier last logged a multi-bit error. **SDMEA**'s one and only field is defined as follows:

SDMEA: **SDRAM Multi-bit Error Address**. This field contains the address of the last multi-bit error logged by the Harrier (refer to the section titled *Error Exception Enable Register* on page 3-170). If the error was due to a PowerPC access, SDMEA matches the value that was on PowerPC address signals 0-28. If the error was due to a scrub, SDMEA matches the value that was in the scrub address counter. Bits 29-31 are always 0's. Note that power-up reset is the only kind of reset that affects SDMEA.

PowerPC to PCI Bridge

The PowerPC to PCI Bridge has registers located within three register groups. Most of the control and status is contained within the PowerPC mapped *XCSR* Register Group. Processor access to PCI Configuration Space is supported using PowerPC mapped registers within the *XCFS* Register Group. Finally, a standard PCI Configuration Space interface is provided within the *PCFS* Register Group.

XCSR Register Group

This section does not cover the reflected PCI Configuration Space registers. Please refer to the section titled *PCI Configuration Space (PCFS) Register Group* on page 3-21 for a detailed description of all *XCSR* Register Group registers applicable to the reflected PCI Configuration Space.

Bridge PCI Control and Status Register

Offset	*XCSR* + $200																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	BPCS																																Control and Status
	OFBR	DLR	HIL	MRC	DRO		PIM				CSM	CSH				P64																
Operation	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R	R	R/W	R/W	R	R	R	R	R								R								
Reset	0	0	0	0	0	0	0	0	0	0	x	x	0	0	0	x	$00								$00								

The Bridge PCI Control and Status Register (**BPCS**) provides control and status information associated with PCI functions. The fields within the **BPCS** are defined as follows:

OFBR: Outbound Flush Before Read. If set, the Harrier guarantees that all outbound write-posted transactions are completed before any inbound read transactions are allowed to complete. If cleared, there is

no correlation between these transaction types and their order of completion. Please refer to the section titled "Transaction Ordering" in the previous chapter for more information.

DLR: Disregard Latency Requirements. If set, the Harrier does not honor PCI initial and subsequent latency requirements. If an access exceeds these requirements, the Harrier does not initiate a disconnect and inserts wait states as needed to complete a transaction. If cleared, the Harrier complies with PCI latency requirements. Users are strongly encouraged to leave the DLR field cleared. Please refer to the section titled "PCI Slave" in the previous chapter for more information.

HIL: Host Bridge Initial Latency. This field is only applicable if the DLR field is cleared. If set, the Harrier implements a 32 clock (i.e. cache hit) host bridge initial latency. If cleared, the Harrier implements a 16 clock non-host bridge initial latency.

MRC: Maximum Retry Count. This field enables a 2^{24} counter in both the DMA and Bridge PCI masters that keep track of the number of sequential retry attempts. If this field is enabled and the maximum number of sequential retries is exceeded, then an error exception is generated and the transaction is aborted. If this field is cleared, then there is no limit to the number of retry attempts. Note that this field must remain cleared to be fully PCI Local Bus compliant.

DRQ: Disregard REQ64_ Qualification. This field is used to modify the qualifications that the PCI slave makes when determining whether or not a transaction is considered part of a previously disconnected delayed transaction. If this field is set, the PCI slave does not consider the latched state of REQ64_ as part of the qualification. If the field is cleared, then the state of REQ64_ must be matched exactly in order to be qualified as the continuation of a delayed transaction. Note that this field must remain cleared to be fully PCI Local Bus compliant.

PIM: PCI Interrupt Mapping. These bits indicate which PCI interrupt signal line a Harrier generated PCI interrupt is routed to. The options available are shown in follwing table.

Table 3-17. BPCS PIM Encoding

PIM	PCI Interrupt
00	INTA_
01	INTB_
10	INTC_
11	INTD_

CSM: Configuration Space Mask. This field controls the visibility of the Harrier's PCI configuration space located above the predefined 64 byte header. Specifically, visibility is affected from offset $40 to offset $FF within the *PCFS* Register Group. Access to these offsets from within the *XCSR* Register Group are not affected by this field.

If set, writes to this range has no effect, and reads return all zeros. If cleared, writes to and reads from this range occur in a normal fashion.

The default state of this bit is determined at the release of reset. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information.

CSH: Configuration Space Holdoff. This field controls the visibility of the Harrier's PCI configuration space. The processor can use this field to hold off accesses to the Harrier's configuration space until after the inbound address and attribute fields have been established.

If set, all attempts to access the Harrier's configuration space results in a disconnect-retry. The retry is always correct at the 16 clock maximum initial latency for a non-host bridge device that has expired. If cleared, writes to and reads from the Harrier's configuration space are completed as normal.

The default state of this bit is determined at the release of reset. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information.

P64: PCI 64-bit. If set, the Harrier is connected to a 64-bit PCI bus. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information on how this field gets set.

Bridge PowerPC Control and Status Register

Offset	*XCSR* + $204																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	BXCS																																Control and Status
	IFBR	BHG	RSF	SSF	RBT		SBT				P1H	P0H	CSF																				
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R	R	R	R								R								
Reset	0	0	1	1	0	0	0	0	0	0	V-L	V-L	1	0	0	0	$00								$00								

The Bridge PowerPC Control and Status Register (**BXCS**) provides control and status information associated with PowerPC functions. The fields within the **BXCS** are defined as follows:

IFBR: Inbound Flush Before Read. If set, the Harrier guarantees that all inbound write-posted transactions are completed before any outbound read transactions are allowed to complete. If cleared, there is no correlation between these transaction types and their order of completion. Please refer to the section titled "Transaction Ordering" in the previous chapter for more information.

BHG: Bus Hog. If set, the Harrier operates in Bus Hog mode. Bus Hog mode means the Harrier continually requests the PowerPC bus for the entire duration of each transfer. If cleared, the Harrier requests the bus in a normal manner. Please refer to the section titled "PPC Master" in the previous chapter for more information.

RSF: Read-Ahead Sync Flush. If set, a read-ahead session is closed and the FIFO cleared if the Harrier detects a Sync cycle from the processor responsible for originating the read-ahead. If cleared, the read-ahead session remains open when a Sync is detected.

SSF: Store-Gather Sync Flush. If set, a Store-Gather collection is flushed if the Harrier detects a Sync cycle originating from the processor responsible for a collection. If cleared, a collection is not flushed when a Sync is detected.

RBT: Read-Ahead Backup Timer. This field specifies the maximum time in processor clock cycles that may take place between qualified reads when using read-ahead. Please refer to the section titled "Read Ahead" in the previous chapter for more information. The options available are shown in .the following table.

Table 3-18. BXCS RBT Encoding

RBT	Time Out Length
00	32 clocks
01	64 clocks
10	256 clocks
11	disabled

SBT: Store-Gather Backup Timer. This field specifies the maximum time in processor clock cycles that may take place between collectable writes when using Store-Gather. Please refer to the section titled *PPC Master* on page 2-33 for more information. The options available are shown in .the following table.

Table 3-19. BXCS SBT Encoding

SBT	Time Out Length
00	32 clocks
01	64 clocks
10	256 clocks
11	disabled

P1H: Processor 1 Holdoff. This field is used to hold processor 1 in a reset state (HRST1_ asserted) following a local bus reset (RST_ asserted). This field may be used to allow a PCI master to program the Harrier (using a wrapped back Inbound Translation Function) before allowing the processor to start code execution.

If set, the HRST1_ signal is held in the asserted state. If cleared, the state of the HRST1_ signal is determined RST_.

The default state of this bit is determined at the release of reset. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information.

P0H: Processor 0 Holdoff. This field is used to hold processor 0 in a reset state (HRST0_ asserted) following a local bus reset (RST_ asserted). This field may be used to allow a PCI master to program the Harrier (using a wrapped back Inbound Translation Function) before allowing the processor to start code execution.

If set, the HRST0_ signal will be held in the asserted state. If cleared, the state of the SRST0_ signal is determined by RST_.

The default state of this bit is determined at the release of reset. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information.

CSE: Copy-back Snarfing Enable. This field is used to enable snarfing on a processor's copy-back cycle. See the section titled *Copyback Snarfing* on page 2-35 for more information. If set, copy-back snarfing will be attempted for inbound read cycles. If cleared, snarfing will not be attempted

PCI Interrupt Acknowledge Register

Offset	***XCSR* + $210**																														**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	PIAC																																**Interrupt Acknowledge**
Operation	R																																
Reset	$00000000																																

The PCI Interrupt Acknowledge Register (**PIAC**) is a read only register that is used to initiate a single PCI Interrupt Acknowledge cycle. Any single byte or combination of bytes may be read from, and the actual byte enable pattern used during the read will be passed on to the PCI bus. Upon completion of the PCI interrupt acknowledge cycle, the Harrier presents

the resulting vector information obtained from the PCI bus as read data. The data from the PCI bus is swapped as described in the section titled *Endian Conversion* on page 2-45.

Outbound Translation Address (0, 1 and 2) Registers

Offset	OTAD0: *XCSR* + $220 OTAD1: *XCSR* + $228 OTAD2: *XCSR* + $230																																
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	OTADx																																Address Translation
	STA																END																
Operation	R/W																R/W																
Reset	$0000																$0000																

The Outbound Translation Address Registers (**OTAD0, OTAD1,** and **OTAD2**) contain address information associated with the mapping of PCI Memory or I/O space to PowerPC memory space. The fields within the **OTADx** registers are defined as follows:

STA: Start. This field determines the start address of a particular memory area on the PowerPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PowerPC address.

END: End. This field determines the end address of a particular memory area on the PowerPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PowerPC address.

Outbound Translation Offset/Translation Attribute (0, 1 and 2) Registers

Offset	OTOF0/OTAT0: *XCSR* + $224 OTOF1/OTAT1: *XCSR* + $22C OTOF2/OTAT2: *XCSR* + $234																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	OTOFx								OTATx																								Address Translation
																						RXT	RXS		ENA			WPE	SGE	RAE	MEM	IOM	
Operation	R/W								R	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	$0000								0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The Outbound Translation Offset Registers (**OTOF0, OTOF1,** and **OTOF2**) contain offset information associated with the mapping of PCI Memory or I/O space to PowerPC memory space. The **OTOFx** field represents a 16-bit offset that is added to the upper 16 bits of the PowerPC address to determine the PCI address used for outbound transfers. This offset allows PCI resources to reside at addresses that would not normally be visible from the PowerPC bus.

The Outbound Translation Attribute Registers (**OTAT0, OTAT1,** and **OTAT2**) contain attribute information associated with the mapping of PCI Memory or I/O space to PowerPC memory space. The fields within the **OTATx** registers are defined as follows:

RXT: Read Any Threshold. This field sets a threshold for when read-ahead prefetching will be resumed. If set, prefetching will resume once the FIFO is half empty. If cleared, prefetching will resume once the FIFO is completely empty. Please refer to the section titled *FIFO Tuning* on page 2-41 for more information.

RXS: Read Any Size. This field is used by the Harrier to determine a virtual FIFO size for outbound prefetch reads. The selection of a virtual FIFO size affects the number of the initial prefetch read cycles and the duration of the subsequent prefetch read cycles. Please refer to the section titled *FIFO Tuning* on page 2-41 for more information.

The encoding of this field is shown in the following table.

Table 3-20. OTATx RXS Encoding

RXS	Virtual FIFO Size	
	Bytes	Cache Lines
00	64	2
01	128	4
1x	256	8

ENA: Enable. If set, the corresponding Outbound Translation Function is enabled for read and write transactions.

WPE: Write Post Enable. If set, write posting is enabled for the corresponding Outbound Translation Function.

SGE: Store-Gather Enable. If set, the corresponding Outbound Translation Function will participate in Store-Gathering. If cleared, the corresponding Outbound Translation Function will not participate in Store-Gathering. This bit only has meaning if write posting is enabled.

RAE: Read-Ahead Enable. If set, the corresponding Outbound Translation Function will participate in read-ahead. If cleared, the corresponding Outbound Translation Function will not participate in read-ahead.

MEM: Memory/IO. If set, the corresponding Outbound Translation Function will generate transfers to or from PCI Memory space. If cleared, the corresponding Outbound Translation Function will generate transfers to or from PCI I/O space using the addressing mode defined by the IOM field.

IOM: I/O Mode. If set, the corresponding Outbound Translation Function will generate PCI I/O cycles using spread addressing as defined in the section titled "Generating PCI Cycles" in the previous chapter. If cleared, the corresponding Outbound Translation Function will generate PCI I/O cycles using contiguous addressing. This field only has meaning when the MEM bit is clear.

Outbound Translation Address (3) Register

Offset	*XCSR* + $238																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	OTAD3																																Address Translation
	STA																END															
Operation	R/W																R/W																
Reset	*XCSR* = $FEFF0000 => $8000 *XCSR* = $FEFF1000 => $8080 *XCSR* = $FEFF2000 => $8100 *XCSR* = $FEFF3000 => $8180																*XCSR* = $FEFF0000 => $807F *XCSR* = $FEFF1000 => $80FF *XCSR* = $FEFF2000 => $817F *XCSR* = $FEFF3000 => $81FF															

The Outbound Translation Address Register 3 (**OTAD3**) contains address information associated with the mapping of PCI I/O space to PowerPC memory space. **OTAD3** (in conjunction with **OTOF3** and **OTAT3**) is the only register set that can be used to initiate an access to the PCI **CONFIG_ADDRESS** ($80000CF8) and **CONFIG_DATA** ($80000CFC) registers. The power up value of **OTAD3**, **OTOF3** and **OTAT3** are set to allow access to these special register spaces without initializing the PowerPC Control Register Group. The fields within the **OTAD3** register are defined as follows:

STA: Start Address. This field determines the start address of a particular memory area on the PowerPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PowerPC address.

END: End Address. This field determines the end address of a particular memory area on the PowerPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PowerPC address.

Outbound Translation Offset/Translation Attribute (3) Registers

Offset	*XCSR* + $23C																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	OTOF3																OTAT3																Address Translation
																									ENA			WPE				IOM	
Operation	R/W																R	R	R	R	R	R	R	R	R/W	R	R	R/W	R	R	R	R/W	
Reset	*XCSR* = $FEFF0000 => $8000 *XCSR* = $FEFF1000 => $7F80 *XCSR* = $FEFF2000 => $7F00 *XCSR* = $FEFF3000 => $7E80																0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	

The Outbound Translation Offset Register 3 (**OTOF3**) contains offset information associated with the mapping of PCI I/O space to PowerPC memory space. The **OTOF3** field represents a 16-bit offset that is added to the upper 16 bits of the PowerPC address to determine the PCI address used for outbound transfers. This offset allows PCI resources to reside at addresses that would not normally be visible from the PowerPC bus.

The Outbound Translation Attribute Register 3 (**OTAT3**) contains attribute information associated with the mapping of PCI I/O space to PowerPC memory space. The fields within the **OTAT3** register are defined as follows:

ENA: Enable. If set, the corresponding Outbound Translation Function is enabled for read and write transactions.

WPE: Write Post Enable. If set, write posting is enabled for the corresponding Outbound Translation Function.

IOM: I/O Mode. If set, the corresponding Outbound Translation Function will generate PCI I/O cycles using spread addressing as defined in the section titled *Generating PCI Cycles* on page 2-36. If cleared, the corresponding Outbound Translation Function will generate PCI I/O cycles using contiguous addressing.

Passive Slave Address Registers

Offset	PSAD: *XCSR* + $248																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	PSAD																																Address Translation
	BASE																																
Operation	R/W																				R				R								
Reset	$00000																				$0				$00								

The Passive Slave Address Register **(PSAD)** control the mapping of the passive slave function within the PowerPC memory space:

BASE: Base Address. These bits define the memory space base address of the Passive slave translation function. Note that the actual number of writable bits positions depends on the size of the resource being offered. The size of a resource can be changed using the PSSZ register. The previous table in this chapter titled *PCI Message Passing (PMEP) Register Group* on page 3-23 shows the relationship between resource size and the BASE field.

The PSAD decoder is disabled when BASE is all zeros.

Table 3-21. BASE Encoding and Resource Size

Resource Size	PSSZ	Effects on BASE Field (*XCSR* Register Group) "W" => Writable Bit Position "0" => Fixed Zero Bit Position																			
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
4KB	$00	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
8KB	$01	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0
16KB	$02	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0
32KB	$03	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0
64KB	$04	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0
128KB	$05	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0
256KB	$06	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0
512KB	$07	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0
1MB	$08	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0
2MB	$09	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0

Table 3-21. BASE Encoding and Resource Size (Continued)

Resource Size	PSSZ	Effects on BASE Field (*XCSR* Register Group) "W" => Writable Bit Position "0" => Fixed Zero Bit Position																			
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
4MB	$0A	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0
8MB	$0B	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0
16MB	$0C	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0
32MB	$0D	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0
64MB	$0E	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0
128MB	$0F	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
256MB	$10	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
512MB	$11	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1GB	$12	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2GB	$13	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Passive Slave Offset/Translation Attribute Registers

Offset	PSOF/PSAT: *XCSR* + $24c																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	PSOF																PSSZ								PSAT								Address Translation
																									ENA				SGE				
Operation	R/W																R/W								R/W	R	R	R	R/W	R	R	R	
Reset	$0000																$00								0	0	0	0	0	0	0	0	

The Passive Slave Offset Registers (**PSOF**) contain offset information associated with the mapping of PowerPC Memory space to PCI memory space. The **PSOF** field represents a 16-bit offset that is added to the upper 16 bits of the PowerPC address to determine the PCI address used for outbound transfers. This offset allows PCI resources to reside at addresses that would not normally be visible from the PowerPC bus.

The Passive Slave Size Registers (**PSSZ**) establish the size of a resource offered by the Passive slave translation function. The value selected within this register determines the characteristics of the **PSAD** registers. Valid selections for a resource size are shown in the table below:

Table 3-22. PSSZ Encoding

PSSZ	Resource Size	PSSZx	Resource Size
$00	4KB	$0B	8MB
$01	8KB	$0C	16MB
$02	16KB	$0D	32MB
$03	32KB	$0E	64MB
$04	64KB	$0F	128MB
$05	128KB	$10	256MB
$06	256KB	$11	512MB
$07	512KB	$12	1 GB
$08	1MB	$13	2 GB
$09	2MB	$14 - $FF	Reserved
$0A	4MB		

The Passive Slave Attribute Registers (**PSAT**) contain attribute information associated with the mapping of PowerPC Memory space to PCI memory space. The fields within the **PSAT** registers are defined as follows:

ENA: Enable. If set, the corresponding Passive Slave Translation Function is enabled for read and write transactions.

SGE: Store Gathering Enable. If set, the corresponding Passive Slave Translation Function participates in Store-Gathering. If cleared the corresponding Passive Slave Translation Function does not participate in Store-Gathering.

XCFS Register Group

The following subsections describe the registers in the *XCFS* Register Group.

CONFIG_ADDRESS Register

The description of the **CONFIG_ADDRESS** register is presented in two perspectives: from the PCI bus (Little Endian bit ordering), and from the PowerPC bus (Big Endian bit ordering). Note that the view from the PCI bus is purely conceptual since there is no way to access the **CONFIG_ADDRESS** register from the PCI bus.

Conceptual perspective from the PCI bus:

Offset	$CFB								$CFA								$CF9								$CF8								Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	CONFIG_ADDRESS																																CONFIG ADDRESS
	EN								BUS								DEV				FUN				REG								
Operation	R/W	R	R	R	R	R	R	R	R/W								R/W				R/W				R/W						R	R	
Reset	1	0	0	0	0	0	0	0	$00								$00				$0				$00						0	0	

Perspective from the PowerPC bus:

Offset	$CF8								$CF9								$CFA								$CFB								Function
Bit (DH)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	CONFIG_ADDRESS																																CONFIG ADDRESS
	REG								DEV				FUN				BUS								EN								
Operation	R/W						R	R	R/W				R/W				R/W								R/W	R	R	R	R	R	R	R	
Reset	$00						0	0	$00				$0				$00								1	0	0	0	0	0	0	0	

The register fields are defined as follows:

REG: Register Number.

Configuration Cycles: Identifies a target word within a target's configuration space. This field is copied to the PCI AD bus during the address phase of a Configuration cycle.

Special Cycles: This field must be written with all zeros.

FUN: Function Number.

Configuration Cycles: Identifies a function number within a target's configuration space. This field is copied to the PCI AD bus during the address phase of a Configuration cycle.

Special Cycles: This field must be written with all ones.

DEV: Device Number.

Configuration Cycles: Identifies a target's physical PCI device number. Refer to the section titled *Generating PCI Cycles* on page 2-36 for a description of how this field is encoded.

Special Cycles: This field must be written with all ones.

BUS: Bus Number.

Configuration Cycles: Identifies a targeted bus number. If written with all zeros, a Type 0 Configuration Cycle will be generated. If written with any value other than all zeros, then a Type 1 Configuration Cycle will be generated.

Special Cycles: Identifies a targeted bus number. If written with all zeros, a Special Cycle will be generated. If written with any value other than all zeros, then a Special Cycle translated into a Type 1 Configuration Cycle will be generated.

EN: Enable.

Configuration Cycles: Writing a one to this bit enables **CONFIG_DATA** to Configuration Cycle translation. If this bit is a zero, subsequent accesses to **CONFIG_DATA** will be passed though as I/O Cycles.

Special Cycles: Writing a one to this bit enables **CONFIG_DATA** to Special Cycle translation. If this bit is a zero, subsequent accesses to **CONFIG_DATA** will be passed though as I/O Cycles.

CONFIG_DATA Register

The description of the **CONFIG_DATA** register is also presented in two perspectives: from the PCI bus (Little Endian it ordering), and from the PowerPC bus (Big Endian bit ordering). Note that the view from the PCI bus is purely conceptual since there is no way to access the **CONFIG_DATA** register from the PCI bus.

Conceptual perspective from the PCI bus:

Offset	**$CFF**								**$CFE**								**$CFD**								**$CFC**								**Function**
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	CONFIG_DATA																																**CONFIG DATA**
	Data ‘D’								Data ‘C’								Data ‘B’								Data ‘A’								
Operation	R/W								R/W								R/W								R/W								
Reset	n/a								n/a								n/a								n/a								

Perspective from the PowerPC bus:

Offset	**$CFC**								**$CFD**								**$CFE**								**$CFF**								**Function**
Bit (DL)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	CONFIG_DATA																																**CONFIG DATA**
	Data ‘A’								Data ‘B’								Data ‘C’								Data ‘D’								
Operation	R/W								R/W								R/W								R/W								
Reset	n/a								n/a								n/a								n/a								

PCFS Register Group

This register group represents the Harrier's PCI Configuration Space. A reflection of this Configuration Space is represented within the *XCSR* Register Group. Note that in many cases a register represented within the *PCFS* Register Group will have different read/write characteristics than the same register represented within the *XCSR* Register Group. In general, the read/write characteristics of the registers within the *XCSR* Register Group are quite liberal while those of the *PCFS* Register Group are strictly limited to the abilities defined by the PCI Local Bus Specification.

The reflected configuration space in the *XCSR* Register Group is byte-swapped as described in the section titled "Endian Conversation" in the previous chapter. Except where noted, all bit definitions are discussed with respect to the register within the *PCFS* Register Group.

Vendor ID/Device ID Registers

Perspective from the PCI Bus:

Offset	***PCFS* + $00**		**Function**
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Name	DEVI	VENI	**Header**
Operation	R	R	
Reset	$480B	$1057	

Perspective from the PowerPC Bus:

Offset	***XCSR* + $300**		**Function**
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
Name	VENI	DEVI	**Reflected PCI Configuration Space**
Operation	R	R	
Reset	$5710	$0B48	

The Vendor ID Register (**VENI**) is a read-only register that identifies the manufacturer of the device. This identifier is allocated by the PCI SIG to ensure uniqueness. $1057 has been assigned to Motorola and is hardwired as a read-only value.

The Device ID Register (**DEVI**) is a read-only register that uniquely identifies this particular device. The Harrier always returns $480B.

Command/Status Registers

Perspective from the PCI Bus:

Offset	***PCFS* + $04**																												Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	STAT																CMMD																Header
	RCVPE	SIGSE	RCVMA	RCVTA	SIGTA	SELTIM1	SELTIM0	DPAR	FAST		P66M													SERR		PERR				MSTR	MEMSP	IOSP	
Operation	R/C	R/C	R/C	R/C	R/C	R	R	R/C	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R	R/W	R	R	R	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

Perspective from the PowerPC Bus:

Offset	***XCSR* + $304**																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	CMMD																STAT																Reflected PCI Configur-ation Space
		PERR				MSTR	MEMSP	IOSP								SERR	FAST		P66M						RCVPE	SIGSE	RCVMA	RCVTA	SIGTA	SELTIM1	SELTIM0	DPAR	
Operation	R	R/W	R	R	R	R/W	R/W	R/W	R	R	R	R	R	R	R	R/W	R	R	R	R	R	R	R	R	R/C	R/C	R/C	R/C	R/C	R	R	R/C	
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	

The Command Register (**CMMD**) provides course control over the Harrier's ability to generate and respond to PCI cycles. The fields within the **CMMD** register are defined as follows:

IOSP: I/O Space Enable. If set, the Harrier responds to PCI I/O space accesses when appropriate. If cleared, the Harrier does not respond to PCI I/O space accesses.

MEMSP: Memory Space Enable. If set, the Harrier responds to PCI memory space accesses when appropriate. If cleared, the Harrier does not respond to PCI memory space accesses.

MSTR: Bus Master Enable. If set, the Harrier may act as a master on PCI. If cleared, the Harrier may not act as a master.

PERR: Parity Error Response. This bit enables the PERR_ output pin. If cleared, the Harrier will never drive PERR_. If set, the Harrier will drive PERR_ active when a data parity error is detected.

SERR: System Error Enable. This bit enables the SERR_ output pin. If cleared, the Harrier never drives SERR_. If set, the Harrier drives SERR_ active when a system error is detected.

The Status Register (**STAT**) is used to record information for PCI bus related events. The fields within the **STAT** register are defined as follows:

P66M: PCI 66 MHz. This bit indicates the Harrier is capable of supporting a 66.67 MHz PCI bus.

FAST: Fast Back-to-Back Capable. This bit indicates that the Harrier is capable of accepting fast back-to-back transactions with different targets.

DPAR: Data Parity Detected. This bit is set when three conditions are met: 1) the Harrier asserted PERR_ itself or observed PERR_ asserted; 2) Harrier was the PCI master for the transfer in which the error occurred; 3) the PERR bit in the **CMMD** register is set. This bit is cleared by writing it to 1; writing a 0 has no effect.

SELTIM: DEVSEL Timing. This field indicates that the Harrier always asserts DEVSEL_ as a 'medium' responder.

SIGTA: Signalled Target Abort. This bit is set by the Harrier whenever it terminates a slave transaction with a target-abort. It is cleared by writing it to 1; writing a 0 has no effect.

RCVTA: Received Target Abort. This bit is set by the Harrier whenever it detects a master transaction is terminated by a target-abort. It is cleared by writing it to 1; writing a 0 has no effect.

RCVMA: Received Master Abort. This bit is set by the Harrier whenever it detects a master transaction (except for Special Cycles) is terminated by a master-abort. It is cleared by writing it to 1; writing a 0 has no effect.

SIGSE: Signaled System Error. This bit is set whenever the Harrier asserts SERR_. It is cleared by writing it to 1; writing a 0 has no effect.

RCVPE: Detected Parity Error. This bit is set whenever the Harrier detects a parity error, even if parity error response is disabled (see bit PERR in the **CMMD** register). It is cleared by writing it to 1; writing a 0 has no effect.

Revision ID/Class Code Registers

Perspective from the PCI Bus:

Offset	***PCFS* + $08**																												**Function**
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	CLAS																								REVI								**Header**
Operation	R																								R								
Reset	$060000 or $0E0001																								$02								

Perspective from the PowerPC Bus:

Offset	***XCSR* + $308**																												**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	REVI								CLAS																								**Reflected PCI Configur-ation Space**
Operation	R								R/W																								
Reset	$02								$000006 or $01000E																								

The Revision ID Register (**REVI**) is a read-only register that identifies the Harrier revision level.

The Class Code Register (**CLAS**) is a read-only register from within the *PCFS* Register Group, and may be written at any time from within the *XCSR* Register Group. The **CLAS** register identifies the class code of the Harrier. The initial contents of this register is established with external resistors placed on signals sampled at the release of reset. Please refer to section titled "Hardware Configuration" in the previous chapter for more information.

The initial class code of the Harrier depends on the level of participation in the I_2O Message Passing protocol. The table below shows the dependency between class codes and I_2O participation.

Table 3-23. CLAS Encoding

I_2O Participation	**CLAS (*PCFS* Register Group)**		
	Base	**Sub**	**Program**
None	$06 Bridge Device	$00 Host Bridge	$00 Not Used
I_2O Host Bridge	$06 Bridge Device	$00 Host Bridge	$00 Not Used
I_2O IOP Agent	$0E I_2O Controller	$00 Not Used	$01 Includes Interrupt Capability

Cache Line Size/Master Latency Timer/Header Type Register

Perspective from the PCI Bus:

Offset	***PCFS* + $0C**																									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	**Function**
Name									HEAD								MLAT								CLSZ								**Header**
																	LAT												CLS				
Operation	R								R								R/W					R	R	R	R	R	R	R	R				R/W
Reset	$00								$00								$00					0	0	0	0	0	0	0	0				$00

Perspective from the PowerPC Bus:

Offset	*XCSR* + $30C																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	CLSZ								MLAT								HEAD																Reflected PCI Configur-ation Space
					CLS				LAT																								
Operation	R	R	R	R	R/W				R/W					R	R	R	R								R								
Reset	0	0	0	0	$00				$00					0	0	0	$00								$00								

The Header Type Register (**HEAD**) is a read-only register that identifies the Harrier as the following:

Header Type: $00 Single Function Configuration Header

The Master Latency Timer Register (**MLAT**) represents the value used for the Master Latency Timer. The Master Latency Timer specifies the amount of PCI clock periods that the Harrier may remain on the PCI bus during burst cycles after GNT_ is taken away. The **MLAT** register provides a minimum granularity of the 8 PCI clock periods.

The PCI specification states that this register must power up to all zeros. Severe performance degradation may result if this register is not adjusted from the reset value.

The Cache Line Size Register (**CLSZ**) represents the number of 32-bit words that define a Harrier cache-line. A Harrier cache line is defined as 32-bytes, which is eight 32-bit words. If a value of $08 is written to this register, the value will be retained. If any other value is written to this register, a value of $00 will be retained.

The PCI specification states that this register must power up to all zeros. The Harrier is not able to generate the Memory Write and Invalidate command, therefore this register is only used to inform other PCI masters of the Harrier supported cache-line size for Read, Read Line, and Read Multiple commands.

Message Passing Register Group Base Address Register

Perspective from the PCI Bus:

Offset	*PCFS* + $10																										Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	MPBAR																																Header
	BASE																								PRE	MTYP1	MTYP0	IO/MEM					
Operation	R/W																R								R	R	R	R					
Reset	$00000																$0								0	0	0	0					

Perspective from the PowerPC Bus:

Offset	*XCSR* + $310																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MPBAR																																Reflected PCI Configur-ation Space
					PRE	MTYP1	MTYP0	IO/MEM	BASE								BASE																
Operation	R				R	R	R	R	R/W								R								R/W								
Reset	$0				0	0	0	0	$0								$0								$00000								

The Message Passing Register Group Base Address Register (**MPBAR**) controls the mapping of the *PMEP* Register Group within PCI Memory space.

IO/MEM: **I/O Space Indicator.** This bit is hard-wired to a logic zero to indicate PCI memory space.

MTYPx: Memory Type. These bits are hard-wired to zero to indicate that the *PMEP* Register Group can be located anywhere in the 32-bit address space

PRE: Prefetch. This bit is hard-wired to zero to indicate that the *PMEP* Register Group is not prefetchable.

BASE: **Base Address**. These bits define the memory space base address of *PMEP* Register Group. The group may be placed at any 4K Byte boundary. This field is only accessible when the ENA bit field is set in the **MPAT** register.

Note that the BASE field is byte swapped when accessing from the *XCSR* Register Group. For example, programming a base address of $12345 within the *PCFS* Register Group would be the same as programming $53412 within the *XCSR* Register Group.

Inbound Translation Base Address (0, 1, 2 and 3) Registers

Perspective from the PCI Bus:

Offset	**ITBAR0: *PCFS* + $14** **ITBAR1: *PCFS* + $18** **ITBAR2: *PCFS* + $1C** **ITBAR3: *PCFS* + $20**																																**Function**
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	ITBARx																																**Header**
	BASE																												PRE	MTYP1	MTYP0	IO/MEM	
Operation	R/W																R								R				R	R	R	R	
Reset	$00000																$0								$0				1	0	0	0	

Perspective from the PowerPC Bus:

Offset	ITBAR0: *XCSR* + $314 ITBAR1: *XCSR* + $318 ITBAR2: *XCSR* + $31C ITBAR3: *XCSR* + $320																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	ITBAR																																Reflected PCI Configur-ation Space
					PRE	MTYP1	MTYP0	IO/MEM	BASE								BASE																
Operation	R				R	R	R	R	R/W								R								R/W								
Reset	$0				1	0	0	0	$0								$0								$0000								

The Inbound Translation Base Address Registers (**ITBAR0, ITBAR1, ITBAR2** and **ITBAR3**) control the mapping of the Inbound Translation function within PCI Memory space.

IO/ME: I/O Space Indicator. This is a read-only inverted copy of the MEM bit defined in the **ITATx** register. The IO/MEM field reflects the ability of the Inbound Translation Function to support either memory space or I/O space accesses. This field is accessible only when the ENA bit is set in the **ITATx** register. Upon reset, this bit is set to a zero indicating this resource is a memory space resource.

MTYP: Memory Type. These bits are hard-wired to zero to indicate that the Inbound Translation Function can be located anywhere in the 32-bit address space

PRE: Prefetch. This is a read-only copy of the PRE bit defined in the **ITATx** register. The PRE field reflects the ability of the Inbound Translation Function to support prefetching. This field is accessible only when the ENA bit is set in the **ITATx** register. Upon reset, this bit is set to a one indicating this resource is prefetchable.

BASE: **Base Address**. These bits define the memory space base address of the Inbound Translation Function. Note that the actual number of writable bit positions depends on the size of the resource

being offered. The size of a resource can be changed using the **ITSZx** register. Table 3-24 shows the relationship between resource size and the BASE field.

This field is accessible only when the ENA bit is set in the **ITATx** register.

The relationship of a writable and non-writable bit is maintained regardless of whether an access is from within the *PCFS* or *XCSR* Register Groups.

Note that the BASE field is byte swapped when accessing from the *XCSR* Register Group. For example, programming a base address of $12345 within the *PCFS* Register Group is the same as programming $53412 within the *XCSR* Register Group.

Table 3-24. BASE Encoding and Resource Size

Resource Size	ITSZx	Effects on BASE Field (*PCFS* Register Group) "W" => Writable Bit Position "0" => Fixed Zero Bit Position																			
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
4KB	$00	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
8KB	$01	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0
16KB	$02	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0
32KB	$03	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0
64KB	$04	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0
128KB	$05	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0
256KB	$06	W	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0
512KB	$07	W	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0
1MB	$08	W	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0
2MB	$09	W	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0
4MB	$0A	W	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0
8MB	$0B	W	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0
16MB	$0C	W	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0
32MB	$0D	W	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0
64MB	$0E	W	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0
128MB	$0F	W	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
256MB	$10	W	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-24. BASE Encoding and Resource Size (Continued)

Resource Size	ITSZx	Effects on BASE Field (*PCFS* Register Group) "W" => Writable Bit Position "0" => Fixed Zero Bit Position																			
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
512MB	$11	W	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1GB	$12	W	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2GB	$13	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subsystem Vendor ID/Subsystem ID Registers

Perspective from the PCI Bus:

Offset	*PCFS* + $2C																																Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	SUBI																SUBV																Header
Operation	R																R																
Reset	$1057																$0000																

Perspective from the PowerPC Bus:

Offset	*XCSR* + $32C																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SUBV																SUBI																Reflected PCI Configur-ation Space
Operation	R/W																R/W																
Reset	$0000																$5710																

The Subsystem Vendor ID Register (**SUBV**) is a read-only register from within the *PCFS* Register Group, and may be written at any time from within the *XCSR* Register Group. The **SUBV** register provides a second level of identification for the manufacturer of this particular device. This identifier is allocated by the PCI SIG to ensure uniqueness. This register is configured to the Motorola value of $1057 upon release of reset.

The Subsystem ID Register (**SUBI**) is a read-only register from within the *PCFS* Register Group, and may be written at any time from within the *XCSR* Register Group. The **SUBI** register provides a second level of identification for this particular device. This register will be configured to $0000 upon the release of reset.

Interrupt Line/Interrupt Pin/Minimum Grant/Maximum Latency Registers

Perspective from the PCI Bus:

Offset	*PCFS* + $3C				Function
Bit	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
Name	MXLA	MNGN	INTP	INTL	Header
Operation	R	R	R	R/W	
Reset	$00	$00	$01	$00	

Perspective from the PowerPC Bus:

Offset	*XCSR* + $33C				Function
Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	
Name	INTL	INTP	MNGN	MXLA	Reflected PCI Configur-ation Space
		INT			
Operation	R/W	R R R R R R/W R/W R/W	R/W	R/W	
Reset	$00	0 0 0 0 0 0 0 1	$00	$00	

The Interrupt Line Register (**INTL**) contains interrupt routing information. The Harrier does not have any hardware associated with this register, and is not affected in any way by the contents of this register. Initialization software may write interrupt routing information into this register during system configuration.

The Interrupt Pin Register (**INTP**) contains information pertaining to the PCI interrupt pin the Harrier is driving. This register is read-only from the *PCFS* Register Group, and may be written at any time from within the *XCSR* Register Group. The Harrier is a single function device and therefore is limited by the PCI Local Bus Specification to only driving

INTA. In special cases, the Harrier may be programmed to drive any one of the four PCI interrupts. This register may be modified to show which of the four interrupt lines the Harrier is driving. The recommended encoding of this field is show in the table below:

Table 3-25. INTP INT Encoding

INT	PCI Interrupt
000	Undefined
001	INTA
010	INTB
011	INTC
100	INTD
101 - 111	Undefined

Note that the selection of a particular INTx line is handled by the ***XCSR*.BPCS** register. The **INTP** register is for reference only and does not control any hardware

The Minimum Grant Register (**MNGN**) is a read-only register from within the *PCFS* Register Group, and may be written at any time from within the *XCSR* Register Group. The **MNGN** register specifies how long of a burst period the Harrier requires. The value is presented is in units of 0.25 us. This register will be configured to $00 following the release of reset which indicates the Harrier has no particular grant requirements.

The Maximum Latency Register (**MXLA**) is a read-only register from within the *PCFS* Register Group, and may be written at any time from within the *XCSR* Register Group. The **MXLA** register specifies how often the Harrier needs to gain access to the PCI bus. The value is presented is in units of 0.25 us. This register will be configured to $00 following the release of reset which indicates the Harrier has no particular latency requirements.

Message Passing Attribute Register

Perspective from the PCI Bus:

Offset	*PCFS* + $44																												Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name									MPAT																								Address Translation
																GBL									ENA	MEM	WPE	RAE					
Operation	R								R	R	R	R	R	R	R	R/W	R	R	R	R	R	R	R	R	R/W	R	R/W	R/W	R	R	R	R	
Reset	$00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	

Perspective from the PowerPC Bus:

Offset	*XCSR* + $344																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MPAT																												Reflected PCI Configur-ation Space
	ENA	MEM	WPE	RAE																				GBL									
Operation	R/W	R	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R								
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$00								

RAE: Read-Ahead Enable. The *PMEP* Register Group does not support read-ahead, however there is a corner case where this field affects the actions of the PPC Master during a cache-line aligned (referenced to the physical address in memory of the queue element) PCI burst read from the MIIQ/MIOQ registers. If this field is set, the PPC Master performs a cache-line read. If this field is cleared, the PPC Master performs a single beat read.

This field has little (if any) practical value, since a majority of the **MIIQ**/**MIOQ** reads from PPC memory are translated into single beat reads.

WPE: Write-Post Enable. If set, write-posting is enabled on the PCI bus. The Harrier will acknowledge a data transfer, collect data from the PCI bus and forward that data on to local memory. If cleared, each write transaction is fully compelled. The Harrier will write each PCI 32-bit or 64-bit beat as a single beat transaction on the PowerPC bus. After each transaction a single acknowledge will be given to the PCI bus.

MEM: Memory. This read-only field is hardwired to a one to indicate the *PMEP* Register Group may only be located within PCI memory space.

ENA: Enable. If set, the Message Passing Function is enabled for read and write transactions. Writes to the ***PCFS*.MPBAR** register will be accepted and reads will return valid data. If cleared, the Message Passing Function is not enabled. Writes to the ***PCFS*.MPBAR** register will be discarded and reads will return all zeros. The visibility of the ***XCSR*.MPBAR** register is unaffected by the ENA field.

GBL: Global. If set, the GBL_ pin will be asserted for each PowerPC transaction indicating the transaction may be snooped by the processor. If cleared, the GBL_ pin is not asserted and the transaction is not snooped.

Inbound Translation Size/Offset (0, 1, 2 and 3) Registers

Perspective from the PCI Bus:

Offset	**ITSZ0/ITOF0: *PCFS* + $48** **ITSZ1/ITOF1: *PCFS* + $50** **ITSZ2/ITOF2: *PCFS* + $58** **ITSZ3/ITOF3: *PCFS* + $60**																										**Function**
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	ITOFx																								ITSZx								**Address Translation**
Operation	R/W																R								R/W								
Reset	$0000																$00								$00								

Perspective from the PowerPC Bus:

Offset	ITSZ0/ITOF0: *XCSR* + $348 ITSZ1/ITOF1: *XCSR* + $350 ITSZ2/ITOF2: *XCSR* + $358 ITSZ3/ITOF3: *XCSR* + $360																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	ITSZx																ITOFx																Reflected PCI Configur-ation Space
Operation	R/W								R								R/W																
Reset	$00								$00								$0000																

The Inbound Translation Size Registers (**ITSZ0, ITSZ1, ITSZ2** and **ITSZ3**) establish the size of a resource offered by an Inbound Translation Function. The value selected within this register determines the characteristics of the **ITBARx** registers. Valid selections for a resource size are shown in the table below:

Table 3-26. ITSZx Encoding

ITSZx	Resource Size	ITSZx	Resource Size
$00	4KB	$0B	8MB
$01	8KB	$0C	16MB
$02	16KB	$0D	32MB
$03	32KB	$0E	64MB
$04	64KB	$0F	128MB
$05	128KB	$10	256MB
$06	256KB	$11	512MB
$07	512KB	$12	1GB
$08	1MB	$13	2GB
$09	2MB	$14 - $FF	Reserved
$0A	4MB		

The Inbound Translation Offset Registers (**ITOF0, ITOF1, ITOF2,** and **ITOF3**) contain offset information associated with the mapping of PowerPC address space to PCI memory space. The **ITOFx** field represents

a 16-bit offset that is added to the upper 16 bits of the PCI address to determine the PowerPC address used for inbound transfers. This offset allows PowerPC resources to reside at addresses that would not normally be visible from the PCI bus.

Note that the **ITOFx** register is byte swapped when accessing from the *XCSR* Register Group. For example, programming an offset of $1234 within the *PCFS* Register Group would be the same as programming $3412 within the *XCSR* Register Group.

Inbound Translation Attribute (0, 1, 2 and 3) Registers

Perspective from the PCI Bus:

Offset	ITAT0: *PCFS* + $4C ITAT1: *PCFS* + $54 ITAT2: *PCFS* + $5C ITAT3: *PCFS* + $64																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Function
Name									ITATx																								Address Transla--tion
													AWL	CRI	CWF	GBL		RMT	RMS			RXT	RXS		ENA	MEM	WPE	RAE	PRE				
Operation	R								R	R	R	R	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	
Reset	$00								0	0	0	0	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	0	1	0	0	0	

Perspective from the PowerPC Bus:

Offset	ITAT0: *XCSR* + $34C ITAT1: *XCSR* + $354 ITAT2: *XCSR* + $35C ITAT3: *XCSR* + $364																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	ITATx																																Reflected PCI Configur-ation Space
	ENA	MEM	WPE	RAE	PRE					RMT	RMS			RXT	RXS						AWL	CRI	CWF	GBL								
Operation	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R							
Reset	0	1	0	0	1	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1	0	0	0	$00							

The Inbound Translation Attribute Registers (**ITAT0, ITAT1, ITAT2** and **ITAT3**) contain attribute information associated with the mapping of PowerPC address space to PCI memory space. The fields within the **ITATx** registers are defined as follows:

PRE: Prefetch Enable. This field represents a control point for the **ITBARx**.PRE field, enabling software to establish the prefetchable status of an Inbound Translation Function. This field does not affect any hardware. If set, the **ITBARx**.PRE field will be set to a logic 1. If cleared, the **ITBARx**.PRE field will be set to a logic 0.

RAE: Read-Ahead Enable. If set, read-ahead is enabled on the PowerPC bus. This forces the Harrier to perform a predefined number of cache line reads during the first fetch. The number of lines initially read and the subsequent resume characteristics are controlled by the setting of the RMT/RMS and RXT/RXS fields. If cleared, the Harrier performs one single beat read and does not start a new read until a previously read beat has been transferred across PCI. Each subsequent read is always be a single beat read.

WPE: Write-Post Enable. If set, write-posting is enabled on the PCI bus. The Harrier continues to collect data from the PCI bus. Once a certain threshold is met, the Harrier presents the collected data as

cache-line writes to the PowerPC bus. If cleared, each write transaction is fully compelled. The Harrier writes each PCI 32-bit or 64-bit beat as a single beat transaction on the PowerPC bus.

MEM: Memory. If set, the Harrier maps the corresponding Inbound Translation Function to PCI memory space. If cleared, the resource is mapped to PCI I/O space. The state of this field is reflected within the **ITBARx**.IOMEM field. If the ENA field is set, then the IOMEM field is the inversion of the MEM field.

ENA: Enable. If set, the corresponding Inbound Translation Function is enabled for read and write transactions.

RXS: Read Any Size. This field is used by the Harrier to determine a virtual FIFO size for inbound prefetch reads. This field is applicable only to PCI Read or Read Line command types. The selection of a virtual FIFO size will affect the number of the initial prefetch read cycles and the duration of subsequent prefetch read cycles. Please refer to the section titled "FIFO Tuning" in the previous chapter for more information.

The encoding of this field is shown in the following table.

Table 3-27. ITATx RXS/RMS Encoding

RXS/RMS	Virtual FIFO Size	
	Bytes	Cache Lines
00	64	2
01	128	4
1x	256	8

RXT: Read Any Threshold. This field sets a threshold for when read-ahead prefetching will be resumed. This field is applicable only to PCI Read or Read Line transaction types. If set, prefetching will resume once the FIFO is half empty. If cleared, a prefetching will resume once the FIFO is completely empty. Please refer to the section titled *FIFO Tuning* on page 2-41 for more information.

RMS: Read Multiple Size. Functionally this field is the same as the RXS field except that this is the virtual fifo size that is applied during PCI Read Multiple command types. Please refer to the section titled *FIFO Tuning* on page 2-41 for more information.

RMT: Read Multiple Threshold. Functionally this field is the same as the RXT field except that this is the threshold that is applied during PCI Read Multiple command types. Please refer to the section titled *FIFO Tuning* on page 2-41 for more information.

GBL: Global. If set, the GBL_ pin will be asserted for each PowerPC transaction indicating the transaction may be snooped by the processor. If cleared, the GBL_ pin is not asserted and the transaction is not snooped.

CWF: Cache-line Write Flush. When set forces the use of Write-with-Flush transfer types during cache-line writes. Forces the processor to perform copyback writes during snoop hits. When cleared forces Write-with-kill transfer type during cache-line writes. Forces the processor to simply invalidate cache-lines during snoop hits. Used as a work-around for processors that can not handle Write-with-kill correctly. Please refer to the following table for the transfer codes associated with this bit.

CRI: Cache-line Read Invalidate. When set forces the use of Read-with-intent-to-modify commands. This forces processor cached data in the E and S states to be invalidated during snoop hits. When cleared uses non RWITM commands which allows the processor to retain cached data in the E and S states. Please refer to the following table for the transfer codes associated with this bit.

AWL: Atomic With Lock. When set will force the use of atomic transfer types whenever possible during PCI lock cycles. When cleared non-atomic transfer types are used. Please refer to the following table for the transfer codes associated with this bit.

Table 3-28. Harrier Generated Transfer Types

Origin	CRI	CWF	AWL	Transfer Size	TT[0:4]	Transfer Type
PCI Read or DMA Read	0	x	x	Single Beat or Burst	01010	Read
	1	x	x	Single Beat or Burst	01110	Read-with-intent-to-modify
PCI Read Lock	0	x	0	Single Beat or Burst	01010	Read
	0	x	1	Single Beat or Burst	11010	Read-atomic
	1	x	0	Single Beat or Burst	01110	Read-with-intent-to-modify
	1	x	1	Single Beat or Burst	11110	Read-with-intent-to-modify-atomic
PCI Write or DMA Write	x	x	x	Single Beat	00010	Write-with-flush
	x	0	x	Burst	00110	Write-with-kill
	x	1	x	Burst	00010	Write-with-flush
PCI Write Lock	x	x	0	Single Beat	00010	Write-with-flush
	x	0	0	Burst	00110	Write-with-kill
	x	1	0	Burst	00010	Write-with-flush
	x	x	1	Single Beat	10010	Write-with-flush-atomic
	x	0	1	Burst	00110	Write-with-kill
	x	1	1	Burst	10010	Write-with-flush-atomic

PCI Status Register

Perspective from the PCI Bus:

Offset	*PCFS* + $80																										Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	PSTA																																Status
	LBA																																
Operation	R	R	R	R	R	R	R	R	R								R								R								
Reset	n/a	0	0	0	0	0	0	0	$00								$00								$00								

Perspective from the PowerPC Bus:

Offset	*XCSR* + $380																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name																									PSTA								Reflected PCI Configur-ation Space
																									LBA								
Operation	R								R								R							R	R	R	R	R	R	R	R	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The PCI Status Register contains the Loop Back Access bit. The LBA bit allows software to determine if the configuration space access is looping back to the PCI configuration space in the Harrier. In other words, if the Harrier is reading its own configuration space. The LBA bit is defined as follows:

LBA: Loop Back Access. The LBA bit is read as a one when read by the PCI bus master in the Harrier. The LBA bit is read as a zero when read by another PCI bus master. The LBA bit always reads zero from the PowerPC bus.

PCI General Purpose Register

Perspective from the PCI Bus:

Offset	*PCFS* + $84																												Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	PCIGP																												General Purpose Register
Operation	R/W																												
Reset	$00000000																												

Perspective from the PowerPC Bus:

Offset	*XCSR* + $384																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	PCIGP																																Reflected PCI Configur-ation Space
Operation	R/W																																
Reset	$00000000																																

The PCI General Purpose Register is provided for inter-process message passing or general purpose storage. It is accessible from the PCI bus and the PowerPC bus. It does not control any hardware.

Note that the PCIGP register is byte swapped when accessing from the *XCSR* Register Group. For example, programming a value of $01234567 within the *PCFS* Register Group would be the same as programming $67452301 within the *XCSR* Register Group.

DMA Controller

All of the registers for this function are located in the PowerPC address space as a part of the *XCSR* Register Group.

DMA Control Register

Offset	*XCSR* + $250																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	DCTL																																DMA
					ABT	PAU	DGO		MOD		XTH			PBT							CSE	CRI		GBL									
Operation	R	R	R	R	R/S	R/S	R/S	R	R/W	R	R/W	R/W	R	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R	R/W	R	R	R	R	R	R	R	R	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The DMA Control Register (**DCTL**) provides the control fields for the Harrier DMA function. The fields within the **DCTL** register are defined as follows:

ABT: Abort. Writing a one to this field will abort a DMA transaction. An abort is considered an unrecoverable operation to a DMA transaction, meaning that an aborted transaction may not be restarted. When issuing an abort, both the PowerPC and/or PCI masters are immediately stopped and all FIFO contents are invalidated. If the abort took affect before the completion of a transaction, then the **DSTA**.ABT field will be set once the DMA Controller reaches the aborted state. Reading this field will always return a zero.

PAU: Pause. Writing a one to this field will pause a DMA transaction. This bit is only applicable to Linked-List-Mode transactions. When pausing a DMA transaction, the DMA controller will stop at the completion of the current linked-list transfer. If the pause took affect before the completion of a transaction, then the **DTSA**.PAU field will be set once the DMA Controller reaches the paused state. A paused transaction may be restarted by writing a one to the DGO field. Reading this field will always return a zero.

Abort has overriding authority over pause. If a commanded pause is followed by an commanded abort, then the DMA controller will honor the commanded abort.

DGO: DMA Go. Writing a one to this field will start a DMA transaction. Setting the DGO field will cause the **DSTA**.BSY field to be set and will clear all **DSTA** completion status bits (i.e. SMA, RTA, etc.). Reading this field will always return a zero.

MOD: Mode. This field establishes the type of DMA transaction to be performed. If set, a Direct-Mode transaction will be performed. A Direct-Mode transaction performs one transfer according to the contents of the **DSAD**, **DSAT**, **DDAD**, **DDAT**, and **DCNT** registers. If cleared, a Linked-List-Mode transaction will be performed. A Linked-List-Mode transaction will perform multiple transfers that are driven by a list of descriptors stored in PowerPC memory. A Linked-List-Mode transaction will obtain the first descriptor from the starting address placed within the **DNLA** register.

XTH: PowerPC Throttle. This field is used to control the attempted transfer size on the PowerPC bus. The Harrier associates a transfer size with the assertion of the PowerPC bus request. The Harrier will remove the bus request only after completing the desired transfer size. If there are more PowerPC transactions pending, the Harrier will re-assert the bus request after two clocks. The encoding of this field is shown in the table below.

Table 3-29. DCTL XTH Encoding

XTH	Transfer Size	
	Bytes	Cache Lines
00	256	8
01	512	16
10	1024	32
11	Continuous	Continuous

PBT: PCI Back-off Timer. This field establishes a maximum PCI bus bandwidth that the DMA function may use when performing transfers to or from PCI space. The Harrier will attempt to complete an entire

transfer in one burst unless interrupted by the PCI Master Latency Timer. Once the Latency Timer expires and the current burst completes, the PCI Back-off Timer will start to count. The Harrier will not attempt to restart the transfer until after the PCI Back-off Timer has expired. The back off time is specified in units of PCI clock periods. The table below shows the encoding for this field.

Table 3-30. DCTL PBT Encoding

PBT	Back-off Value		
	PCI Clock Counts	Approx Time @ 33 MHz	Approx Time @ 66 MHz
000	0	0.000 µs	0.000 µs
001	32	0.960 µs	0.480 µs
010	64	1.920 µs	0.960 µs
011	128	3.840 µs	1.920 µs
100	256	7.680 µs	3.840 µs
101	512	15.360 µs	7.680 µs
110	1024	30.720 µs	15.360 µs
111	2048	61.440 µs	30.720 µs

CSE: Copy-back Snarfing Enable. This field is used to enable snarfing on a processor's copy-back cycle. Refer to the section titled *Copyback Snarfing* on page 2-35 for more information. If set, copy-back snarfing is attempted for PowerPC DMA cycles. If cleared, snarfing will not be attempted.

CRI: Cache-line Read Invalidate. If set, the DMA will use the "Read-with-intent-to-modify" transfer type during PowerPC descriptor fetch read cycles. This will force processor cached data in the E and S states to be invalidated during snoop hits. If cleared, the DMA will use the "Read" transfer type. This will allow the processor to retain cached data in the E and S states.

GBL: Global. If set, the DMA will assert the GBL_ pin during PowerPC descriptor fetch read cycles. This allows the processor to snoop the DMA transfer. If cleared, the GBL_ pin is not asserted and the processor will not be able to snoop the DMA transaction.

DMA Status Register

Offset	*XCSR* + $254																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DSTA																																DMA
	SMA	RTA	MRC	XBT	ABT	PAU	DON	BSY																									
Operation	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R								R								
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	$00								$00								

The DMA Status Register (**DSTA**) provides the status fields for the Harrier DMA function. The BSY field represents the current state of the DMA controller, and the remaining fields indicate completion status. Note that there will always be only one field set within the **DSTA** register at all times. When the DMA controller is starting a transaction (i.e. the **DCTL**.DGO field is set) the BSY field will be asserted and all of the completion status fields will be cleared. The BSY field will remain asserted and the completion status fields will remain cleared throughout the entire DMA transaction. Once the DMA Controller is finished, then the BSY field will be cleared and only one of the completion status fields will be asserted. A functional interrupt will be sent to the Exception module whenever the BSY field transitions to the deasserted state.

The completion status fields are prioritized from left to right, with the left most status field holding the highest priority. For example, if the DMA Controller incurs a simultaneous SMA error and an XBT error, then the **DSTA** register will only reflect the SMA completion status. In a similar fashion, if the DMA Controller incurs an XBT error while attempting a commanded abort, then the **DSTA** register will only reflect the XBT completion status.

If the DMA Controller incurs multiple errors that are NOT simultaneously detected, then the **DSTA** register will only reflect status pertaining to the first occurring error. This is of particular importance to the PAU and ABT fields. If an error is detected before the pause or abort takes affect, then the **DSTA** register will only reflect status pertaining to the error.

The fields within the **DSTA** register are defined as follows:

SMA: Signalled Master Abort. This read-only field will be set if the PCI master has signalled a master-abort. An error of this type will cause the DMA Controller to abort the current transaction.

RTA: Received Target Abort. This read-only field will be set if the PCI master has received a target-abort. An error of this type will cause the DMA Controller to abort the current transaction.

MRC: Maximum Retry Count. This read-only field will be set if the PCI master has exceeded the maximum 2^{24} attempts to a target that is continually issuing disconnect-retries. An error of this type will cause the DMA Controller to abort the current transaction.

XBT: PowerPC Bus Time-out. This read-only field will be set if the PowerPC master has received an acknowledge due to a PowerPC bus time-out. An error of this type will cause the DMA Controller to abort the current transaction.

ABT: Abort. This read-only field will be set if the DMA Controller has successfully completed a commanded abort. A successful command abort must meet the following criteria:

- A write of a logic ‘1’ to the **DCTL**.ABT field.
- The DMA Controller has not received any other errors (SMA, RTA, MRC or XBT) between the time the transaction was started and the time that the DMA Controller goes to the idle state.
- The commanded abort took place before the DMA Controller was able to complete a transaction.

PAU: Pause. This read-only field will be set if the DMA Controller has successfully completed a commanded pause. A successful command pause must meet the following criteria:

- A write of a logic ‘1’ to the **DCTL**.PAU field.
- The DMA Controller has not received any other errors (SMA, RTA, MRC or XBT) between the time the transaction was started and the time that the DMA Controller goes to the idle state.
- The DMA Controller has not been issued a commanded abort.

– The commanded pause took place before the DMA Controller was able to complete a transaction.

DON: Done. This read-only field will be set of the DMA Controller has successfully completed a DMA transaction. A successful transaction must meet the following criteria:

– The DMA Controller has not received any other errors (SMA, RTA, MRC or XBT) between the time the transaction was started and the time that the DMA Controller goes to the idle state.

– If a commanded abort was issued, then it did not take affect before the transaction was completed.

– If a commanded pause was issued, then it did not take affect before the transaction was completed.

BSY: Busy. This read-only field reflects the status of the DMA Controller. If set, the DMA Controller is currently processing a DMA transaction. If cleared, the DMA Controller has completed a previous transaction and is now idle.

DMA Source Address Register

Offset	***XCSR* + $260**																												**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DSAD																																**DMA**
Operation	R/W																																
Reset	$00000000																																

The DMA Source Address Register (**DSAD**) contains the source address for a DMA transfer. If the source is PCI space then this field will represent a PCI address. If the source is PowerPC space then this field will represent a PowerPC address.

If the source is a data pattern then this field will represent the beginning pattern. This register is interpreted differently depending on if the data pattern transfer is represented in bytes or words. When the pattern size is

bytes, then the starting pattern is represented by bit positions 24 thru 31 of this register. If the pattern size is words, then the starting pattern is represented by the entire 32-bits of this register.

User software must program this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the source address field of the current descriptor.

DMA Source Attribute Register

Offset	*XCSR* + $264																													Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DSAT																																DMA
			TYP					NIN								PSZ					PRC									CRI		GBL	
Operation	R	R	R/W	R/W	R	R	R	R/W	R	R	R	R	R	R	R	R/W	R	R	R	R	R/W	R/W	R/W	R	R	R	R	R	R	R/W	R	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	

The DMA Source Attribute Register (**DSAT**) contains the source attributes for a DMA transfer. Not all fields are used for all transfer types. Fields that do not pertain to a particular transfer type are ignored.

User software must program this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the source attribute field of the current descriptor.

The fields within the **DSAT** register are defined as followed:

TYP: Type. This field indicates the type of source to be used for a DMA transfer. Different fields within the **DSAT** register are used depending on the type of source selected. The table below shows the different source types and the associated fields within the **DSAT** register that apply.

Table 3-31. DSAT TYP Encoding

TYP	DMA Source	Applicable Fields			
		NIN	PSZ	CRI	GBL
00	PowerPC bus			X	X
01	PCI Bus	X			
1x	Data Pattern	X	X		

NIN: No Increment. If set, source increment will be disabled during a DMA transfer. If a PCI bus source is selected then the source address will not be incremented. If the source is a data pattern, then the data pattern will not be incremented. If cleared, the source will be incremented.

PSZ: Pattern Size. If set, the data size used during Data Pattern transfers will be bytes. If cleared, the data size will be words. This field only applies to the generation of the data patterns used for a transfer. It does not specify how the patterns are actually placed into the destination space. (i.e. selecting a byte pattern size does not result in a stream of single-beat PowerPC or PCI bus cycles)

PRC: PCI Read Command. This field represents the command used during PCI read cycles. Note that this field is only applicable if the TYP field represents a PCI bus DMA source. The encoding of this field matches that described within the section titled “Command Definition” in the *PCI Local Bus Specification*. The following table shows the recommended values for PRC.

Table 3-32. DSAT PRC Encoding

PRC	PCI Command
0010	IO Read
0110	Memory Read
1100	Memory Read Multiple
1110	Memory Read Line

Using an encoding other than the recommended value will result in unpredictable DMA operation.

CRI: Cache-line Read Invalidate. If set, the DMA will use the "Read-with-intent-to-modify" transfer type during PowerPC read cycles. This will force processor cached data in the E and S states to be invalidated during snoop hits. If cleared, the DMA will use the "Read" transfer type. This will allow the processor to retain cached data in the E and S states.

GBL: Global. If set, the DMA will assert the GBL_ pin during PowerPC read cycles. This will allow the processor to snoop the DMA transfer. If cleared, the GBL_ pin will not be asserted and the processor will not be able to snoop the DMA transaction.

DMA Destination Address Register

Offset	*XCSR* + $268																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DDAD																																DMA
Operation	R/W																																
Reset	$00000000																																

The DMA Destination Address Register (**DDAD**) contains the destination address for a DMA transfer. If the destination is PCI space then this field will represent a PCI address. If the destination is PowerPC space then this field will represent a PowerPC address.

User software must program this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the destination address field of the current descriptor.

DMA Destination Attribute Register

Offset	*XCSR* + $26C																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DDAT																																DMA
				TYP				NIN													PWC									CWF	GBL	
Operation	R	R	R	R/W	R	R	R	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R	R	R	R	R	R	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	

The DMA Destination Attribute Register (**DDAT**) contains the destination attributes for a DMA transfer. Not all fields are used for all transfer types. Fields that do not pertain to a particular transfer type are ignored.

User software must program this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the destination attribute field of the current descriptor.

The fields within the **DDAT** register are defined as followed:

TYP: Type. This field indicates the type of destination to be used for a DMA transfer. Different fields within the **DDAT** register are used depending on the type of destination selected. The table below shows the different destination types and the associated fields within the **DDAT** register that apply.

Table 3-33. DDAT TYP Encoding

TYP	DMA Destination	Applicable Fields		
		NIN	CWF	GBL
0	PowerPC bus		X	X
1	PCI Bus	X		

NIN: No Increment. If set, destination increment will be disabled during a DMA transfer. If a PCI bus destination is selected then the destination address will not be incremented. If cleared, the destination will be incremented.

PWC: PCI Write Command. This field represents the command used during PCI write cycles. Note that this field is only applicable if the TYP field represents a PCI bus DMA destination. The encoding of this field matches that described within “Section 3.1.1 Command Definition” of the *PCI Local Bus Specification*. The table below shows the recommended values for PWC.

Table 3-34. DDAT PWC Encoding

PRC	PCI Command
0011	IO Write
0111	Memory Write
1111	Memory Write and Invalidate

Using an encoding other than the recommended value will result in unpredictable DMA operation.

CWF: Cache-line Write Flush. If set, the DMA will use the “Write-with-flush” transfer type during PowerPC burst write cycles. This will force the processor to perform copyback writes during snoop hits. If cleared, the DMA will use the “Write-with-kill” transfer type. This

will force the processor to invalidate it's associated cache entry during snoop hits. This bit provides a work-around solution for processors that cannot support "Write-with-kill" correctly. This bit only applies to burst write cycles since all single beat cycles will use the "Write-with-flush" transfer type.

GBL: Global. If set, the DMA will assert the GBL_ pin during PowerPC write cycles. This will allow the processor to snoop the DMA transfer. If cleared, the GBL_ pin will not be asserted and the processor will not be able to snoop the DMA transaction.

DMA Next Link Address Register

Offset	*XCSR* + $270																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DNLA																																DMA
	NLA																															LLA	
Operation	R/W																								R	R	R	R	R/W				
Reset	$00000000																								0	0	0	0	0				

The DMA Next Link Address Register (**DNLA**) contains information pertaining to the next Linked-List-Mode descriptor.

This register is not used when performing Direct-Mode transactions. When starting a Linked-List-Mode transaction, software will program this register with the address of the first Linked-List-Mode descriptor. When continuing a Linked-List-Mode transaction, the register is automatically loaded from the next link address field of the current descriptor.

The fields within the **DNLA** register are defined as follows:

NLA: Next Link Address. This is the address of the next descriptor when using Linked-List-Mode. This is a PowerPC address and is presented in 32-byte cache-line resolution.

LLA: Last Link Address. If set, the current descriptor will be the last descriptor of a Linked-List transaction. If cleared, the current descriptor will not be the last descriptor.

DMA Count Register

Offset	*XCSR* + $274																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DCNT																																DMA
Operation	R/W																																
Reset	$00000000																																

The DMA Count Register (**DCNT**) contains the byte count for a DMA transfer. Note that a count of all zeros represents the maximum count of 4 GBytes.

User software must program this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the count field of the current descriptor.

DMA Current Source Address Register

Offset	*XCSR* + $280																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DCSA																																DMA
Operation	R																																
Reset	$00000000																																

The DMA Current Source Address Register (**DCSA**) is a read-only register that contains the current source address for a DMA transfer. If the source is PCI space then this field will represent a PCI address. If the source is PowerPC space then this field will represent a PowerPC address. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

During a FIFO fill SMA, RTA, or MRC error with the PCI bus as the transfer source, this register will represent the PCI address at which a read error occurred. During a FIFO fill XBT error with the PowerPC bus as the transfer source, this register will represent the PowerPC address at which a read error occurred.

DMA Current Destination Address Register

Offset	*XCSR* + $284																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DCDA																																DMA
Operation	R																																
Reset	$00000000																																

The DMA Current Destination Address Register (**DCDA**) is a read-only register that contains the current destination address for a DMA transfer. The current destination address is the next address that the DMA Controller is about to attempt. If the destination is PCI space then this field will represent a PCI address. If the destination is PowerPC space then this field will represent a PowerPC address. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

During a FIFO empty SMA, RTA, or MRC error with the PCI bus as the transfer destination, this register will represent the PCI address at which a write error occurred. During a FIFO empty XBT error with the PowerPC bus as the transfer destination, this register will represent the PowerPC address at which a write error occurred.

DMA Current Link Address Register

Offset	*XCSR* + $288																										Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	DCLA																																DMA
Operation	R																											R	R	R	R	R	
Reset	$00000000																											0	0	0	0	0	

The DMA Current Link Address Register (**DCLA**) is a read-only register that contains the current Linked-List-Mode descriptor address for a DMA transfer. This will always represent a PowerPC address and is presented in 32-byte cache-line resolution. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

At the beginning of and up to the first descriptor fetch of a Linked-List transaction, this register will hold the address of the ‘to-be fetched’ descriptor. Once the descriptor has been fetched, this register will hold the address of the next descriptor. If an XBT error occurs during a descriptor fetch, this register will hold the address that incurred the error.

Message Passing

The Message Passing function has registers located within two register groups. The processor may gain access to a majority of the control and status registers located within PowerPC address space as a part of the *XCSR* Register Group. A standard PCI Memory Space interface for I_2O and Generic Message Passing is provided within the *PMEP* Register Group.

XCSR Register Group

The message passing registers associated with the *XCSR* register group are described in the following subsections.

MP Generic Outbound Message (0 and 1) Registers

Offset	**MGOM0: *XCSR* + $290** **MGOM1: *XCSR* + $294**				**Function**
Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	
Name	MGOMx				**Generic**
	BYTE0	BYTE1	BYTE2	BYTE3	
Operation	R/W	R/W	R/W	R/W	
Reset	$00	$00	$00	$00	

The MP Generic Outbound Message Registers (**MGOM0** and **MGOM1**) are used for generating outbound messages from the processor to PCI. Writing to either **MGOM0** or **MGOM1** will cause the generation of a PCI interrupt if the **MGMS** register permits. These registers are visible from within the *PMEP* Register Group, allowing a PCI master to receive the outbound message being passed from the processor.

MP Generic Outbound Doorbell Register

Offset	*XCSR* + $298																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MGOD																																Generic
	ODBI0	ODBI1	ODBI2	ODBI3	ODBI4	ODBI5	ODBI6	ODBI7	ODBI8	ODBI9	ODBI10	ODBI11	ODBI12	ODBI13	ODBI14	ODBI15	ODBI16	ODBI17	ODBI18	ODBI19	ODBI20	ODBI21	ODBI22	ODBI23	ODBI24	ODBI25	ODBI26	ODBI27	ODBI28	ODBI29	ODBI30	ODBI31	
Operation	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The MP Generic Outbound Doorbell Register (**MGOD**) is used for generating outbound doorbell interrupts from the processor to PCI. Writing a one to any bit position will cause the generation of a PCI interrupt if the **MGODM** register permits. This register is visible from within the *PMEP* Register Group, allowing a PCI master to determine which doorbell bit was used to generate the outbound doorbell interrupt.

The fields within the **MGOD** register are defined as follows:

ODBIx: Outbound Doorbell Interrupt. Writing a one to a particular bit position will set the bit and cause the generation of a PCI interrupt.

MP Generic Inbound (0 and 1) Message Registers

Offset	**MGIM0:** ***XCSR*** **+ $2A0** **MGIM1:** ***XCSR*** **+ $2A4**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	MGIMx																																Generic
	BYTE0								BYTE1								BYTE2								BYTE3								
Operation	R/W								R/W								R/W								R/W								
Reset	$00								$00								$00								$00								

The MP Generic Inbound Message Registers (**MGIM0** and **MGIM1**) are used for receiving inbound messages from PCI to the processor. The processor reads the **MGIM0** or **MGIM1** register to obtain the inbound message. These registers are visible from within the *PMEP* Register

Group, allowing a PCI master to write a message to either **MGIM0** or **MGIM1** which will then generate a processor interrupt. The interrupts generated by these registers may be cleared from within the **FECL** register.

MP Generic Inbound Doorbell Register

Offset	***XCSR* + $2A8**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	MGID																															
	IDBI7	IDBI6	IDBI5	IDBI4	IDBI3	IDBI2	IDBI1	IDBI0	IDBI15	IDBI14	IDBI13	IDBI12	IDBI11	IDBI10	IDBI9	IDBI8	IDBI23	IDBI22	IDBI21	IDBI20	IDBI19	IDBI18	IDBI17	IDBI16	IDBI31	IDBI30	IDBI29	IDBI28	IDBI27	IDBI26	IDBI25	IDBI24
Operation	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Generic

The MP Generic Inbound Doorbell Register (**MGID**) is used for receiving inbound doorbell interrupts from PCI to the processor. The processor reads this register to determine which doorbell bit was used to generate an inbound doorbell interrupt. This register is visible from within the *PMEP* Register Group, allowing a PCI master to write any doorbell bit which will then generate an interrupt to the processor. The interrupts generated by this register may be cleared by writing to the appropriate fields within this register.

The fields within the **MGID** register are defined as follows:

IDBIx: Inbound Doorbell Interrupt. If any one of these bits are set from the *PMEP* Register Group, a processor interrupt will be generated. Writing a one to a particular bit position will clear the bit and remove the interrupt associated with the bit.

MP Generic Inbound Doorbell Mask Register

Offset	*XCSR* + $2B0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MGIDM																																Generic
	IDBM7	IDBM6	IDBM5	IDBM4	IDBM3	IDBM2	IDBM1	IDBM0	IDBM15	IDBM14	IDBM13	IDBM12	IDBM11	IDBM10	IDBM9	IDBM8	IDBM23	IDBM22	IDBM21	IDBM20	IDBM19	IDBM18	IDBM17	IDBM16	IDBM31	IDBM30	IDBM29	IDBM28	IDBM27	IDBM26	IDBM25	IDBM24	
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

The MP Generic Inbound Doorbell Mask Register (**MGIDM**) is used for individually masking inbound doorbell interrupts from PCI to the processor. This is the first level of a two level masking scheme. The second level of masking is provided by the **FEMA**.MIDB bit, which provides a global masking of all inbound doorbell interrupts independent of the state of the **MGIDM** register.

The fields within the **MGIDM** register are defined as follows:

IDBMx: Inbound Doorbell Mask. Writing a zero to a particular bit position will enable the generation of the associated inbound doorbell interrupt. Writing a one will mask the interrupt.

MP I_20 Outbound Free_list Head Register

Offset	*XCSR* + $2C0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIOFH																																I_2O
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												1	0			$00000														0	0	

The MP I_2O Outbound Free_list Head Register (**MIOFH**) is a word aligned pointer into PowerPC address space that represents the head location of the Outbound Free_list FIFO. This pointer is automatically

incremented whenever a PCI master performs a write to the Outbound Free_list FIFO. The processor may also modify this register as needed. The fields within the **MIOFH** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP I_20 Outbound Free_list Tail Register

Offset	***XCSR* + $2C4**																														**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIOFT																																**I_2O**
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												1	0			$00000														0	0	

The MP I_2O Outbound Free_list Tail Register (**MIOFT**) is a word aligned pointer into PowerPC address space that represents the tail location of the Outbound Free_list FIFO. This pointer is manually maintained by the processor. The fields within the **MIOFT** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP $I_2$0 Outbound Post_list Head Register

Offset	*XCSR* + $2C8																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIOPH																																I_2O
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												1	1			$00000														0	0	

The MP I_2O Outbound Post_list Head Register (**MIOPH**) is a word aligned pointer into PowerPC address space that represents the head location of the Outbound Post_list FIFO. This pointer is manually maintained by the processor. The fields within the **MIOPH** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP $I_2$0 Outbound Post_list Tail Register

Offset	*XCSR* + $2CC																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIOPT																																I_2O
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												1	1			$00000														0	0	

The MP I_2O Outbound Post_list Tail Register (**MIOPT**) is a word aligned pointer into PowerPC address space that represents the tail location of the Outbound Post_list FIFO. This pointer is automatically incremented whenever a PCI master performs a read from the Outbound Post_list FIFO. The processor may also modify this register as needed. The fields within the **MIOPT** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP $I_2$0 Inbound Free_list Head Register

Offset	*XCSR* + $2D0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIIFH																																I_2O
	QBA																PTR																
Operation	R												R	R			R/W												R	R			
Reset	$000												0	0			$00000												0	0			

The MP I_2O Inbound Free_list Head Register (**MIIFH**) is a word aligned pointer into PowerPC address space that represents the head location of the Inbound Free_list FIFO. This pointer is manually maintained by the processor. The fields within the **MIIFH** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP $I_2$0 Inbound Free_list Tail Register

Offset	*XCSR* + $2D4																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIIFT																																I_2O
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												0	0			$00000														0	0	

The MP I_2O Inbound Free_list Tail Register (**MIIFT**) is a word aligned pointer into PowerPC address space that represents the tail location of the Inbound Free_list FIFO. This pointer is automatically incremented whenever a PCI master performs a read from the Inbound Free_list FIFO. The processor may also modify this register as needed. The fields within the **MIIFT** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP $I_2$0 Inbound Post_list Head Register

Offset	*XCSR* + $2D8																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIOPH																																I_2O
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												1	1			$00000														0	0	

The MP I_2O Inbound Post_list Head Register (**MIIPH**) is a word aligned pointer into PowerPC address space that represents the head location of the Inbound Post_list FIFO. This pointer is automatically incremented

whenever a PCI master performs a write to the Inbound Post_list FIFO. The processor may also modify this register as needed. The fields within the **MIIPH** register are defined as followss:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP I_20 Inbound Post_list Tail Register

Offset	***XCSR* + $2DC**																														**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIIPT																																**I_2O**
	QBA																PTR																
Operation	R												R	R			R/W														R	R	
Reset	$000												0	1			$00000														0	0	

The MP I_2O Outbound Post_list Tail Register (**MIIPT**) is a word aligned pointer into PowerPC address space that represents the tail location of the Inbound Post_list FIFO. This pointer is manually maintained by the processor. The fields within the **MIIPT** register are defined as follows:

QBA: Queue Base Address. This read-only field is a copy of the QBA field within the **MIQB** register.

PTR: Pointer. This is an offset from the QBA where the FIFO component resides.

MP $I_2$0 Control Register

Offset	*XCSR* + $2E0																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MICT																																I_2O
								ENA						QSZ																			
Operation	R	R	R	R	R	R	R	R/W	R	R	R	R	R	R/W	R/W	R/W	R								R								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	$00								$00								

The MP I_2O Control Register (**MICT**) is used for controlling the I_2O Message Passing function. The fields within the **MICT** register are defined as follows:

ENA: Enable. If this bit is set, the I_2O Message Passing function will be enabled. When enabled, the I_2O FIFOs will be accessible to PCI. If cleared, the I_2O Message Passing function is disabled.

QSZ: Queue Size. This field represents the size of the I_2O quad-FIFO circular queues. Each of the four FIFOs are sized the same, with the QSZ field indicating the size of each FIFO. The following table shows the encoding of the QSZ field.

Table 3-35. MICT QSZ Encoding

QSZ	FIFO Size
000	2 KBytes
001	4 KBytes
010	8 KBytes
011	16 KBytes
100	32 KBytes
101	64 KBytes
110	128 KBytes
111	256 KBytes

MP $I_2$0 Queue Base Register

Offset	*XCSR* + $2E4																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MIQB																																I_2O
	QBA																																
Operation	R/W								R								R							R								
Reset	$000								$0								$00								$00								

The MP I_2O Queue Base Register (**MIQB**) represents the base address of the I_2O Quad-FIFO circular queues.The circular queues may be placed anywhere within PowerPC address space on 1MB boundaries. The field within the **MIQB** register is defined as follows:

QBA: Queue Base Address. Base address of circular queue structure.

PMEP Register Group

The following subsections identify and describe the registers within the *PMEP* Register Group.

MP $I_2$0 Interrupt Status Register

Offset	*PMEP* + $030																												Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name																									MIST								I_2O
																													OPI				
Operation	R								R								R							R	R	R	R	R	R	R	R	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The MP I_2O Interrupt Status Register (**MIST**) contains status information for interrupts associated with the Outbound Post_list circular queue. The field within the **MIST** register is defined as follows:

OPI: Outbound Post_list Interrupt. This is a read-only bit indicating a new entry resides within the Outbound Post_list FIFO. If the **MIMS** register permits, the setting of this bit will also generate a PCI interrupt. This bit and the interrupt associated with this bit may be cleared by a PCI master by reading the **MIOQ** register until there are no more new entries within the Outbound Post_list FIFO.

MP I_20 Interrupt Mask Register

Offset	*PMEP* + $034																								Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name																									MIMS								I_2O
																													OPIM				
Operation	R								R								R							R	R	R	R	R/W	R	R	R	
Reset	$00								$00								$00								0	0	0	0	1	0	0	0	

The MP I_2O Interrupt Mask Register (**MIMS**) controls the masking of interrupts associated with the Outbound Post_list circular queue. The field within the **MIMS** register is defined as follows:

OPIM: Outbound Post_list Interrupt Mask. If set, the OPI bit within the **MIST** register will not generate a PCI interrupt. If cleared, a PCI interrupt will be generated whenever the OPI bit is set.

MP I_20 Inbound Queue Register

Offset	*PMEP* + $040																								Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	MIIQ																																I_2O
Operation	R/W																																
Reset	$FFFFFFFF																																

The MP I_2O Inbound Queue Register (**MIIQ**) is an access port to the Inbound Free_list and Post_list FIFOs.

Reading this register will return the oldest MFA entry from the Inbound Free_list FIFO. If there are no entries in the FIFO or the I_2O Message Passing function is disabled then this register will return 0xFFFF_FFFF.

Writing this register will place the newest MFA entry into the Inbound Post_list FIFO. If the I_2O Message Passing function is disabled then the write will be discarded.

MP I_20 Outbound Queue Register

Offset	***PMEP* + $044**																												**Function**
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	MIOQ																																**I_2O**
Operation	R/W																																
Reset	$FFFFFFFF																																

The MP I_2O Outbound Queue Register (**MIOQ**) is an access port to the Outbound Free_list and Post_list FIFOs.

Reading this register will return the oldest MFA entry from the Outbound Post_list FIFO. If there are no entries in the FIFO or the I_2O Message Passing function is disabled then this register will return 0xFFFF_FFFF.

Writing this register will place the newest MFA entry into the Outbound Free_list FIFO. If the I_2O Message Passing function is disabled then the write will be discarded.

MP Generic Outbound Message (0 and 1) Registers

Offset	MGOM0: *PMEP* + $100 MGOM1: *PMEP* + $104				Function
Bit	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
Name	MGOMx				Generic
	BYTE3	BYTE2	BYTE1	BYTE0	
Operation	R/W	R/W	R/W	R/W	
Reset	$00	$00	$00	$00	

The MP Generic Outbound Message Registers (**MGOM0** and **MGOM1**) are used for receiving outbound messages from the processor to PCI. A PCI master reads the **MGOM0** or **MGOM1** register to obtain the outbound message. These registers are visible from within the *XCSR* Register Group, allowing the processor to write a message to either **MGOM0** or **MGOM1** which will then generate a PCI interrupt if the **MGMS** register permits. The interrupt associated with this register may be cleared from within the **MGST** register.

MP Generic Outbound Doorbell Register

Offset	*PMEP* + $108																								Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	MGOD																																Generic
	ODBI24	ODBI25	ODBI26	ODBI27	ODBI28	ODBI29	ODBI30	ODBI31	ODBI16	ODBI17	ODBI18	ODBI19	ODBI20	ODBI21	ODBI22	ODBI23	ODBI8	ODBI9	ODBI10	ODBI11	ODBI12	ODBI13	ODBI14	ODBI15	ODBI0	ODBI1	ODBI2	ODBI3	ODBI4	ODBI5	ODBI6	ODBI7	
Operation	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The MP Generic Outbound Doorbell Register (**MGOD**) is used for receiving outbound doorbell interrupts from the processor to PCI. A PCI master reads this register to determine which doorbell bit was used to generate the outbound doorbell interrupt. This register is visible from within the *XCSR* Register Group, allowing the processor to write any

doorbell bit which will then generate a PCI interrupt, if the **MGODM** register permits. The interrupts generated by this register may be cleared by writing to the appropriate fields within this register.

The fields within the **MGOD** register are defined as follows:

ODBIx: Outbound Doorbell Interrupt. If any one of these bits are set from the *XCSR* Register Group, a PCI interrupt will be generated. Writing a one to a particular bit position will clear the bit and remove the interrupt associated with the bit.

MP Generic Inbound Message (0 and 1) Registers

Offset	**MGIM0: *PMEP* + $110** **MGIM1: *PMEP* + $114**				
Bit	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	**Function**
Name	MGIMx				**Generic**
	BYTE3	BYTE2	BYTE1	BYTE0	
Operation	R/W	R/W	R/W	R/W	
Reset	$00	$00	$00	$00	

The MP Generic Inbound Message Registers (**MGIM0** and **MGIM1**) are used for generating inbound messages from PCI to the processor. Writing to either **MGIM0** or **MGIM1** will cause the generation of a processor interrupt if the ***XCSR*.FEMA** register permits. These registers are visible from within the *XCSR* Register Group, allowing the processor to receive the inbound message being passed from PCI.

MP Generic Inbound Doorbell Register

Offset	*PMEP* + $118																																Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	MGID																																Generic
	IDBI31	IDBI30	IDBI29	IDBI28	IDBI27	IDBI26	IDBI25	IDBI24	IDBI23	IDBI22	IDBI21	IDBI20	IDBI19	IDBI18	IDBI17	IDBI16	IDBI15	IDBI14	IDBI13	IDBI12	IDBI11	IDBI10	IDBI9	IDBI8	IDBI7	IDBI6	IDBI5	IDBI4	IDBI3	IDBI2	IDBI1	IDBI0	
Operation	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	R/S	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The MP Generic Inbound Doorbell Register (**MGID**) is used for generating inbound doorbell interrupts from PCI to the processor. Writing a one to any bit position will cause the generation of a processor interrupt if the ***XCSR*.FEMA** register permits. This register is visible from within the *XCSR* Register Group, allowing the processor to determine which doorbell bit was used to generate the inbound doorbell interrupt.

The fields within the **MGID** register are defined as follows:

IDBIx: Inbound Doorbell Interrupt. Writing a one to a particular bit position will set the bit and cause the generation of a processor interrupt.

MP Generic Interrupt Status Register

Offset	*PMEP* + $120																				Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name																									MGST								Generic
																											OMI1	OMI0					
Operation	R								R								R							R	R	R/C	R/C	R	R	R	R	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The MP Generic Interrupt Status Register (**MGST**) is used for handling interrupt status associated with outbound message passing registers **MGOM0** and **MGOM1**. The fields within the **MGST** register are defined as follows:

OMI1: Outbound Message Interrupt 1. If set, an outbound message has been written by the processor into the **MGOM1** register. If the **MGMS** register allowed it, a PCI interrupt was also generated. This bit and the associated interrupt may be cleared by writing a one to this field.

OMI0: Outbound Message Interrupt 0. If set, an outbound message has been written by the processor into the **MGOM0** register. If the **MGMS** register allowed it, a PCI interrupt was also generated. This bit and the associated interrupt may be cleared by writing a one to this field.

MP Generic Interrupt Mask Register

Offset	*PMEP* + $124																								Function
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name																									MGMS								Generic
																											OMIM1	OMIM0					
Operation	R								R								R							R	R	R/W	R/W	R	R	R	R	
Reset	$00								$00								$00								0	0	1	1	0	0	0	0	

The MP Generic Interrupt Mask Register (**MGMS**) is used for controlling the assertion of exceptions associated with outbound message passing registers **MGOM0** and **MGOM1**. The fields within the **MGMS** register are defined as follows:

OMIM1: Outbound Message Interrupt Mask 1. This bit controls the masking of the **MGOM1** register interrupt. If this bit is cleared, a write to the ***XCSR*.MGOM1** register by the processor will generate a PCI interrupt. If set, no interrupt will be generated.

OMIM0: Outbound Message Interrupt Mask 0. This bit controls the masking of the **MGOM0** register interrupt. If this bit is cleared, a write to the ***XCSR*.MGOM0** register by the processor will generate a PCI interrupt. If set, no interrupt will be generated.

MP Generic Outbound Doorbell Mask Register

Offset	*PMEP* + $128																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Function
Name	MGODM																																Generic
	ODBM24	ODBM25	ODBM26	ODBM27	ODBM28	ODBM29	ODBM30	ODBM31	ODBM16	ODBM17	ODBM18	ODBM19	ODBM20	ODBM21	ODBM22	ODBM23	ODBM8	ODBM9	ODBM10	ODBM11	ODBM12	ODBM13	ODBM14	ODBM15	ODBM0	ODBM1	ODBM2	ODBM3	ODBM4	ODBM5	ODBM6	ODBM7	
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

The MP Generic Outbound Doorbell Mask Register (**MGODM**) is used for individually masking outbound doorbell interrupts from the processor to PCI. The fields within the **MGODM** register are defined as follows:

ODBMx: Outbound Doorbell Mask. Writing a zero to a particular bit position will enable the generation of the associated outbound doorbell interrupt. Writing a one will mask the interrupt.

Multi-Processor Interrupt Controller

The Multi-Processor Interrupt Controller has registers located within two register groups. A majority of the control and status registers are located within PowerPC address space as a part of the *XMPI* Register Group. Some additional control and status, including the relocation of the *XMPI* Register group, is located within PowerPC address space as part of the *XCSR* Register Group.

XCSR Register Group

The following subsections describe the registers in the *XCSR* Register Group.

MPIC Base Address Register

Offset	*XCSR* + $0E0																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MBAR															ENA																	MPIC
Operation	R/W														R	R/W	R								R								
Reset	$0000														0	0	$00								$00								

The MPIC Base Address Register (**MBAR**) is used to control the mapping of the *XMPI* Register Group within PowerPC address space. The fields within the **MBAR** register are defined as follows:

MBAR: MPIC Base Address Register. This field holds the base address of the *XMPI* Register Group. This group may be located anywhere in PowerPC address space on any 256KByte boundary.

ENA: Enable. This field controls the visibility of the *XMPI* Register Group. Setting this field to a one will enable the *XMPI* Register Group to be seen starting at the base address specified by the MBAR field.

Clearing this bit makes the *XMPI* Register Group inaccessible. This field powers up cleared, meaning the *XMPI* Register Group is not visible until programmed by the processor.

MPIC Control and Status/Interrupt Request Sample Registers

Offset	*XCSR* + $0E4																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	MCSR																MIRS																MPIC
		OPI																															
Operation	R	R/W	R	R	R	R	R	R	R								R																
Reset	0	0	0	0	0	0	0	0	$00								$FFFF																

The MPIC Control and Status Register (**MCSR**) provides various control and status of the MPIC function. The field within the **MCSR** register is defined as follows:

OPI: OpenPIC Interrupt. If set, the Harrier generated interrupts will be passed on to the MPIC. If cleared, the Harrier generated interrupts will be passed on to the IRQ0_ pin. Note that the MPIC can only drive the IRQx_ lines, therefore if this bit is cleared then MPIC is essentially disabled.

The MPIC Interrupt Request Sample Register (**MIRS**) is used to determine the actual state of the EXTI pins. Bit position 16 reflects the state of the EXTI15 pin, and bit position 31 reflects the state of the EXTI0 pin. Note that this register samples the same EXTI lines that are routed to the MPIC, therefore the effects of MPIC interrupt conditioning (i.e. edge detection, signal polarity, etc.) will not be reflected in this register.

XMPI Register Group

The following subsections describe the registers within the *XMPI* Register Group.

Feature Reporting Register

Offset	*XMPI* + $01000																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	FREP																														Misc.
									NIRQ												NCPU								VID		
Operation	R								R								R	R	R		R								R		
Reset	$0								$00F								0	0	0		$01								$03		

The Feature Reporting Register (**FREP**) is a read-only register that contains MPIC sizing and version information. The fields within the **FREP** register are defined as follows:

NIRQ: Number of IRQ's. The number of the highest external IRQ source supported. The IPI, Timer, and Harrier Detected Error interrupts are excluded from this count.

NCPU: Number of CPU's. The number of the highest physical CPU supported. There are two CPU's supported by this design. CPU #0 and CPU #1.

VID: Version ID. Version ID for this Interrupt Controller. This value reports what level of the MPIC specification is supported by this implementation. Version level of 02 is used for the initial release of the MPIC specification.

Global Configuration Register

Offset	*XMPI* + $01020																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	GLBC																																Misc.
	RESET		M	TIE																													
Operation	C	R	R/W	R/W	R	R	R	R	R								R								R								
Reset	0	0	0	0	0	0	0	0	$00								$00								$00								

The Global Configuration Register (**GLBC**) contains global MPIC control information. The fields within the **GLBC** register are defined as follows:

RESET: **Reset Controller**. Writing a one to this bit forces the controller logic to be reset. This bit is cleared automatically when the reset sequence is complete. While this bit is set, the values of all other register are undefined.

M: Cascade Mode. Allows cascading of an external 8259 pair connected to the first interrupt source input pin (0). In the pass through mode, interrupt source 0 is passed directly through to the processor 0 INT pin. The MPIC is essentially disabled. In the mixed mode, 8259 interrupts are delivered using the priority and distribution mechanism of the MPIC. The Vector/Priority and Destination registers for interrupt source 0 are used to control the delivery mode for all 8259 generated interrupt sources.

Table 3-36. Cascade Mode Encoding

M	MODE
0	Pass Through
1	Mixed

TIE: Tie Mode. Writing a one to this register bit will cause a tie in external interrupt processing to, swap back and forth between processor 0 and 1. The first tie in external interrupt processing always

goes to Processor 0 after a reset. When this register bit is set to 0, a tie in external interrupt processing will always go to processor 0 (Mode used on Version $02 of the MPIC).

Table 3-37. Tie Mode Encoding

T	MODE
0	Processor 0 always selected
1	Swap between Processor's

Vendor Identification Register

Offset	*XMPI* + $01080				Function
Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	
Name	VENI				Misc.
		STP			
Operation	R	R	R	R	
Reset	$00	$00	$00	$00	

The Vendor Identification Register (**VENI**) is a read-only register that returns vendor identification. There are two fields in the **VENI** which are not defined for the MPIC implementation but are defined in the MPIC specification. They are the vendor identification and device ID fields. The field within the **VENI** register is defined as follows:

STP: Stepping. The stepping or silicon revision number of the MPIC.

Processor Init Register

Offset	*XMPI* + $01090																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	PINT																														P1	P0	Misc.
Operation	R								R								R							R	R	R	R	R	R	R/W	R/W	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The Processor Init Register (**PINT**) is used to assert a soft reset to the processors. The fields within the **PINT** register are defined as follows:

P1: Processor 1. Writing a one to P1 will assert the Soft Reset input of processor 1 (SRST1_). Writing a zero to P1 will negate the SRST1_ signal.

P0: Processor 0. Writing a one to P0 will assert the Soft Reset input of processor 0 (SRST0_). Writing a zero to P0 will negate the SRST0_ signal.

The Soft Reset input to the PowerPC processor is negative edge-sensitive.

IPI Vector/Priority (0, 1, 2, and 3) Registers

Offset	IPVP0: *XMPI* + $010A0 IPVP1: *XMPI* + $010B0 IPVP2: *XMPI* + $010C0 IPVP3: *XMPI* + $010D0																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	IPVPx																																IPI
	MASK	ACT											PRIOR												VECTOR								
Operation	R/W	R	R	R	R	R	R	R	R				R/W				R								R/W								
Reset	1	0	0	0	0	0	0	0	$0				$0				$00								$00								

The IPI Vector/Priority Registers (**IPVP0**, **IPVP1**, **IPVP2**, and **IPVP3**) establish vectoring and priority information for IPI interrupts. The fields within the **IPVPx** registers are defined as follows:

MASK: Mask. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT: Activity. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

PRIOR: Priority. Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR: **Vector**. This vector is returned when the Interrupt Acknowledge register is examined when the interrupt associated with this vector is requested.

Spurious Vector Register

Offset	*XMPI* + $010E0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	SPVE																																Spur
																									VECTOR								
Operation	R								R								R							R/W								
Reset	$00								$00								$00								$FF								

The Spurious Vector Register (**SPVE**) contains vector information for spurious interrupts. The field within the **SPVE** register is defined as follows:

VECTOR: **Vector**. This vector is returned when the Interrupt Acknowledge register is read during a spurious vector fetch.

Timer Frequency Register

Offset	*XMPI* + $010F0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	TIFR																																Timers
Operation	R/W																																
Reset	$00000000																																

The Timer Frequency Register (**TIFR**) is used to report the frequency (in Hz) of the clock source for the global timers. Following reset, this register contains zero. System initialization code must initialize this register to one-eighth the MPIC clock frequency (PowerPC (60x) bus clock frequency). For the MPIC implementation of the Harrier, a typical value would be $BEBC20 which is 100/8 MHz or 12.5 MHz.

Timer Current Count (0, 1, 2, and 3) Registers

Offset	TICC0: *XMPI* + $01100 TICC1: *XMPI* + $01140 TICC2: *XMPI* + $01180 TICC3: *XMPI* + $011C0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	TICCx																																Timers
	T	CC																															
Operation	R	R																															
Reset	0	$00000000																															

The Timer Current Count Registers (**TICC0**, **TICC1**, **TICC2**, and **TICC3**) are read-only registers that return the current count value of the MPIC timers. The fields within the **TICCx** registers are defined as follows:

T: Toggle. This bit toggles when ever the current count decrements to zero. This bit is cleared when a value is written into the corresponding Timer Base Count Register (TIBC) and the Count Inhibit (CI) bit in the corresponding TIBC register transitions from a 1 to a 0.

CC: Current Count. The Current Count field decrements while the Count Inhibit bit in the Base Count register is zero. When the Current Count register counts down to zero, the Current Count register is reloaded from the Base Count register and the timer's interrupt becomes pending in MPIC processing.

Timer Base Count (0, 1, 2, and 3) Registers

Offset	TIBC0: *XMPI* + $01110 TIBC1: *XMPI* + $01150 TIBC2: *XMPI* + $01190 TIBC3: *XMPI* + $011D0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	TIBCx																																Timers
	CI	BC																															
Operation	R/W	R/W																															
Reset	1	$00000000																															

The Timer Base Count Registers (**TIBC0**, **TIBC1**, **TIBC2**, and **TIBC3**) are programmed with the base count value of the MPIC timers. The fields within the **TIBCx** registers are defined as follows:

CI: Count Inhibit. Setting this bit to one inhibits counting for this timer. Setting this bit to zero allows counting to proceed.

BC: Base Count. This field contains the 31bit count for this timer. When a value is written into this register and the CI bit transitions from a 1 to a 0, it is copied into the corresponding Current Count register and the toggle bit in the Current Count register is cleared. When the Current Count register counts down to zero, the Current Count register is reloaded from the Base Count register and the timer's interrupt becomes pending in MPIC processing.

Timer Vector/Priority (0, 1, 2, and 3) Registers

Offset	TIVP0: *XMPI* + $01120 TIVP1: *XMPI* + $01160 TIVP2: *XMPI* + $011A0 TIVP3: *XMPI* + $011E0																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	TIVPx																																Timers
	MASK	ACT								PRIOR															VECTOR								
Operation	R/W	R	R	R	R	R	R	R	R	R/W							R								R/W								
Reset	1	0	0	0	0	0	0	0	$0								$0								$00								$00

The Timer Vector/Priority Registers (**TIVP0**, **TIVP1**, **TIVP2**, and **TIVP3**) establish vectoring and priority information for timer interrupts. The fields within the **TIVPx** registers are defined as follows:

MASK: Mask. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT: Activity. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

PRIOR: Priority. Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR: **Vector**. This vector is returned when the Interrupt Acknowledge register is examined when the interrupt associated with this vector is acknowledged.

Timer Destination (0, 1, 2, and 3) Registers

Offset	TIDE0: *XMPI* + $01130 TIDE1: *XMPI* + $01170 TIDE2: *XMPI* + $011B0 TIDE3: *XMPI* + $011F0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	TIDEx																																Timers
																														P1	P0		
Operation	R								R								R							R						R/W	R/W		
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The Timer Destination Registers (**TIDE0**, **TIDE1**, **TIDE2**, and **TIDE3**) indicate the destination for timer interrupts. Timer interrupts operate in the Directed delivery interrupt mode. This register may specify multiple destinations (multicast delivery). The fields within the **TIDEx** registers are defined as follows:

P1: Processor 1. The interrupt is directed to processor 1.

P0: Processor 0. The interrupt is directed to processor 0.

External Source Vector/Priority (0 through 15) Registers

Offset	EXVP0: *XMPI* + $10000 EXVP1: *XMPI* + $10020 ...etc... EXVP14: *XMPI* + $101C0 EXVP15: *XMPI* + $101E0																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	EXVPx																												External				
	MASK	ACT							POL	SENSE			PRIOR												VECTOR								
Operation	R/W	R	R	R	R	R	R	R	R/W	R/W	R	R	R/W				R								R/W								
Reset	1	0	0	0	0	0	0	0	0	0	0	0	$0				$00								$00								

The External Source Vector/Priority Registers (**EXVP0** through **EXVP15**) establish vectoring and priority information for external interrupts. The fields within the **EXVPx** registers are defined as follows:

MASK: Mask. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT: Activity. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

POL: Polarity. This bit sets the polarity for external interrupts. Setting this bit to a zero enables active low or negative edge. Setting this bit to a one enables active high or positive edge. Only External Interrupt Source 0 uses this bit in this register. For external interrupt sources 1 through 15, this bit is hard-wired to 0.

SENSE: Sense. This bit sets the sense for external interrupts. Setting this bit to a zero enables edge sensitive interrupts. Setting this bit to a one enables level sensitive interrupts. For external interrupt sources 1

through 15, setting this bit to a zero enables positive edge triggered interrupts. Setting this bit to a one enables active low level triggered interrupts.

PRIOR: Priority. Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR: **Vector**. This vector is returned when the Interrupt Acknowledge register is examined when the interrupt associated with this vector is acknowledged.

External Source Destination (0 through 15) Registers

Offset	EXDE0: *XMPI* + $10010 EXDE1: *XMPI* + $10030 ...etc... EXDE14: *XMPI* + $101D0 EXDE15: *XMPI* + $101F0																													Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	EXDEx																																External
																															P1	P0	
Operation	R								R								R							R	R	R	R	R	R	R/W	R/W	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The External Source Destination Registers (**EXDE0** through **EXDE15**) indicate the destination of external interrupts. These interrupts operate in the Distributed interrupt delivery mode. The fields of the **EXDEx** registers are defined as follows:

P1: Processor 1. The interrupt is pointed to processor 1.

P0: Processor 0. The interrupt is pointed to processor 0.

Harrier Internal Functional/Error Interrupt Vector/Priority Register

Offset	IFEVP: *XMPI* + $10200 IEEVP: *XMPI* + $10220																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	IFEVP/IEEVP																																Internal
	MASK	ACT								SENSE			PRIOR												VECTOR								
Operation	R/W	R	R	R	R	R	R	R	R	R	R	R	R/W				R								R/W								
Reset	1	0	0	0	0	0	0	0	0	1	0	0	$0				$00								$00								

The Harrier Internal Functional/Error Interrupt Vector/Priority Register (**IFEVP/IEEVP**) establishes vectoring and priority information for the Harrier internal functional/error interrupts. An internal interrupt is an interrupt that is generated within the Harrier. The fields within the **IFEVP/IEEVP** registers are defined as follows:

MASK: Mask. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT: Activity. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set

SENSE: Sense. This bit sets the sense for the Harrier internal interrupts/errors. This bit is hard-wired to 1 to enable active-low level sensitive interrupts/errors.

PRIOR: Priority Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR: **Vector**. This vector is returned when the Interrupt Acknowledge register is examined when the interrupt associated with this vector is acknowledged.

Harrier Internal Functional/Error Interrupt Destination Register

Offset	*XMPI* + $10210 *XMPI* + $10230																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	IFEDE/IEEDE																														P1	P0	Internal
Operation	R								R								R							R	R	R	R	R	R	R/W	R/W	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The Harrier Internal Functional/Error Interrupt Destination Register (**IFEDE/IEEDE**) indicates the destination of internal interrupts. These interrupts operate in the Distributed interrupt delivery mode. The fields of the **IFEDE/IEEDE** register are defined as follows:

P1: Processor 1. The interrupt is pointed to processor 1.

P0: Processor 0. The interrupt is pointed to processor 0.

Processor 0/Processor 1 IPI Dispatch (0, 1, 2, and 3) Registers

Offset	**P0IPD0: *XMPI* + $20040** **P0IPD1: *XMPI* + $20050** **P0IPD2: *XMPI* + $20060** **P0IPD3: *XMPI* + $20070** **P1IPD0: *XMPI* + $21040** **P1IPD1: *XMPI* + $21050** **P1IPD2: *XMPI* + $21060** **P1IPD3: *XMPI* + $21070**																																Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	P0IPDx/P1IPDx																																**CPU 0 or CPU 1**
																															P1	P0	
Operation	R								R								R							R	R	R	R	R	R	W	W	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The Processor 0/Processor 1 IPI Dispatch Registers (**P0IPD0**, **P0IPD1**, **P0IPD2**, **P0IPD3**, **P1IPD0**, **P1IPD1**, **P1IPD2**, and **P1IPD3**) are used to send interrupts to one or more processors. Writing to an IPI Dispatch Register causes an interprocessor interrupt request to be sent to one or more processors. A processor is interrupted if the bit in the IPI Dispatch Register corresponding to that processor is set during the write. Two processors are supported, with each processor owning four dispatch registers. Reading these registers returns zeros. The fields within the **P0IPDx/P1IPDx** registers are defined as follows:

P1: Processor 1. The interrupt is directed to processor 1.

P0: Processor 0. The interrupt is directed to processor 0.

Processor 0/Processor 1 Current Task Priority Registers

Offset	P0CTP: *XMPI* + $20080 P1CTP: *XMPI* + $21080																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	P0CTP/P1CTP																																CPU 0 or CPU 1
																														TP			
Operation	R								R								R							R						R/W			
Reset	$00								$00								$00								$0						$F			

The Processor 0/Processor 1 Current Task Priority Registers (**P0CTP** and **P1CTP**) establish a task priority level for a processor. Two processors are supported, with each processor owning a task priority register. Priority levels from 0 (lowest) to 15 (highest) are supported. Setting the current task priority register to 15 masks all interrupts to this processor. Hardware will set the current task register to $F when it is reset or when the Init bit associated with this processor (P0 or P1 in PINT register) is written to a one. The fields within the **P0CTP**/**P1CTP** registers are defined as follows:

TP: Task Priority of processor.

Processor 0/Processor 1 Interrupt Acknowledge Registers

Offset	P0IAC: *XMPI* + $200A0 P1IAC: *XMPI* + $210A0																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	P0IAC/P1IAC																																CPU 0 or CPU 1
																									VECTOR								
Operation	R								R								R							R									
Reset	$00								$00								$00								$FF									

The Processor 0/Processor 1 Interrupt Acknowledge Registers (**P0IAC** and **P1IAC**) are used to declare an interrupt acknowledge. Two processors are supported, with each processor owning an interrupt acknowledge register. On PowerPC-based systems, Interrupt Acknowledge is implemented as a read request to a memory-mapped Interrupt Acknowledge register. Reading the Interrupt Acknowledge register

returns the interrupt vector corresponding to the highest priority pending interrupt. Reading this register without a pending interrupt will return a value of $FF hex. Reading this register also has the following side affects.

- The associated bit in the Interrupt Pending Register is cleared.
- The In-Service register is updated.

The fields within the **P0IAC**/**P1IAC** registers are defined as follows:

VECTOR: **Vector**. This vector is returned when the Interrupt Acknowledge register is read.

Processor 0/Processor 1 End-Of-Interrupt Registers

Offset	**P0EOI: *XMPI* + $200B0** **P1EOI: *XMPI* + $210B0**																												**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	P0EOI/P1EOI																																**CPU 0 or CPU 1**
																																EOI	
Operation	R								R								R							R								W	
Reset	$00								$00								$00								$0								$0	

The Processor 0/Processor 1 End-Of-Interrupt Registers (**P0EOI** and **P1EOI**) are used to declare the end of an interrupt. Two processors are supported, with each processor owning an End-Of-Interrupt register. The field within the **P0EOI**/**P1EOI** registers is defined as follows:

EOI: End of Interrupt. EOI Code values other than 0 are currently undefined. Data values written to this register are ignored, zero is assumed. Writing to this register signals the end of processing for the highest priority interrupt currently in service by the associated processor. The write operation will update the In-Service register by retiring the highest priority interrupt. Reading this register returns zeros.

I²C Controller

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

I²C Clock Prescaler Register

Offset	***XCSR* + $180** ***XCSR* + $1A0**																														**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name																	I2PS0 I2PS1																**I2C**
Operation	R								R								R/W															
Reset	$00								$00								$01F3															

The I2C Clock Prescaler Register (**I2PSx**) is used to specify a frequency of the I²C gated clock signal. The formula for calculating a value is shown below:

I²C CLOCK = SYSTEM CLOCK / (**I2PSx**+1) / 2

After power-up, **I2PSx** is initialized to $01F3 which produces a 100KHz I²C gated clock signal based on a 100.0 MHz system clock. Due to the SDAx hold time requirement, the value written to **I2PSx** *must* be greater than $0020. **Writes to this register will be restricted to 2, 4, or 8-byte only and must be aligned.**

I²C Control Register

Offset	*XCSR* + $184 *XCSR* + $1A4																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name																									I2CO0 I2CO1								I2C
																													STA	STP	ACKO	ENA	
Operation	R								R								R							R	R	R	R	R/W	R/W	R/W	R/W	
Reset	$00								$00								$00								0	0	0	0	0	0	0	0	

The I2C Control Register (**I2COx**) provides control for the Harrier's I²C function. Please refer to the section titled "I²C Interface" in the previous chapter for more information. **Writes to this register are restricted to 4-bytes only and must be aligned.** The fields within the **I2COx** register are defined as follows:

STA: **Start**. When set, the I²C master controller generates a start sequence on the I²C bus on the next write to the **I2TDx** Register and clears the CMP bit in the **I2STx** register. After the start sequence and the **I2TDx** Register contents have been transmitted, the I²C master controller will automatically clear the STA bit and then set the CMP bit in the **I2STx** Register.

STP: **Stop**. When set, the I²C master controller generates a stop sequence on the I²C bus on the next dummy write (data=don't care) to the **I2TDx** Register and clears the CMP bit in the **I2STx** Register. After the stop sequence has been transmitted, the I²C master controller will automatically clear the STP bit and then set the CMP bit in the **I2STx** Register.

ACKO: **Acknowledge Out.** When set, the I²C master controller generates an acknowledge on the I²C bus during read cycles. This bit should be used *only* in the I²C *sequential* read operation and *must*

remain cleared for all other I^2C operations. For I^2C sequential read operation, this bit should be set for every single byte received except on the last byte in which case it should be cleared.

ENA: **Enable.** When set, the I^2C master interface will be enable for I^2C operations. If clear, reads and writes to all I^2C registers are still allowed but no I^2C bus operations will be performed.

I^2C Transmitter Data Register

Offset	*XCSR* + $18C *XCSR* + $1AC																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name																									I2TD0 I2TD1							I2C
Operation	R								R								R							R/W								
Reset	$00								$00								$00								$00								

The I2C Transmitter Data Register (**I2TDx**) contains the transmit byte for I^2C data transfers. If a value is written to **I2TDx** when the STA and ENA bits in the **I2COx** Register are set, a start sequence is generated immediately followed by the transmission of the contents of the **I2TDx** to the responding slave device. The **I2TDx[24:30]** is the device address, and the **I2TDx[31]** is the WR/RD bit (0=WRite, 1=ReaD). After a start sequence with **I2TDx[31]**=0, subsequent writes to the **I2TDx** Register will cause the contents of **I2TDx** to be transmitted to the responding slave device. After a start sequence with **I2TDx[31]**=1, subsequent writes to the **I2TDx** Register (data=don't care) will cause the responding slave device to transmit data to the **I2RDx** Register. If a value is written to **I2TDx** (data=don't care) when the STP and ENA bits in the **I2COx** Register are set, a stop sequence is generated.

I²C Status Register

Offset	*XCSR* + $194 *XCSR* + $1B4																														Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name																									I2ST0 I2ST1								I2C
																													DIN	ERR	ACKI	CMP	
Operation	R								R								R							R	R	R	R	R	R	R	R	
Reset	$00								$00								$00								0	0	0	0	0	0	0	1	

The I2C Status Register (**I2STx**) provides status for the Harrier's I²C function. Please refer to the section titled "I²C Interface" in the previous chapter for more information. The fields within the **I2STx** register are defined as follows:

DIN: **Data In**. This bit is set whenever the I²C master controller has successfully received a byte of read data from an I²C bus slave device. This bit is cleared after the **I2RDx** Register is read.

ERR: **Error**. This bit is set when both STA and STP bits in the **I2COx** Register are set at the same time. The I²C master controller will then clear the contents of the **I2COx** Register, and further writes to the **I2COx** Register will not be allowed until after the **I2STx** Register is read. A read from the **I2STx** Register will clear this bit.

ACKI: **Acknowledge In**. This bit is set if the addressed slave device is acknowledged to either start sequence or data writes from the I²C master controller and cleared otherwise. The I²C master controller will automatically clear this bit at the beginning of the next valid I²C operation.

CMP: **Complete**. This bit is set after the I²C master controller has successfully completed the requested I²C operation and cleared at the beginning of every valid I²C operation. This bit is also set after power-up.

I²C Receiver Data Register

Offset	*XCSR* + $19C *XCSR* + $1BC																												Function
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name																									I2RD0 I2RD1							**I2C**
Operation	R								R								R							R								
Reset	$00								$00								$00								$00								

The I2C Receiver Data Register (**I2RDx**) contains the receive byte for I²C data transfers. During I²C *sequential* read operation, the current receive byte must be read before any new one can be brought in. A read of this register will automatically clear the DIN bit in the **I2STx** Register.

UART Controller

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group, and only **aligned** read/write accesses are allowed.

UART Core Registers

Offset	***XCSR* + $0C0** ***XCSR* + $0C8**																																**Function**
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Name	RTDL0 RTDL1								IEDH0 IEDH1								IDFC0 IDFC1								LCTL0 LCTL1								**UART**
													EDSSI	ELSI	FTBEI	ERBFI	FENS1/RFTL1	FENS0/RFTL0			IID2/DMAS	IID1/TFR	IID0/RFR	IPEN/FIFOEN	DLAB	SB	SP	EPS	PEN	STB	WLS1	WLS0	
Operation	R/W								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	$00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	

The Receiver Buffer/Transmitter Holding/Divisor Latch Low Register (**RTDLx**) consists of three separate registers whose accessibility is based on the state of the **DLAB** bit in the Line Control Register.

When **DLAB** = 0:

RTDLx: **RTDLx** is the *read-only* Receiver Buffer Register that contains the data byte transmitted to the serial port. The data in this register is valid only if the **DR** bit in the Line Status Register (**LSTAx**) is set. In the non-FIFO mode (**FIFOEN** = 0 in the FIFO Control Register), the data in this register must be read before the next data arrives, otherwise it will be overwritten. In the FIFO mode (**FIFOEN** = 1 in the FIFO Control Register), this register accesses the top of the receiver FIFO. If the receiver FIFO is full, the data already in the FIFO will be preserved, but any new incoming data will be lost.

RTDLx: **RTDLx** is the *write-only* Transmitter Holding Register that contains the data to be transmitted from the serial port. In the non-FIFO mode (**FIFOEN** = 0 in the FIFO Control Register), a byte written to this register will be preserved if the **THRE** bit in the Line Status Register (**LSTAx**) is set. If the **THRE** bit is cleared, a write to this register causes the existing data in the register to be overwritten. In the FIFO mode (**FIFOEN** = 1 in the FIFO Control Register), up to 16 bytes of data may be written to this register before the FIFO is full, after which any new data written to this register will be lost.

When **DLAB** = 1:

RTDLx: **RTDLx** is the Divisor Latch Low (least significant byte) Register that contains the lower 8 bits of the baud rate divisor for the UART. This register in conjunction with the Divisor Latch High Register form a 16-bit baud rate divisor. The output baud rate is calculated as follows:

BAUD RATE = OSCILLATOR FREQUENCY / (16 * BAUD RATE DIVISOR)

The Interrupt Enable/Divisor Latch High Register (**IEDHx**) consists of two separate registers whose accessibility is based on the state of the **DLAB** bit in the Line Control Register.

When **DLAB** = 0:

IEDHx: **IEDHx** is the Interrupt Enable Register which contains four bits that control the generation of UART interrupts. Each interrupt can be indicated as active in the Interrupt Identification Register (**IDFCx**) and can individually activate the interrupt output signal. The fields within this register are defined as follows:

EDSSI: **Enable MODEM Status Interrupt.** This bit, when set, enables the MODEM Status Interrupt.

ELSI: **Enable Receiver Line Status Interrupt**. This bit, when set, enables the Receiver Line Status Interrupt.

ETBEI: **Enable Transmitter Holding Register Empty Interrupt**. This bit, when set, enables the Transmitter Holding Register Empty Interrupt.

ERBFI: **Enable Received Data Available Interrupt**. This bit, when set, enables the Received Data Available Interrupt.

When **DLAB** = 1:

IEDHx: **IEDHx** is the Divisor Latch High (most significant byte) Register that contains the upper 8 bits of the baud rate divisor for the UART. This register in conjunction with the Divisor Latch Low Register form a 16-bit baud rate divisor.

The Interrupt Identification/FIFO Control Register (**IDFCx**) consists of two separate registers whose accessibility is based on the mode of operation (either read or write).

IDFCx: **IDFCx** is the *read-only* Interrupt Identification Register that identifies the source of an interrupt. The fields within this register are defined as follows:

FENS1-0: **FIFOs Enabled Status**. These bits are the FIFO status bits that are shown in the table below.

Table 3-38. FENS1-0 Status

FIFOEN	FENS1-0	Status
0	00	FIFOs Disabled
1	11	FIFOs Enabled

IID2-0,IPEN: Interrupt ID, Interrupt Pending. These bits identify the interrupt control functions that are shown in the table below.

Table 3-39. UART Interrupt Control Functions

IID2	IID1	IID0	IPEN	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	---	No Interrupt Pending	---	---
0	1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register

Table 3-39. UART Interrupt Control Functions (Continued)

IID2	IID1	IID0	IPEN	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	0	0	Second	Received Data Available	Receiver Data Available or Trigger Level reached	Reading the Receiver Buffer Register or the FIFO drops below the Trigger Level
1	1	0	0	Second	Character Time-out Indication	No characters have been removed from or input to the RCVR FIFO during the last 4 Char. Times and there is at least 1 Char. in it during this time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the Interrupt Identification Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	0	Fourth	MODEM Status	Clear To Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

IDFCx: **IDFCx** is the *write-only* FIFO Control Register that is used to enable the FIFOs, clear the FIFOs, set the receiver FIFO trigger level, and select the type of DMA signalling. The fields within this register are defined as follows:

RFTL1-0: **Receiver Trigger Level**. These bits are used to set the trigger level (as shown in the following table) for the Received Data Available interrupt (**ERBFI**).

Table 3-40. Receiver FIFO Trigger Level

RFTL1	RFTL0	Receiver FIFO Trigger Level
0	0	1 byte
0	1	4 bytes
1	0	8 bytes
1	1	16 bytes

DMAS: **DMA Mode Select**. This bit determines the DMA mode in which the TXRDYx_ and RXRDYx_ pins support. When this bit is cleared, the device operates in DMA Mode 0. When this bit is set, the device operates in DMA mode 1. This bit has no effect unless the **FIFOEN** bit is set as well.

TFR: **Transmitter FIFO Reset**. Writing a 1 to this bit clears all the bytes in the Transmitter FIFO and resets its counter logic to 0. This bit is self-clearing.

RFR: **Receiver FIFO Reset**. Writing a 1 to this bit clears all the bytes in the Receiver FIFO and resets its counter logic to 0. This bit is self-clearing.

FIFOEN: **FIFO Enable**. Writing a 1 to this bit enables both the Transmitter and Receiver FIFOs. This bit must be a 1 when other bits in this register are written to or they will not be programmed.

The Line Control Register (**LCTLx**) controls the format of the data that is transmitted and received by the UART. It also contains a bit that controls the data loaded into the Divisor Latch High/Low Registers. The fields within the **LCTLx** register are defined as follows:

DLAB: **Divisor Latch Access Bit**. This bit must be set to logic 1 to access the Divisor Latches of the baud rate generator during a read or write operation. It must be set to logic 0 to access the Receiver Buffer, the Transmitter Holding, or the Interrupt Enable Register.

SB: **Set Break**. This bit causes a break condition to be transmitted to the receiving UART. When it is set to a logic 1, the serial output (SOUTx) is forced to the logic 0 state. The break is disabled by setting this bit to a logic 0.

SP: **Stick Parity**. When this bit is set to a logic 1, the parity bit is forced into a defined state as follows:

If **EPS** = 1 and **PEN** = 1, the parity bit is transmitted and checked as a logic 0.

If **EPS** = 0 and **PEN** =1, the parity bit is transmitted and checked as a logic 1.

EPS: **Even Parity Select**. When **EPS** = 0 and **PEN** = 1, an odd number of ones is transmitted or checked. When **EPS** = 1 and **PEN** = 1, an even number of ones is transmitted or checked.

PEN: **Parity Enable**. When this bit is set to logic 1, a parity bit is generated (transmit data) or checked (receive data) between the last data word bit and stop bit of the serial data.

STB: **Number Of Stop Bits**. When this bit is set to logic 1, two stop bits are added after each transmitted character, unless the character length is 5 then only one and a half stop bits are added. When this bit is set to logic 0, one stop bit is always added. Note that only transmitted stop bits are programmable. The receiver checks only the first stop bit, so this bit has no effect when the UART receives the data.

WLS1-0: **Word Length Select**. These two bits specify the number of bits per character that are transmitted or received. The encoding of the **WLS1-0** bits is shown in the following table.

Table 3-41. WLS1-0 Encoding

WLS1	WLS0	Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

Offset	*XCSR* + $0C4 *XCSR* + $0CC																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	MCTL0 MCTL1								LSTA0 LSTA1								MSTA0 MSTA1								SCRT0 SCRT1							
				LOOP	OUT2	OUT1	RTS	DTR	FERR	TEMT	THRE	BI	FE	PE	OE	DR	DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS							
Operation	R	R	R	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W						
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	$XX							

Function
UART

The MODEM Control Register (**MCTLx**) controls the interface with the MODEM or a peripheral device emulating a MODEM. The fields within the **MCTLx** register are defined as follows:

LOOP: **Loopback Mode**. This bit provides a local loopback feature for diagnostic testing of the UART. In the loopback mode, the receiver and transmitter interrupts are fully operational. When this bit is set to logic 1, the following conditions occur:

a. SOUTx is set to logic 1 state.

b. SINx is disconnected.

c. The output of the Transmitter Shift Register is looped back into the Receiver Shift Register input.

d. The four MODEM control inputs (DSRx_, CTSx_, RIx_, and DCDx_) are disconnected.

e. The four MODEM control outputs (DTRx_, RTSx_, OUT1Ux_, and OUT2Ux_) are internally connected to the four MODEM control inputs.

OUT2: **Output 2 signal**. This bit controls the Output 2 (OUT2Ux_) signal which is an auxiliary user-designated output. When this bit is set to logic 0, the OUT2Ux_ output is forced to a logic 1. When this bit is set to logic 1, the OUT2Ux_ output is forced to a logic 0.

OUT1: **Output 1 signal**. This bit controls the Output 1 (OUT1Ux_) signal which is an auxiliary user-designated output. When this bit is set to logic 0, the OUT1Ux_ output is forced to a logic 1. When this bit is set to logic 1, the OUT1Ux_ output is forced to a logic 0.

RTS: **Request To Send output**. This bit controls the Request To Send (RTSx_) output. When this bit is set to logic 0, the RTSx_ output is forced to a logic 1. When this bit is set to logic 1, the RTSx_ output is forced to a logic 0.

DTR: **Data Terminal Ready output**. This bit controls the Data Terminal Ready (DTRx_) output. When this bit is set to logic 0, the DTRx_ output is forced to a logic 1. When this bit is set to logic 1, the DTRx_ output is forced to a logic 0.

The Line Status Register (**LSTAx)** contains the status of the data transfer. The fields within the **LSTAx** register are defined as follows:

FERR: **Error in Receiver FIFO**. This bit is set to 1 when there is at least one **PE**, **FE** or **BI** in the Receiver FIFO. It is cleared by a read from the **LSTAx** register provided there are no subsequent errors in the receiver FIFO.

TEMT: **Transmitter Empty**. In the non-FIFO mode, this bit is set to a logic 1 whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty. In the FIFO mode, this bit

is set to a logic 1 whenever the transmitter FIFO and the Transmitter Shift Register are both empty. In both cases, this bit is cleared when a byte is written to the transmitter data channel.

THRE: **Transmitter Holding Register Empty**. In the non-FIFO mode, this bit is set to a logic 1 whenever a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register, and it is reset to logic 0 concurrently with the write to the Transmitter Holding Register. In the FIFO mode, this bit is set to a logic 1 whenever the transmitter FIFO is empty, and it is cleared when at least one byte is written to the transmitter FIFO. In both cases, this bit when set also causes the UART to issue an interrupt when the Transmitter Holding Register Empty Interrupt enable (**ETBEI**) is set high.

BI: **Break Interrupt**. In the non-FIFO mode, this bit is set to a logic 1 whenever the SINx line is held in the logic 0 state for more than a full word transmission time (start bit + data bits + parity + stop bits). It is cleared by a read from the **LSTAx** register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to and is revealed when its associated character is at the top of the FIFO. When a break occurs, only one zero character is loaded into the FIFO. The next character transfer is enabled after the SIN line goes to the marking state and receives the next valid start bit.

FE: **Framing Error**. In the non-FIFO mode, this bit is set to a logic 1 whenever the stop bit following the last data bit or parity bit is detected as a logic 0 bit. It is cleared by a read from the **LSTAx** register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to and is revealed when its associated character is at the top of the FIFO. The UART will try to re-synchronize after a framing error by assuming that the framing error was due to the next start bit, so it samples this start bit twice and then takes in the data.

PE: **Parity Error**. In the non-FIFO mode, this bit is set to a logic 1 upon detection of a parity error if the **PEN** bit in the **LCTLx** Register is set. It is cleared by a read from the **LSTAx** register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to and is revealed when its associated character is at the top of the FIFO.

OE: **Overrun Error**. In the non-FIFO mode, this bit is set to a logic 1 upon detection of an overrun condition in the receiver. This means that the data in the Receiver Buffer Register was not read before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. It is cleared by a read from the **LSTAx** register. In the FIFO mode, an overrun error will occur only after the receiver FIFO is full and the next character has been completely received in the Receiver Shift Register. The **OE** bit is set as soon as this happens. The character in the Receiver Shift Register is then overwritten, but it is not transferred to the receiver FIFO.

DR: **Data Ready**. This bit is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the receiver FIFO. It is reset to a logic 0 by reading the Receiver Buffer Register or by reading all of the data in the receiver FIFO.

The MODEM Status Register (**MSTAx**) provides the current state of the MODEM control lines. In addition, four bits of the **MSTAx** Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state and are reset to a logic 0 whenever the **MSTAx** is read. The fields within the **MSTAx** register are defined as follows:

DCD: **Data Carrier Detect**. When **LOOP** = 0, this bit is the complement of the Data Carrier Detect (DCDx_) input. When **LOOP** = 1, this bit is equivalent to **OUT2** in the **MCTLx** Register.

RI: **Ring Indicator**. When **LOOP** = 0, this bit is the complement of the Ring Indicator (RIx_) input. When **LOOP** = 1, this bit is equivalent to **OUT1** in the **MCTLx** Register.

DSR: **Data Set Ready**. When **LOOP** = 0, this bit is the complement of the Data Set Ready (DSRx_) input. When **LOOP** = 1, this bit is equivalent to **DTR** in the **MCTLx** Register.

CTS: **Clear To Send**. When **LOOP** = 0, this bit is the complement of the Clear To Send (CTSx_) input. When **LOOP** = 1, this bit is equivalent to **RTS** in the **MCTLx** Register.

DDCD: **Delta Data Carrier Detect**. This bit is set to logic 1 if the DCDx_ input to the UART has changed state since the last time it was read.

TERI: **Trailing Edge Ring Indicator**. This bit is set to logic 1 if the RIx_ input to the UART has changed from logic 0 to logic 1 state since the last time it was read.

DDSR: **Delta Data Set Ready**. This bit is set to logic 1 if the DSRx_ input to the UART has changed state since the last time it was read.

DCTS: **Delta Clear To Send**. This bit is set to logic 1 if the CTSx_ input to the UART has changed state since the last time it was read.

The Scratch Register (**SCRTx**) is an 8-bit general-purpose read/write register for programmer to use as a temporary storage space. It has no defined purpose in the UART.

UART General Registers

Offset	*XCSR* + $0D0																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	UCTL																																UART
	U0GRTS	U1GRTS	U0TXP	U1TXP			XTAl64	UCOS																	UPS								
Operation	R	R	R/W	R/W	R	R	R	R/W	R								R								R/W								
Reset	0	0	0	0	0	0	0	0	$00								$00								$1B								

The UART Control Register (**UCTL**) contains the Hardware Flow Control bits, the state of the prescaled UART crystal clock, and the UART clock selection. The fields within the **UCTL** register are defined as follows:

U0GRT: UART 0 Gated Request To Send. This bit, when high, indicates RTS0_ is active (low) and the UART 0 receiver FIFO contains less than 14 characters. Used as Hardware Flow Control Request To Send signal.

U1GRTS: **UART 1 Gated Request To Send**. This bit, when high, indicates RTS1_ is active (low) and the UART 1 receiver FIFO contains less than 14 characters. Used as Hardware Flow Control Request To Send signal.

U0TXP: **UART 0 Transmitter Pause**. When set, the UART 0 will hold subsequent transmissions after the current character is completely transmitted. Used as Hardware Flow Control signal from receiving device.

U1TXP: **UART 1 Transmitter Pause**. When set, the UART 1 will hold subsequent transmissions after the current character is completely transmitted. Used as Hardware Flow Control signal from receiving device.

XTAL64: Prescaled UART Crystal Oscillator Output. This bit represents the logic state of the external/internal UART crystal clock divided by 64.

UCOS: **UART Clock Select**. This bit is used to select the clock source for both UARTs as shown in the following table .

Table 3-42. UART Clock Selection

XAD[29]	UCOS	UART Clock
0	0 (default)	external
0	1	internal
1	1 (always)	internal

The UART Clock Prescaler Register (**UPS**) is used to specify a frequency of the UART internal clock. The formula for calculating a value is shown below:

UART INTERNAL CLOCK = SYSTEM CLOCK / **UPS**/ 2

After power-up, **UPS** is initialized to $1B which produces a 1.8519 MHz (approximate) UART clock signal based on a 100.0 MHz system clock. Due to the IBM AC requirements on the UART oscillator clock input, the value written to **UPS** *must* be greater than $02. The following table provides the decimal divisors (**IEDHx** & **RTDLx** when **DLAB** = 1) to use with clock frequencies of 1.8432 MHz and 1.8519 MHz.

Table 3-43. Baud Rates and Divisors

Baud Rate	Decimal Divisor	Percent Error	
		1.8432 MHz	1.8519 MHz
50	2304	0	0.469
75	1536	0	0.469
110	1047	0.026	0.496
134.5	857	0.058	0.411
150	768	0	0.469
300	384	0	0.469
600	192	0	0.469
1200	96	0	0.469
1800	64	0	0.469
2000	58	0.69	0.224

Table 3-43. Baud Rates and Divisors (Continued)

Baud Rate	Decimal Divisor	Percent Error	
		1.8432 MHz	1.8519 MHz
2400	48	0	0.469
3600	32	0	0.469
4800	24	0	0.469
7200	16	0	0.469
9600	12	0	0.469
19200	6	0	0.469
38400	3	0	0.469
56000	2	2.86	3.34
128000	(No suitable divisor)	-	-
256000	(No suitable divisor)	-	-

Xport

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

For a functional description refer to the earlier section on Xport in this manual.

Xport Address Range (0, 1, 2, 3) Registers

Offset	**XPAR0: *XCSR* + $150** **XPAR1: *XCSR* + $158** **XPAR2: *XCSR* + $160** **XPAR3: *XCSR* + $168**																																
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Function**
Name	XPAR																																**Xport**
	STA																END																
Operation	R/W																R/W																
Reset	$0000																$0000																

The Xport Address Range Registers (**XPAR0, XPAR1, XPAR2 and XPAR3**) define the PowerPC address ranges to which each of Xport channels 0, 1, 2, and 3 will respond. The fields within the **XPARx** registers are defined as follows:

STA: **Starting Address**. This field defines the beginning of the address range. The Harrier compares STA to the 16 upper PowerPC address signals.

END: Ending Address. This field defines the ending of the address range. The Harrier compares END to the 16 upper PowerPC address signals.

Note that each channel's Address Range Register should be programmed so that its Xport Bus devices appear at an integer multiple of the size of the largest device on the channel. This mapping is required because the Harrier performs no address translation between the PowerPC and Xport buses.

Xport Attributes (0, 1, 2, 3) Registers

Offset	XPAT0: *XCSR* + $154 XPAT1: *XCSR* + $15C XPAT2: *XCSR* + $164 XPAT3: *XCSR* + $16C																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Function**
Name	XPAT																																**Xport**
	REN	WEN	BAM		RVEN		DW		AD				BLE							BREN		BRD						BWEN		BWD		
Operation	R/W	R/W	R/W	R	R/W	R	R/W	R/W	R/W	R	R	R/W	R/W	R	R	R	R/W	R	R/W	R	R	R	R	R	R/W	R	R/W					
Reset	0	0	1-P	0	V-P	0	V-P	V-P	$F	0	0	0	0	0	0	0	0	0	0	0	0	$0	0	0	0	0	0	0	0	0	0	$0	

The Xport Attributes Registers (**XPAT0, XPAT1, XPAT2, and XPAT3**) define the attributes for each of Xport channels 0, 1, 2, and 3. The fields within the **XPATx** registers are defined as follows:

REN: **Read Enable**. When set, the channel responds to PowerPC reads. When cleared, it does not respond unless the access is to the $FFF00000-$FFFFFFFF address range and RVEN0, RVEN1, RVEN2 or RVEN3 is set..

WEN: **Write Enable**. When set, the channel responds to PowerPC writes. When cleared, it does not respond unless the access is to the $FFF00000-$FFFFFFFF address range and RVEN0, RVEN1, RVEN2 or RVEN3 is set.

BAM: **Basic Mode**. When set, the channel uses basic conservative timing as shown in the section titled "Xport Bus Transaction Examples".

RVEN: **Reset Vector Enable**. Together with RVEN from the other three channels, RVEN determines which if any of Xport channels 0, 1, 2, or 3 is the source for reset vector fetches or for any other accesses in the range $FFF00000-$FFFFFFFF. The following table shows the encoding. Note that RVEN enables $FFF00000-$FFFFFFFF accesses regardless of the state of the REN and WEN control bits. RVEN0,

RVEN1, RVEN2, and RVEN3 initialize at power-up reset to match the values on 4 certain signal pins. Refer to the section titled *Hardware Configuration* on page 2-133.

Table 3-44. XPATx RVENx Encoding

RVEN3	RVEN2	RVEN1	RVEN0	Source of Reset Vectors
0	0	0	0	None of Xport Channels 0-3
X	X	X	1	Xport Channel 0
X	X	1	0	Xport Channel 1
X	1	0	0	Xport Channel 2
1	0	0	0	Xport Channel 3

DW: **Data Width**. This field indicates the Xport Bus's data width. The following table shows the encoding. Note that the DW bits initialize at power-up reset to match the values on two certain input signal pins. Refer to the section titled *Hardware Configuration* on page 2-133 for more details..

Table 3-45. XPATx DW Encoding

DW	Data Width
00	8 bits
01	16 bits
10	32 bits
11	16 bits (Hawk compatibility mode)

AD: **Access Delay**. This field determines the number of CLK periods the Harrier will add to the Xport Bus access time. The following table shows the encoding.

Table 3-46. XPATx AD Encoding

AD	Added Delay	Device Access Time
0000	0 CLK's	1ons
0001	1 CLK's	20ns
...	...	...
1110	14 CLK's	150ns
1111	15 CLK's	160ns

BLE: **Burst Length**. This field determines the maximum number of bytes Harrier will transfer per burst during burst reads or writes to the Xport Bus. Note that burst length refers to the number of bytes transferred per burst not to the number of data beats per burst. The following table shows the encoding.

Table 3-47. XPATx BLE Encoding

BLE	Bytes (not beats) per Burst
00	4
01	8
10	16
11	32

BREN: **Burst Read Enable**. When set, the Harrier performs burst reads for the corresponding channel if appropriate. When cleared, it does not.

BRD: **Burst Read Delay**. This field determines the number of CLK periods the Harrier will add for burst reads to the Xport Bus. The following table shows the encoding.

Table 3-48. XPATx BRD Encoding

BRD	Added Delay	Device Burst (Page) Read Access Time
000	0 CLK's	1 CLK (10ns @ 100 MHz)
001	1 CLK's	2 CLK's (20ns @ 100 MHz)
010	2 CLK's	3 CLK's (30ns @ 100 MHz)
011	3 CLK's	4 CLK's (40ns @ 100 MHz)
100	4 CLK's	5 CLK's (50ns @ 100 MHz)
101	5 CLK's	6 CLK's (60ns @ 100 MHz)
110	6 CLK's	7 CLK's(70ns @ 100 MHz)
111	7 CLK's	8 CLK's (80ns @ 100 MHz)

BWEN: **Burst Write Enable**. When set, the Harrier performs burst writes for the corresponding channel if appropriate. When cleared, it does not.

BWD: Burst Write Delay. This field determines the number of CLK periods the Harrier will add for burst writes to the Xport Bus. The following table shows the encoding.

Table 3-49. XPATx BWD Encoding

BWD	Added Delay
000	0 CLK's
001	1 CLK's
010	2 CLK's
011	3 CLK's
100	4 CLK's
101	5 CLK's
110	6 CLK's
111	7 CLK's

Xport General Control Register

Offset	*XCSR* + $170																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	XPGC																																Xport
								HDM																								
Operation	R	R	R	R	R	R	R	R/W	R								R															
Reset	0	0	0	0	0	0	0	V-P	$00								$00								$00							

The Xport General Control Register (**XPGC**) defines the attributes that apply globally, i.e. to all of Xport channels 0, 1, 2, and 3. The fields within the **XPGC** register are defined as follows:

HDM: **Hawk Data Compatibility Mode**. When set, any channel that is configured for the Hawk compatibility mode also uses Hawk 16-bit data ordering. When cleared, no channel uses Hawk 16-bit data ordering. Note that HDM initializes at power-up reset to match the value on a certain signal pin at power-up reset. Refer to the previous section titled "Hardware Configuration" for more information.

Arbiters

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

PCI Arbiter Register

Offset	*XCSR* + $090																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	PARB																															
	PRI		PRK				HIE			POL						ENA																
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R								R						
Reset	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	V-P	$00								$00						

Function
Arbiters

The PCI Arbiter Register (**PARB**) provides control and status for the PCI Arbiter. Please refer to the previous section titled "PCI Arbiter" for more information. The fields within the **PARB** register are defined as follows:

PRI: Priority. This field is used by the PCI Arbiter to establish a particular bus priority scheme. The encoding of this field is show in the following table.

Table 3-50. PARB PRI Encoding

PRI	Priority Scheme
00	Fixed
01	Round Robin
10	Mixed
11	Reserved

PRK: Parking. This field is used by the PCI Arbiter to establish a particular bus parking scheme. The encoding of this field is shown in the following table.

Table 3-51. PARB PRK Encoding

PRK	Parking Scheme
0000	Park on last master
0001	Park always on PARB6
0010	Park always on PARB5
0011	Park always on PARB4
0100	Park always on PARB3
0101	Park always on PARB2
0110	Park always on PARB1
0111	Park always on PARB0
1000	Park always on HARR
1111	None

HIE: Hierarchy. This field is used by the PCI Arbiter to establish a particular priority ordering when using a fixed or mixed mode priority scheme.

When using the fixed priority scheme, the encoding of this field is show in the following table.

Table 3-52. PARB HIE (Fixed Mode) Encoding

HIE	Priority ordering, highest to lowest
000	PARB6 -> PARB5 -> PARB4 -> PARB3 -> PARB2 -> PARB1 -> PARB0 -> HARR
001	HARR -> PARB6 -> PARB5 -> PARB4 -> PARB3 -> PARB2 -> PARB1 -> PARB0
010	PARB0 -> HARR -> PARB6 -> PARB5 -> PARB4 -> PARB3 -> PARB2 -> PARB1
011	PARB1 -> PARB0 -> HARR -> PARB6 -> PARB5 -> PARB4 -> PARB3 -> PARB2
100	PARB2 -> PARB1 -> PARB0 -> HARR -> PARB6 -> PARB5 -> PARB4 -> PARB3
101	PARB3 -> PARB2 -> PARB1 -> PARB0 -> HARR -> PARB6 -> PARB5 -> PARB4
110	PARB4 -> PARB3 -> PARB2 -> PARB1 -> PARB0 -> HARR -> PARB6 -> PARB5
111	PARB5 -> PARB4 -> PARB3 -> PARB2 -> PARB1 -> PARB0 -> HARR -> PARB6

When using the mixed priority scheme, the encoding of this field is show in the following table.

Table 3-53. PARB HIE (Mixed Mode) Encoding

HIE	Priority ordering, highest to lowest
000	Group 1 -> Group 2 -> Group 3 -> Group 4
001	Group 4 -> Group 1 -> Group 2 -» Group 3
010	Group 3 -> Group 4 -> Group 1 -> Group 2
011	Group 2 -> Group 3 -> Group 4 -> Group 1
100	Reserved
101	Reserved
110	Reserved
111	Reserved

POL: Park on Lock. If set, the PCI Arbiter will park the bus on the master who successfully obtains a PCI bus lock. The PCI Arbiter will keep the locking master parked and will not allow any non-locked masters to obtain access of the PCI bus until the locking master releases the lock. If cleared, the PCI Arbiter does not distinguish between locked and non-locked cycles.

ENA: Enable. This read only bit indicates the enabled state of the PCI Arbiter. If set, the PCI Arbiter is enabled and is acting as the system arbiter. If cleared, the PCI Arbiter is disabled and external logic is implementing the system arbiter. Please refer to the previous section titled "Hardware Configuration" for more information on how this bit is set.

PowerPC Arbiter Register

Offset	*XCSR* + $094																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	XARB																														
	FBR		FSR		FBW		FSW		TBS		PRK					ENA															
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R	R	R	R	R								R						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	V-P	$00								$00						

Function
Arbiters

The PowerPC Arbiter Register (**XARB**) provides control and status for the PowerPC Arbiter. Please refer to the previous section titled "PPC Arbiter" for more information. The fields within the **XARB** register are defined as follows:

FBR: Flatten Burst Read. This field is used by the PowerPC Arbiter to control how bus pipelining will be affected after all burst read cycles. The encoding of this field is shown in the following table.

FSR: Flatten Single Read. This field is used by the PowerPC Arbiter to control how bus pipelining will be affected after all single beat read cycles. The encoding of this field is shown in the following table.

FBW: Flatten Burst Write. This field is used by the PowerPC Arbiter to control how bus pipelining will be affected after all burst write cycles. The encoding of this field is shown in the following table.

FSW: Flatten Single Write. This field is used by the PowerPC Arbiter to control how bus pipelining will be affected after all single beat write cycles. The encoding of this field is shown in the following table.

Table 3-54. XARB FBR/FSR/FBW/FSW Encoding

FBR/FSR/FBW/FSW	Effects on Bus Pipelining
00	None
01	None
10	Flatten always
11	Flatten if switching masters

TBS: Three Bridge System. If set, the PowerPC Arbiter will enable the latching request protocol on the CPU1 request/grant pair. This protocol is normally reserved for bridge devices, and is not intended to be used with processor devices. If cleared, the latching request protocol will not be enabled for the CPU1 request/grant pair.

PRK: Parking. This field determines how the PowerPC Arbiter will implement CPU parking. The encoding of this field is shown in the following table.

Table 3-55. XARB PRK Encoding

PRK	CPU Parking
00	None
01	Park on last CPU
10	Park always on CPU0
11	Park always on CPU1

ENA: Enable. This read only bit indicates the enabled state of the PowerPC Arbiter. If set, the PowerPC Arbiter is enabled and is acting as the system arbiter. If cleared, the PowerPC Arbiter is disabled and external logic is implementing the system arbiter. Please refer to the section titled *Hardware Configuration* on page 2-133 for more information on how this bit gets set.

Watchdog Timers

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

Watchdog Timer Control Registers

Offset	WT0C: *XCSR* + $080 WT1C: *XCSR* + $088																																
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	WTxC																																Watchdog Timers
	KEY								ENA	ARM			RES				RLD																
Operation	W								R/W	R	R	R	R/W				R/W																
Reset	$00								1	0	0	0	$7 or $8				$FFFF																

The Watchdog Timer Control Registers (**WT0C and WT1C**) are used to provide control information to the watchdog timer functions within the Harrier. The fields within **WTxC** registers are defined as follows:

KEY: Key. This field is used during the two step arming process of the Control register. This field is sensitive to the following data patterns:

PATTERN_1 = $55

PATTERN_2 = $AA

The Control register will be in the armed state if PATTERN_1 is written to the KEY field. The Control register will be changed if in the armed state and PATTERN_2 is written to the KEY field. An incorrect sequence of patterns will cause the Control register to be in the unarmed state.

A value of all zeros will always be returned within the KEY field during read cycles.

ENA: Enable. This field determines whether or not the WDT is enabled. If a one is written to this bit, the timer will be enabled. A zero written to this bit will disable the timer. The ENA bit may only be modified on the second step of a successful two step arming process.

ARM: Armed. This read-only bit indicates the armed state of the register. If this bit is a zero, the register is unarmed. If this bit is a one, the register is armed for a write.

RES: Resolution. This field determines the resolution of the timer. The RES field may only be modified on the second step of a successful two step arming process. The following table shows the different options associated with this bit.

Table 3-56. WTxC RES Encoding

RES	Timer Resolution	Approximate Max Time
0000	1 µs	64 msec
0001	2 µs	128 msec
0010	4 µs	256 msec
0011	8 µs	512 msec
0100	16 µs	1 sec
0101	32 µs	2 sec
0110	64 µs	4 sec
0111	128 µs	8 sec
1000	256 µs	16 sec
1001	512 µs	32 sec
1010	1024 µs	1 min
1011	2048 µs	2 min
1100	4096 µs	4 min
1101	8192 µs	8 min
1110	16,384 µs	16 min
1111	32,768 µs	32 min

RLD: Reload. This field is written with a value that will be used to reload the timer. The RLD field may only be modified on the second step of a successful two step arming process.

Watchdog Timer Status Registers

Offset	WT0S: *XCSR* + $084 WT1S: *XCSR* + $08C			
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	Function
	WTxS			Watchdog Timers
Name	CNT			
Operation	R	R	R	
Reset	$FFFF	$00	$00	

The Watchdog Timer Status Registers (**WT0S and WT1S**) are used to provide status information from the watchdog timer functions within the Harrier. The field within **WTxS** registers is defined as follows:

CNT: Count. This read-only field reflects the instantaneous counter value of the WTx.

Exceptions

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

Functional Exception Enable Register

Offset	*XCSR* + $040																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	FEEN																															
	DMA	MIDB	MIM0	MIM1	MIP	UA0	UA1	ABT																								
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function: Exceptions

The Functional Exception Enable Register (**FEEN**) provides an array of enable bits pertaining to the various Functional Exceptions that the Harrier can generate. The fields within the **FEEN** register are defined as follows:

DMA: DMA. If set, the DMA Controller exception is enabled. When the exception is enabled, the status bit (**FEST**.DMA) will indicate the state of the DMA Controller exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.DMA). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

MIDB: Message Passing Inbound Doorbell. If set, the Message Passing Inbound Doorbell exception is enabled. When the exception is enabled, the status bit (**FEST**.MIDB) will indicate the state of the Message Passing Inbound Doorbell exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.MIDB). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

MIM0: Message Passing Inbound Message Register 0. If set, the Message Passing Inbound Message 0 exception is enabled. When the Message Passing Inbound Message 0 exception is enabled, the status bit (**FEST**.MIM0) will indicate the state of the exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.MIM0). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

MIM1: Message Passing Inbound Message Register 1. If set, the Message Passing Inbound Message 1 exception is enabled. When the exception is enabled, the status bit (**FEST**.MIM1) will indicate the state of the Message Passing Inbound Message 1 exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.MIM1). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

MIP: Message Passing Inbound Post_list. If set, the Message Passing Inbound Post_list exception is enabled. When the exception is enabled, the status bit (**FEST**.MIP) will indicate the state of the Message Passing Inbound Post_list exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.MIP). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

UA0: UART 0. If set, the UART 0 exception is enabled. When the exception is enabled, the status bit (**FEST**.UA0) will indicate the state of the UART 0 exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.UA0). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

UA1: UART 1. If set, the UART 1 exception is enabled. When the exception is enabled, the status bit (**FEST**.UA1) will indicate the state of the UART 1 exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.UA1). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

ABT: Abort. If set, the Abort exception is enabled. When the exception is enabled, the status bit (**FEST**.ABT) will indicate the state of the Abort exception. The interrupt can be polled by setting the enable bit and setting the mask bit (**FEMA**.ABT). When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

Functional Exception Status Register

Offset	*XCSR* + $044																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	FEST																															
	DMA	MIDB	MIM0	MIM1	MIP	UA0	UA1	ABT																								
Operation	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function
Exceptions

The Functional Exception Status Register (**FEST**) is a read-only register that provides an array of status bits pertaining to the various Functional Exceptions that the Harrier can generate. The fields within the **FEST** register are defined as follows:

DMA: DMA. If set, the DMA Controller has completed a DMA transaction. If the **FEMA** register permits, a processor interrupt will be generated. The processor may clear this field and the associated interrupt by writing a one to the **FECL**.DMA bit.

MIDB: Message Passing Inbound Doorbell. If set, a doorbell bit within the ***PMEP*:MGID** register has been written by a PCI master. If the **FEMA** register permits, a processor interrupt will be generated. The processor may clear this field and the associated interrupt by writing a one to the applicable doorbell bits within the ***XCSR*.MGID** register.

MIM0: Message Passing Inbound Message Register 0. If set, an inbound message has been written to the ***PMEP*:MGIM0** register by a PCI master. If the **FEMA** register permits, a processor interrupt will be generated. The processor may clear this field and the associated interrupt by writing a one to the **FECL**.MIM0 bit.

MIM1: Message Passing Inbound Message Register 1. If set, an inbound message has been written to the ***PMEP*:MGIM1** register by a PCI master. If the **FEMA** register permits, a processor interrupt will be generated. The processor may clear this field and the associated interrupt by writing a one to the **FECL**.MIM1 bit.

MIP: Message Passing Inbound Post_list. This read-only field indicates a message has been written by a PCI master to the Inbound Post_list fifo. If the **FEMA** register permits, a processor interrupt will be generated. The processor may clear this field and the associated interrupt by continuously reading the Post_list fifo (i.e. look at the fifo contents and maintain the fifo pointers) until all entries have been read.

UA0: UART 0. This field will be set whenever UART 0 is requesting service. If the **FEMA** register permits, a processor interrupt will be generated.

UA1: UART 1. This field will be set whenever UART 1 is requesting service. If the **FEMA** register permits, a processor interrupt will be generated.

ABT: Abort. This field will be set whenever the Harrier's ABTSW_ pin has been asserted for a short period (at least 30 ms). If the **FEMA** register permits, a processor interrupt will be generated. The processor may clear this field and the associated interrupt by writing a one to the **FECL**.ABT.

Functional Exception Mask Register

Offset	*XCSR* + $048																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	FEMA																															
	DMA	MIDB	MIM0	MIM1	MIP	UA0	UA1	ABT																								
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function	Exceptions

The Functional Exception Mask Register (**FEMA**) provides an array of bits pertaining to the masking of Functional Exceptions. All mask bits will power up with the mask enabled. The fields within the **FEMA** register are defined as follows:

DMA: DMA. If cleared, an interrupt is generated whenever the **FEST**.DMA bit is set. If set, an interrupt is not generated.

MIDB: Message Passing Inbound Doorbell. If cleared, an interrupt is generated whenever the **FEST**.MIDB bit is set. If set, an interrupt is not generated.

MIM0: Message Passing Inbound Message Register 0. If cleared, an interrupt is generated whenever the **FEST**.MIM0 bit is set. If set, an interrupt is not generated.

MIM1: Message Passing Inbound Message Register 1. If cleared, an interrupt is generated whenever the **FEST**.MIM1 bit is set. If set, an interrupt is not generated.

MIP: Message Passing Inbound Post_list. If cleared, an interrupt is generated whenever the **FEST**.MIP bit is set. If set, an interrupt is not generated.

UA0: UART 0. If cleared, an interrupt is generated whenever the **FEST**.UA0 bit is set. If set, an interrupt is not generated.

UA1: UART 1. If cleared, an interrupt is generated whenever the **FEST**.UA1 bit is set. If set, an interrupt is not generated.

ABT: Abort. If cleared, an interrupt is generated whenever the **FEST**.ABT bit is set. If set, an interrupt is not generated.

Functional Exception Clear Register

Offset	*XCSR* + $04C																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	FECL																															
	DMA		MM0	MM1				ABT																								
Operation	R/C	R	R/C	R/C	R	R	R	R/C	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function: Exceptions

The Functional Exception Clear Register (**FECL**) provides a way to clear a previously asserted Functional Exception. Note that not all Functional Exceptions are represented within this register. Please refer to the previous section titled "Exceptions" for more information. The fields within the **FECL** register are defined as follows:

DMA: DMA. If the **FEST**.DMA status bit is set, writing a one to this field will clear the status bit and the associated interrupt.

MM0: Message Passing Message Register 0. If the **FEST**.MM0 status bit is set, writing a one to this field will clear the status bit and the associated interrupt.

MM1: Message Passing Message Register 1. If the **FEST**.MM1 status bit is set, writing a one to this field will clear the status bit and the associated interrupt.

ABT: Abort. If the **FEST**.ABT status bit is set, writing a one to this field will clear the status bit and the associated interrupt.

Error Exception Enable Register

Offset	*XCSR* + $050																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EEEN																															
	PMA	PTA	PAP	PDP	PDT	PSE	PPE	PMR	SSE	SSC	SMX	SMS					XBT	XAP	XDP	XDT												
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function	Exceptions

The Error Exception Enable Register (**EEEN**) provides an array of enable bits pertaining to the various Error Exceptions that the Harrier can generate. The fields within the **EEEN** register are defined as follows:

PMA: PCI Master Abort. If set, the PCI Master Abort exception is enabled. When the exception is enabled, the status bit (**EEST**.PMA) will indicate the state of the PCI Master Abort exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PTA: PCI Target Abort. If set, the PCI Target Abort exception is enabled. When the exception is enabled, the status bit (**EEST**.PTA) will indicate the state of the PCI Target Abort exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PAP: PCI Address Parity Error. If set, the PCI Address Parity Error exception is enabled. When the exception is enabled, the status bit (**EEST**.PAP) will indicate the state of the PCI Address Parity Error exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PDP: PCI Data Parity Error. If set, the PCI Data Parity Error exception is enabled. When the exception is enabled, the status bit (**EEST**.PDP) will indicate the state of the PCI Data Parity Error exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PDT: PCI Bus Delayed Transaction Time-out. If set, the PCI Delayed Transaction Time-out exception is enabled. When the exception is enabled, the status bit (**EEST**.PDT) will indicate the state of the PCI Delayed Transaction Time-out exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PSE: PCI SERR. If set, the PCI SERR exception is enabled. When the exception is enabled, the status bit (**EEST**.PSE) will indicate the state of the PCI SERR exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PPE: PCI PERR. If set, the PCI PERR exception is enabled. When the exception is enabled, the status bit (**EEST**.PPE) will indicate the state of the PCI PERR exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

PMR: PCI Master Retry. If set, the PCI Master Retry exception is enabled. When the exception is enabled, the status bit (**EEST**.PMR) will indicate the state of the PCI Master Retry

SSE: SDRAM Memory Controller Single Bit Error. If set, the SDRAM Memory Controller Single Bit Error exception is enabled. When the exception is enabled, the status bit (**EEST**.SSE) will indicate the state of the SDRAM Memory Controller Single Bit Error exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

SSC: SDRAM Memory Controller Single Bit Error Count Overflow. If set, the SDRAM Memory Controller Single Bit Count Overflow exception is enabled. When the exception is enabled, the status bit (**EEST**.SSC) will indicate the state of the SDRAM Memory Controller Single Bit Count Overflow exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

SMX: SDRAM Memory Controller Multi Bit Error on PowerPC Access. If set, the SDRAM Memory Controller Multi Bit Error on PowerPc Access exception is enabled. When the exception is enabled, the status bit (**EEST**.SMX) will indicate the state of the SDRAM

Memory Controller Multi Bit Error on PowerPC Access exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

SMS: SDRAM Memory Controller Multi Bit Error on Scrub. If set, the SDRAM Memory Controller Multi Bit Error on Scrub exception is enabled. When the exception is enabled, the status bit (**EEST**.SMS) will indicate the state of the SDRAM Memory Controller Multi Bit Error on Scrub exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

XBT: PowerPC Bus Time-out. If set, the PowerPC Bus Time-out exception is enabled. When the exception is enabled, the status bit (**EEST**.XBT) will indicate the state of the PowerPC Bus Time-out exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

XAP: PowerPC Bus Address Parity Error. If set, the PowerPC Bus Address Parity Error exception is enabled. When the exception is enabled, the status bit (**EEST**.XAP) will indicate the state of the PowerPC Address Parity Error exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

XDP: PowerPC Bus Data Parity Error. If set, the PowerPC Bus Data Parity Error exception is enabled. When the exception is enabled, the status bit (**EEST**.XDP) will indicate the state of the PowerPC Bus Data Parity Error exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

XDT: PowerPC Bus Delayed Transaction Time-out. If set, the PowerPC Delayed Transaction Time-out exception is enabled. When the exception is enabled, the status bit (**EEST**.XDT) will indicate the state of the PowerPC Delayed Transaction Time-out exception. When the enable bit is cleared, the exception is disabled and the status bit will always read zero.

Note: PowerPC Bus delayed transaction time-outs can occur during normal operation. The XDT bit should NOT be set.

Error Exception Status Register

Offset	*XCSR* + $054																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EEST																															
	PMA	PTA	PAP	PDP	PDT	PSE	PPE	PMR	SSE	SSC	SMX	SMS					XBT	XAP	XDP	XDT									POF	SSOE	SMOE	XOF
Operation	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function	Exceptions

The Error Exception Status Register (**EEST**) is a read-only register that provides an array of status bits pertaining to the various Error Exceptions that the Harrier can generate. The fields within the **EEST** register are defined as follows:

PMA: PCI Master Abort. This bit is set when the PowerPC to PCI Bridge or the DMA Controller performs a master abort when the **EEEN**.PMA is enabled and all other PCI error status bits are cleared. If the **EEINT**.PMA is set, a processor interrupt will be generated. If the **EEMCK0**.PMA is set a machine check to processor 0 will be generated and if the **EEMCK1**.PMA is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PMA bit.

PTA: PCI Target Abort. This bit is set when the PowerPC to PCI Bridge or the DMA Controller receives a target abort when the **EEEN**.PTA is enabled and all other PCI error status bits are cleared. If the **EEINT**.PTA is set, a processor interrupt will be generated. If the **EEMCK0**.PTA is set a machine check to processor 0 will be generated and if the **EEMCK1**.PTA is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PTA bit.

PAP: PCI Address Parity Error. This bit is set when an address parity error is detected by the Harrier on the PCI bus when the **EEEN**.PAP is enabled and all other PCI error status bits are cleared. If

the **EEINT**.PAP is set, a processor interrupt will be generated. If the **EEMCK0**.PAP is set a machine check to processor 0 will be generated and if the **EEMCK1**.PAP is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PAP bit.

PDP: PCI Data Parity Error. This bit is set when a data parity error is detected by the Harrier on the PCI bus when the **EEEN**.PDP is enabled and all other PCI error status bits are cleared. If the **EEINT**.PDP is set, a processor interrupt will be generated. If the **EEMCK0**.PDP is set a machine check to processor 0 will be generated and if the **EEMCK1**.PDP is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PDP bit.

PDT: PCI Bus Delayed Transaction Time-out. This bit is set when a delayed transaction time-out is detected on the PowerPC bus when the **EEEN**.PDT is enabled and all other PCI error status bits are cleared. If the **EEINT**.PDT is set, a processor interrupt will be generated. If the **EEMCK0**.PDT is set a machine check to processor 0 will be generated and if the **EEMCK1**.PDT is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PDT bit.

PSE: PCI SERR. This bit is set anytime the PCI bus SERR signal is asserted when the **EEEN**.PSE is enabled and all other PCI error status bits are cleared. If the **EEINT**.PSE is set, a processor interrupt will be generated. If the **EEMCK0**.PSE is set a machine check to processor 0 will be generated and if the **EEMCK1**.PSE is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PSE bit.

PPE: PCI PERR. This bit is set anytime the PCI bus PERR signal is asserted when the **EEEN**.PPE is enabled and all other PCI error status bits are cleared. If the **EEINT**.PPE is set, a processor interrupt will be generated. If the **EEMCK0**.PPE is set a machine check to processor 0 will be generated and if the **EEMCK1**.PPE is set a machine check to

processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PPE bit.

PMR: PCI Master Retry. This bit is set when the PowerPC to PCI Bridge or the DMA Controller as a PCI master has exceeded the maximum number of unsuccessful attempts to transfer data due to retries when the **EEEN**.PMR is enabled and all other PCI error status bits are cleared. If the **EEINT**.PMR is set, a processor interrupt will be generated. If the **EEMCK0**.PMR is set a machine check to processor 0 will be generated and if the **EEMCK1**.PMR is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.PMR bit.

SSE: SDRAM Memory Controller Single Bit Error. This bit is set when the SDRAM Memory Controller detects and corrects a single bit error when the **EEEN**.SSE is enabled. If the **EEINT**.SSE is set, a processor interrupt will be generated. If the **EEMCK0**.SSE is set a machine check to processor 0 will be generated and if the **EEMCK1**.SSE is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.SSE bit.

SSC: SDRAM Memory Controller Single Bit Error Count Overflow. This bit is set when the SDRAM Memory Controller detects a single bit error count overflow when the **EEEN**.SSC is enabled. If the **EEINT**.SSC is set, a processor interrupt will be generated. If the **EEMCK0**.SSC is set a machine check to processor 0 will be generated and if the **EEMCK1**.SSC is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.SSC bit.

SMX: SDRAM Memory Controller Multi Bit Error on PowerPC Access. This bit is set when the SDRAM Memory Controller detects a multi bit error on any PowerPC access when the **EEEN**.SMX is enabled. If the **EEINT**.SMX is set, a processor interrupt will be generated. If the **EEMCK0**.SMX is set a machine check to processor 0 will be generated and if the **EEMCK1**.SMX is set a machine check

to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.SMX bit.

SMS: SDRAM Memory Controller Multi Bit Error on Scrub. This bit is set when the SDRAM Memory Controller detects a multi bit error on a scrub when the **EEEN**.SMS is enabled. If the **EEINT**.SMS is set, a processor interrupt will be generated. If the **EEMCK0**.SMS is set a machine check to processor 0 will be generated and if the **EEMCK1**.SMS is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.SMS bit.

XBT: PowerPC Bus Time-out. This bit is set when the PowerPC Address Bus Timer times out if the **EEEN**.XBT is enabled and all other PowerPC error status bits are cleared. If the **EEINT**.XBT is set, a processor interrupt will be generated. If the **EEMCK0**.XBT is set a machine check to processor 0 will be generated and if the **EEMCK1**.XBT is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.XBT bit.

XAP: PowerPC Bus Address Parity Error. This bit is set when an address parity error is detected on the PowerPC bus if the **EEEN**.XAP is enabled and all other PowerPC error status bits are cleared. If the **EEINT**.XAP is set, a processor interrupt will be generated. If the **EEMCK0**.XAP is set a machine check to processor 0 will be generated and if the **EEMCK1**.XAP is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.XAP bit.

XDP: PowerPC Bus Data Parity Error. This bit is set when an data parity error is detected on the PowerPC bus if the **EEEN**.XDP is enabled and all other PowerPC error status bits are cleared. If the **EEINT**.XDP is set, a processor interrupt will be generated. If the **EEMCK0**.XDP is set a machine check to processor 0 will be generated and if the **EEMCK1**.XDP is set a machine check to

processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.XDP bit.

XDT: PowerPC Bus Delayed Transaction Time-out. This bit is set when a delayed transaction time-out is detected on the PowerPC bus if the **EEEN**.XDT is enabled and all other PowerPC error status bits are cleared. If the **EEINT**.XDT is set, a processor interrupt will be generated. If the **EEMCK0**.XDT is set a machine check to processor 0 will be generated and if the **EEMCK1**.XDT is set a machine check to processor 1 will be generated. This bit and the interrupt or machine check associated with it may be cleared by writing a one to the **EECL**.XDT bit.

POF: PCI Error Overflow. This bit is set when any PCI error is detected and any of the PCI error status bits is already set.

SSOF: SDRAM Memory Controller Single Bit Error Overflow. This bit is set when any SDRAM Memory Controller single bit error is detected and any of the SDRAM Memory Controller single bit error status bits is already set.

SMOF: SDRAM Memory Controller Multi Bit Error Overflow. This bit is set when any SDRAM Memory Controller multi bit error is detected and any of the SDRAM Memory Controller multi bit error status bits is already set.

XOF: PowerPC Error Overflow. This bit is set when any PowerPC error is detected and any of the PowerPC error status bits is already set.

Error Exception Clear Register

Offset	*XCSR* + $058																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EECL																															
	PMA	PTA	PAP	PDP	PDT	PSE	PPE	PMR	SSE	SSC	SMX	SMS					XBT	XAP	XDP	XDT									POF	SSOF	SMOF	XOF
Operation	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R	R	R	R	R/C	R/C	R/C	R/C	R	R	R	R	R	R	R	R	R/C	R/C	R/C	R/C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function	Exceptions

The Error Exception Clear Register (**EECL**) provides a way to clear a previously asserted Error Exception. The fields within the **EECL** register are defined as follows:

PMA: PCI Master Abort. If the **EEST**.PMA status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PTA: PCI Target Abort. If the **EEST**.PTA status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PAP: PCI Address Parity Error. If the **EEST**.PAP status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PDP: PCI Address Parity Error. If the **EEST**.PDP status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PDT: PCI Bus Delayed Transaction Time-out. If the **EEST**.PDT status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PSE: PCI SERR. If the **EEST**.PSE status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PMR: PCI Master Retry. If the **EEST**.PMR status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

PPE: PCI PERR. If the **EEST**.PPE status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

SSE: SDRAM Memory Controller Single Bit Error. If the **EEST**.SSE status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

SSC: SDRAM Memory Controller Single Bit Error Count Overflow. If the **EEST**.SSC status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

SMX: SDRAM Memory Controller Multi Bit Error on PowerPC Access. If the **EEST**.SMX status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

SMS: SDRAM Memory Controller Multi Bit Error on Scrub. If the **EEST**.SMS status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

XBT: PowerPC Bus Time-out. If the **EEST**.XBT status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

XAP: PowerPC Bus Address Parity Error. If the **EEST**.XAP status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

XDP: PowerPC Bus Data Parity Error. If the **EEST**.XDP status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

XDT: PowerPC Bus Delayed Transaction Time-out. If the **EEST**.XDT status bit is set, writing a one to this field will clear the status bit and the associated interrupt or machine check.

POF: PCI Error Overflow. If the **EEST**.POF bit is set, writing a one to this field will clear the **EEST**.POF bit and the associated interrupt or machine check.

SSOF: SDRAM Memory Controller Single Bit Error Overflow. If the **EEST**.SSOF bit is set, writing a one to this field will clear the **EEST**.SSOF bit and the associated interrupt or machine check.

SMOF: SDRAM Memory Controller Multi Bit Error Overflow. If the **EEST**.SMOF bit is set, writing a one to this field will clear the **EEST**.SMOF bit and the associated interrupt or machine check.

XOF: PowerPC Bus Error Overflow. If the **EEST**.XOF bit is set, writing a one to this field will clear the **EEST**.XOF bit and the associated interrupt or machine check.

Error Exception Interrupt Enable Register

Offset	***XCSR* + $05c**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EEINT																															
	PMA	PTA	PAP	PDP	PDT	PSE	PPE	PMR	SSE	SSC	SMX	SMS					XBT	XAP	XDP	XDT												
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function	**Exceptions**

The Error Exception Interrupt Enable Register (**EEINT**) provides an array of interrupt enable bits pertaining to the various Error Exceptions that the Harrier can generate. The fields within the **EEINT** register are defined as follows:

PMA: PCI Master Abort. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PMA bit is set. If cleared, an interrupt will not be generated.

PTA: PCI Target Abort. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PTA bit is set. If cleared, an interrupt will not be generated.

PAP: PCI Address Parity Error. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PAP bit is set. If cleared, an interrupt will not be generated.

PDP: PCI Data Parity Error. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PDP bit is set. If cleared, an interrupt will not be generated.

PDT: PCI Bus Delayed Transaction Time-out. If set, an interrupt will be generated through the Harrier MPIC whenever the **EEST**.PDT bit is set. If cleared, an interrupt will not be generated.

PSE: PCI SERR. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PSE bit is set. If cleared, an interrupt will not be generated.

PPE: PCI PERR. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PPE bit is set. If cleared, an interrupt will not be generated.

PMR: PCI Master Retry. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.PMR bit is set. If cleared, an interrupt will not be generated.

SSE: SDRAM Memory Controller Single Bit Error. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.SSE bit is set. If cleared, an interrupt will not be generated.

SSC: SDRAM Memory Controller Single Bit Error Count Overflow. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.SSC bit is set. If cleared, an interrupt will not be generated.

SMX: SDRAM Memory Controller Multi Bit Error on PowerPC Access. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.SMX bit is set. If cleared, an interrupt will not be generated.

SMS: SDRAM Memory Controller Multi Bit Error on Scrub. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.SMS bit is set. If cleared, an interrupt will not be generated.

XBT: PowerPC Bus Time-out. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.XBT bit is set. If cleared, an interrupt will not be generated.

XAP: PowerPC Bus Address Parity Error. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.XAP bit is set. If cleared, an interrupt will not be generated.

XDP: PowerPC Bus Data Parity Error. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.XDP bit is set. If cleared, an interrupt will not be generated.

XDT: PowerPC Bus Delayed Transaction Time-out. If set, an interrupt will be generated through the Harrier's MPIC whenever the **EEST**.XDT bit is set. If cleared, an interrupt will not be generated.

Note: PowerPC Bus delayed transaction time-outs can occur during normal operation. The XDT bit should NOT be set.

Error Exception Machine Check 0 Enable Register

Offset	*XCSR* + $060																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EEMCK0																															
	PMA	PTA	PAP	PDP	PDT	PSE	PPE	PMR	SSE	SSC	SMX	SMS					XBT	XAP	XDP	XDT												
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function	Exceptions

The Error Exception Machine Check 0 Enable Register (**EEMCK0**) provides an array of machine check 0 enable bits pertaining to the various Error Exceptions. The fields within the **EEMCK0** register are defined as follows:

PMA: PCI Master Abort. If set, a machine check to processor 0 will be generated whenever the **EEST**.PMA bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PTA: PCI Target Abort. If set, a machine check to processor 0 will be generated whenever the **EEST**.PTA bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PAP: PCI Address Parity Error. If set, a machine check to processor 0 will be generated whenever the **EEST**.PAP bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PDP: PCI Data Parity Error. If set, a machine check to processor 0 will be generated whenever the **EEST**.PDP bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PDT: PCI Bus Delayed Transaction Time-out. If set, a machine check to processor 0 will be generated whenever the **EEST**.PDT bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PSE: PCI SERR. If set, a machine check to processor 0 will be generated whenever the **EEST**.PSE bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PPE: PCI PERR. If set, a machine check to processor 0 will be generated whenever the **EEST**.PPE bit is asserted. If cleared, a machine check to processor 0 will not be generated.

PMR: PCI Master Retry. If set, a machine check to processor 0 will be generated whenever the **EEST**.PMR bit is asserted. If cleared, a machine check to processor 0 will not be generated.

SSE: SDRAM Memory Controller Single Bit Error. If set, a machine check to processor 0 will be generated whenever the **EEST**.SSE bit is asserted. If cleared, a machine check to processor 0 will not be generated.

SSC: SDRAM Memory Controller Single Bit Error Count Overflow. If set, a machine check to processor 0 will be generated whenever the **EEST**.SSE bit is asserted. If cleared, a machine check to processor 0 will not be generated.

SMX: SDRAM Memory Controller Multi Bit Error on PowerPC Access. If set, a machine check to processor 0 will be generated whenever the **EEST**.SMX bit is asserted. If cleared, a machine check to processor 0 will not be generated.

SMS: SDRAM Memory Controller Multi Bit Error on Scrub. If set, a machine check to processor 0 will be generated whenever the **EEST**.SMS bit is asserted. If cleared, a machine check to processor 0 will not be generated.

XBT: PowerPC Bus Time-out. If set, a machine check to processor 0 will be generated whenever the **EEST**.XBT bit is asserted. If cleared, a machine check to processor 0 will not be generated.

XAP: PowerPC Bus Address Parity Error. If set, a machine check to processor 0 will be generated whenever the **EEST**.XAP bit is asserted. If cleared, a machine check to processor 0 will not be generated.

XDP: PowerPC Bus Data Parity Error. If set, a machine check to processor 0 will be generated whenever the **EEST**.XDP bit is asserted. If cleared, a machine check to processor 0 will not be generated.

XDT: PowerPC Bus Delayed Transaction Time-out. If set, a machine check to processor 0 will be generated whenever the **EEST**.XDT bit is asserted. If cleared, a machine check to processor 0 will not be generated.

Note: PowerPC Bus delayed transaction time-outs can occur during normal operation. The XDT bit should NOT be set.

Error Exception Machine Check 1 Enable Register

Offset	*XCSR* + $064																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EEMCK1																															
	PMA	PTA	PAP	PDP	PDT	PSE	PPE	PMR	SSE	SSC	SMX	SMS					XBT	XAP	XDP	XDT												
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function
Exceptions

The Error Exception Machine Check 1 Enable Register (**EEMCK1**) provides an array of machine check 1 enable bits pertaining to the various error exceptions. The fields within the **EEMCK1** register are defined as follows:

PMA: PCI Master Abort. If set, a machine check will be generated to processor 1 whenever the **EEST**.PMA bit is asserted. If cleared, a machine check will not be generated to processor 1.

PTA: PCI Target Abort. If set, a machine check will be generated to processor 1 whenever the **EEST**.PTA bit is asserted. If cleared, a machine check will not be generated to processor 1.

PAP: PCI Address Parity Error. If set, a machine check will be generated to processor 1 whenever the **EEST**.PAP bit is asserted. If cleared, a machine check will not be generated to processor 1.

PDP: PCI Data Parity Error. If set, a machine check will be generated to processor 1 whenever the **EEST**.PDP bit is asserted. If cleared, a machine check will not be generated to processor 1.

PDT: PCI Bus Delayed Transaction Time-out. If set, a machine check will be generated to processor 1 whenever the **EEST**.PDT bit is asserted. If cleared, a machine check will not be generated to processor 1.

PSE: PCI SERR. If set, a machine check will be generated to processor 1 whenever the **EEST**.PSE bit is asserted. If cleared, a machine check will not be generated to processor 1.

PPE: PCI PERR. If set, a machine check will be generated to processor 1 whenever the **EEST**.PPE bit is asserted. If cleared, a machine check will not be generated to processor 1.

PMR: PCI Master Retry. If set, a machine check will be generated to processor 1 whenever the **EEST**.PMR bit is asserted. If cleared, a machine check will not be generated to processor 1.

SSE: SDRAM Memory Controller Single Bit Error. If set, a machine check will be generated to processor 1whenever the **EEST**.SSE bit is asserted. If cleared, a machine check will not be generated to processor 1.

SSC: SDRAM Memory Controller Single Bit Error Count Overflow. If set, a machine check will be generated to processor 1 whenever the **EEST**.SSC bit is asserted. If cleared, a machine check will not be generated to processor 1.

SMX: SDRAM Memory Controller Multi Bit Error on PowerPC Access. If set, a machine check will be generated to processor 1 whenever the **EEST**.SMX bit is asserted. If cleared, a machine check will not be generated to processor 1.

SMS: SDRAM Memory Controller Multi Bit Error on Scrub. If set, a machine check will be generated to processor 1 whenever the **EEST**.SMS bit is asserted. If cleared, a machine check will not be generated to processor 1.

XBT: PowerPC Bus Time-out. If set, a machine check will be generated to processor 1 whenever the **EEST**.XBT bit is asserted. If cleared, a machine check will not be generated to processor 1.

XAP: PowerPC Bus Address Parity Error. If set, a machine check will be generated to processor 1 whenever the **EEST**.XAP bit is asserted. If cleared, a machine check will not be generated to processor 1.

XDP: PowerPC Bus Data Parity Error. If set, a machine check will be generated to processor 1whenever the **EEST**.XDP bit is asserted. If cleared, a machine check will not be generated to processor 1.

XDT: PowerPC Bus Delayed Transaction Time-out. If set, a machine check will be generated to processor 1whenever the **EEST**.XDT bit is asserted. If cleared, a machine check will not be generated to processor 1.

Note: PowerPC Bus delayed transaction time-outs can occur during normal operation. The XDT bit should NOT be set.

Error Diagnostics

All of the registers for this function are located within PowerPC address space as a part of the *XCSR* Register Group.

Error Diagnostics Error Injection Register

Offset	*XCSR* + $06c																												
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	EDEI																																Error Diagnostics
	DPE0	DPE1	DPE2	DPE3	DPE4	DPE5	DPE6	DPE7					APE0	APE1	APE2	APE3			XDTT	TMRT1	TMRT2	TMRT3	PMRT	PDTT									
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0-P	0-P	0-P	0-P	0-P	0-P	$00								

The Error Diagnostics Error Injection Register (**EDEI**) provides a way to inject certain types of errors to test the Harrier error capture and status circuitry. The fields within the **EDEI** register are defined as follows:

DPEx: Data Parity Error Enable. These bits are used for test reasons to purposely inject data parity errors whenever the Harrier is sourcing PowerPC data. A data parity error will be created on the corresponding PowerPC data parity bus if a bit is set. For example, setting DPE0 will cause DP0 to be generated incorrectly. If the bit is cleared, the Harrier will generate correct data parity.

APEx: Address Parity Error Enable. These bits are used for test reasons to purposely inject address parity errors whenever the Harrier is acting as a PowerPC bus master. An address parity error will be created on the corresponding PowerPC address parity bus if a bit is set. For example, setting APE0 will cause AP0 to be generated incorrectly. If the bit is cleared, the Harrier will generate correct address parity.

XDTT: PPC Slave Retry Test Bit. This bit is used for test reasons to impose a short PPC delayed transaction time-out. A short PPC delayed transaction time-out will occur when this bit is set.

Note: This bit is not used in Harrier 2.

TMRT1: **Timer Test Bit 1.** This bit is used for testing purposes only. When set, the sampling rate of the switch debouncers will change from 1ms to 1µs.

TMRT2: **Timer Test Bit 2.** This bit is used for testing purposes only. When set, the RSTO_ pulse extender will change from 100µs to 1µs.

TMRT3: Timer Test Bit 3. This bit is used for testing purposes only. When set, the valid hold time for the push button RSTSW_ signal will be 3µs instead of 3s, and the RSTO_ signal will be generated upon the completion of that valid hold time.

PMRT: PCI Master Retry Test Bit. This bit is used for test reasons to reduce the amount of PCI master retries needed before generating the PCI Master Retry exception. Setting this bit puts the PCI master in test mode.

PDTT: PCI Delayed Transaction Test Bit. This bit is used for test reasons to impose a short PCI delayed transaction time-out. A short PCI delayed transaction time-out will occur when this bit is set.

Error Diagnostics PowerPC Address Register

Offset	***XCSR* + $070**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Function**
Name	EXAD																																**Error Diagnostics**
Operation	R																																
Reset	$00000000																																

The Error Diagnostics PowerPC Address Register (**EXAD**) captures addressing information from the PowerPC bus whenever an applicable Error Exception has been detected. The register contents will only be updated when there are no status bits set within the **EEST** register. During a string of successive errors, this register will retain information pertaining to the first error.

Error Diagnostics PowerPC Attribute Register

Offset	*XCSR* + $074																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EXAT																															
			MID		DMA	DDF	DDI	TBST	TSIZ0	TSIZ1	TSIZ2	TT0	TT1	TT2	TT3	TT4																
Operation	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Function
Error Diagnostics

The Error Diagnostics PowerPC Attribute Register (**EXAT**) captures attribute information from the PowerPC bus whenever an applicable Error Exception has been detected. The register contents will only be updated when there are no status bits set within the **EEST** register. During a string of successive errors, this register will retain information pertaining to the first error. The fields within the **EXAT** register are defined as follows:

MID: Master ID. This field contains the ID of the PowerPC master which originated the transfer in which the Error Exception occurred. The encoding scheme is identical to that used in the **GCSR** register.

DMA: DMA Error. This read only bit will be set to a “1” when the current transaction is DMA originated and will be set to a “0” when the current transaction is PowerPC to PCI bridge originated. Additionally this bit may only be set to a “1” if the error being captured is a PowerPC bus time-out and the MID field of the error indicates the Harrier is the bus master. If the error being captured is not a PowerPC bus time-out or the MID field of the error indicates the Harrier is not the bus master then this bit will read as a “0”.

DDF: DMA Descriptor Fetch. This read only bit will be set to a “1” when the DMA error occurred during descriptor fetch and will be set to a “0” when the DMA error occurred during data movement. Additionally this bit may only be set to a “1” if the error being captured is a PowerPC bus time-out and the MID field of the error indicates the Harrier is the bus master and the error being captured is DMA. If the

error being captured is not a PowerPC bus time-out or the MID field of the error indicates the Harrier is not the bus master or the error being captured is not DMA then this bit will read as a “0”.

DDI: DMA Direction. This read only bit will be set to a “1” when the DMA error occurred during fifo fill and will be set to a “0” when the DMA error occurred during fifo empty. Additionally this bit may only be set to a “1” if the error being captured is a PowerPC bus time-out and the MID field of the error indicates the Harrier is the bus master and the error being captured is DMA. If the error being captured is not a PowerPC bus time-out or the MID field of the error indicates the Harrier is not the bus master or the error being captured is not DMA then this bit will read as a “0”.

TBST: Transfer Burst. This bit is cleared when the transfer in which the Error Exception occurred was a burst transfer. This bit is set during single beat transfers.

TSIZx: Transfer Size. This field contains the transfer size of the PowerPC transfer in which the Error Exception occurred.

TTx: Transfer Type. This field contains the transfer type of the PowerPC transfer in which the Error Exception occurred.

Error Diagnostics PCI Address Register

Offset	***XCSR* + $078**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EPAD																															
Operation	R																															
Reset	$00000000																															

Function
Error Diagnostics

The Error Diagnostics PCI Address Register (**EPAD**) captures addressing information from the PCI bus whenever an applicable Error Exception has been detected. The register contents will only be updated when there are no status bits set within the **EEST** register. During a string of successive errors, this register will retain information pertaining to the first error.

Error Diagnostics PCI Attribute Register

Offset	*XCSR* + $07C																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	EPAT																																Error Diagnostics
			DMA	DDI	COMM3	COMM2	COMM1	COMM0																									
Operation	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The Error Diagnostics PCI Attribute Register (**EPAT**) captures attribute information from the PCI bus whenever an applicable Error Exception has been detected. The register contents will only be updated when there are no status bits set within the **EEST** register. During a string of successive errors, this register will retain information pertaining to the first error. The fields within the **EPAT** register are defined as follows:

DMA: DMA Error. This read only bit will be set to a “1” when the current transaction is DMA originated and will be set to a “0” when the current transaction is PowerPC to PCI bridge originated. Additionally this bit may only be set to a “1” if the error being captured is a PMA, PTA or PMR. Otherwise this bit will read as a “0”.

DDI: DMA Direction. This read only bit will be set to a “1” when the DMA error occurred during fifo fill and will be set to a “0” when the DMA error occurred during fifo empty. Additionally this bit may only be set to a “1” if the error being captured is a PMA, PTA or PMR and the error being captured is DMA. Otherwise this bit will read as a “0”.

COMMx: PCI Command. This field contains the PCI command of the PCI transfer in which the Error Exception occurred.

Miscellaneous Functions

All of the remaining miscellaneous registers described in this section are located within PowerPC address space as a part of the *XCSR* Register Group.

Vendor ID/Device ID Registers

Offset	***XCSR* + $000**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Function**
Name	VENI																DEVI																**ID**
Operation	R																R																
Reset	$1057																$480B																

The Vendor ID Register (**VENI**) identifies the manufacturer of the device. This identifier is allocated by the PCI SIG to ensure uniqueness. $1057 has been assigned to Motorola and is hardwired as a read-only value.

The Device ID Register (**DEVI**) uniquely identifies this particular device. The Harrier will always return $480B.

Revision ID Register

Offset	*XCSR* + $004																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name									REVI																								ID
Operation	R								R								R							R								
Reset	$00								$01								$00								$00								

The Revision ID Register (**REVI**) identifies the Harrier revision level.

Global Control and Status Register

Offset	*XCSR* + $010																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	GCSR																															
	PUR	SRST	ARST	PRST	PUST3	PUST2	PUST1	PUST0				AOAO	XBS		BTO				MID		RAT										
Operation	R/C	R	R	R	R				R	R	R	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	1-P	L_V	L-V	l-V	V-P				0	0	0	V-P	0	1	0	0	0	0	0	0	0	x	x	x	0	0	0	0	0	0	0	0

Function
Control and Status

The Global Control-Status Register (**GCSR**) provides miscellaneous control and status information for the Harrier. The fields within the **GCSR** are defined as follows:

PUR: Power-up Reset. This bit tells software when a power-up has occurred. Only power-up reset sets PUR. Only user software writing a one to PUR alone clears it.

SRST: Push Button Reset. This bit tells software that the push button reset signal was asserted when the Harrier received an RST_ signal. This bit is cleared by power-up reset.

ARST: Auxiliary Reset. This bit tells software that the auxiliary reset signal was asserted when the Harrier received an RST_ signal. This bit is cleared by power-up reset.

PRST: Program Reset. This bit tells software that the software reset out bit was asserted when the Harrier received an RST_ signal. This bit is cleared by power-up reset.

PUSTx: Power-up Status. PUSTx indicates the levels that were on four certain input signal pins during power-up reset. This field provides a means to pass information to software using pull-up/pull-down resistors on the four inputs. Refer to Table 2-24 on page 2-133 for more information.

AOAO: Address-Only Acknowledge Other. When cleared, the Harrier responds to PowerPC address-only transactions only if they address one of its own resources. When set, the Harrier also responds to address-only transactions to non-Harrier addresses, provided no other slave responds within 8 CLK periods. AOAO reflects the value on a certain input signal at power-up reset. Refer to Table 2-24 on page 2-133 for more information.

XBS: PowerPC Burst Size. This field specifies the latching request burst size that is used by all the Harrier PowerPC bus masters. When performing multiple burst transfers, a master will leave its request asserted for the number of transfers indicated within this field. Once the specified number of transfers has happened, the master will momentarily remove its request. If there are other masters requesting the PowerPC bus, then the PowerPC Arbiter will grant the bus to the next highest priority master. If there are no other requests pending, then the original transaction continues on until the next burst boundary.

The Harrier is tuned to work with a burst size of 128 bytes, however the actual average transfer size for each system varies and this field may be changed accordingly.

The burst size is encoded as shown in the following table.

Table 3-57. GCSR XBS Encoding

XBS	Burst Size
00	64 bytes
01	128 bytes
10	256 bytes
11	continuous

BTO: Bus Time-out. This field specifies the enabling and time-out length to be used by the PowerPC Address Bus Timer. The time-out length is encoded as shown in the following table.

Table 3-58. GCSR BTO Encoding

BTO	Time Out Length
00	256 µsec
01	64 µsec
10	8 µsec
11	disabled

MID: Master ID. This field is encoded as shown in the following table to indicate who is currently PowerPC bus master. This information is obtained by sampling the XARB0 thru XARB3 pins when in external PowerPC arbitration mode. When in internal PowerPC arbitration mode this information is generated by the PowerPC Arbiter. In a multiprocessor environment, these bits allow software to determine on which processor it is currently running.

Table 3-59. GCSR MID Encoding

MID	PowerPC Data Bus Master
00	CPU0
01	CPU1
10	EXTL
11	Harrier

RAT: Ratio. This is a read only field that is used to indicate the PowerPC to PCI clock ratio that has been established by the Harrier at the release of reset. The encoding of this field is show in the following table.

Table 3-60. GCSR RAT Encoding

RAT	PowerPC/PCI clock ratio
000	Undefined
001	1:1
010	2:1
011	3:1
100	3:2
101	Undefined
110	5:2
111	Undefined

PowerPC Clock Frequency Register

Offset	*XCSR* + $014																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	XCFR																																Control and Status
Operation	R/W								R								R							R								
Reset	$64								$00								$00								$00								

The PowerPC Clock Frequency Register (**XCFR**) should be programmed with the hexadecimal value of the operating clock frequency in MHz (e.g. $64 for 100 MHz). When these bits are programmed this way, the chip's prescale counter produces a 1 MHz (approximate) output. The output of the chip prescale counter is used by the refresher/scrubber and the 32-bit counter. After power-up, this register is initialized to $64 (for 100 MHz). The formula is:

Counter_Output_Frequency = (Clock Frequency)/**XCFR**

For example, if the Clock Frequency is 100 MHz and **XCFR** is $64, then the counter output frequency is 100 MHz/100 = 1 MHz.

When the CLK pin is operating slower than 100 MHz, software should program **XCFR** to be at least as slow as the CLK pin's frequency as soon as possible after power-up reset so that SDRAM refresh does not get behind. It is okay for the software then to take some time to "up" **XCFR** to the correct value. Refresh will get behind only when the actual CLK pin's frequency is lower than the value programmed into **XCFR**.

Count 32-bit Register

Offset	***XCSR* + $018**																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Function**
Name	CT32																																**Control and Status**
Operation	R/W																																
Reset	$00000000																																

The Count 32-bit Register (**CT32**) is a 32-bit, free-running counter that increments once per microsecond if the **XCFR** register has been programmed properly. **CT32** is cleared by power-up and local reset.

Note that when the system clock is a fractional frequency (e.g. 66.67 MHz), **CTR32** will count at a fractional amount faster or slower than 1 MHz, depending on the programming of the **XCFR** register.

Miscellaneous Control and Status Register

Offset	*XCSR* + $01c																														
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	MCSR																																Control and Status
	BRDFLS	BRDFL	ERDYS	EREADY	SCON	TBEN0	TBEN1	RSTOUT	ASWS							RSTINH																
Operation	R	R/W	R	R/W	R	R/W	R/W	S	R	R	R	R	R	R	R	R/W	R								R							
Reset	1	1	0	0	X	1	1	1	0	0	0	0	0	0	0	0	$00								$00							

The Miscellaneous Control-Status Register (**MCSR**) provides miscellaneous control and status information for the Harrier. The fields within the **MCSR** are defined as follows:

BRDFLS: Board Fail Status. This bit is the board fail status bit. It is high when the Harrier's board fail (BRDFL_) pin is asserted. The board fail pin is open drain and may be driven low by the Harrier or external logic.

BRDFL: Board Fail. This bit is the board fail bit. When the board fail bit is asserted, the Harrier will assert its board fail (BRDFL_) pin.

ERDYS: PCI Bus Enumeration Ready Status. This bit is the EREADY status bit. It is high when the Harrier EREADY pin is high and low when the Harrier EREADY pin is low. The EREADY pin is open drain and may be driven low by the Harrier or external logic.

EREADY: PCI Bus Enumeration Ready. This bit is the EREADY bit. When the EREADY bit is low, the Harrier will drive its EREADY pin low with an open drain driver. When the EREADY bit is high, the Harrier will not drive the EREADY pin.

SCON: System Controller. This bit is high when the Harrier SCON_ pin is asserted. When this bit is high, the Harrier is the system controller.

TBEN0: Time Base Enable 0. When set the Harrier will assert its TBEN0 pin.

TBEN1: Time Base Enable 1. When set the Harrier will assert its TBEN1 pin.

RSTOUT: RESET OUT. When a one is written to this bit, the Harrier will assert its reset out (RSTO_) pin.

ASWS: **Abort Switch Status.** This bit reflects the state of the ABTSW_ pin. This bit is a one when the Abort Switch is pressed.

RSTINH: Reset Inhibit. When this bit is set, the Harrier will inhibit asserting its reset out (RSTO_) pin when RSTSW_ pin is asserted.

General Purpose Registers

Offset	**GPRG0: *XCSR* + $020** **GPRG1: *XCSR* + $024** **GPRG2: *XCSR* + $028** **GPRG3: *XCSR* + $02C** **GPRG4: *XCSR* + $030** **GPRG5: *XCSR* + $034**	
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	**Function**
Name	GPRGx	**General Purpose Registers**
Operation	R/W	
Reset	GPRG0 -> GPRG3: $00000000 GPRG4 -> GPRG5: $00000000-P	

The General Purpose Registers (**GPRG0, GPRG1, GPRG2, GPRG3, GPRG4,** and **GPRG5**) are provided for inter-process message passing or general purpose storage. They do not control any hardware. Register groups **GPRG0**/**GPRG1**, **GPRG2**/**GPRG3**, and **GPRG4**/**GPRG5** and may be combined as single 64-bit registers.

GPRG0 thru **GPRG3** will always be reset to all zeros following a power up reset or a local reset. **GPRG4** and **GPRG5** are only affected by power up reset. A local reset will not change the contents of these registers, therefore these registers may be used for storing critical information when performing commanded reset cycles.

General Purpose Memory

Offset	*XCSR* + $400 -> $BFC																																
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Function
Name	GPM																																General Purpose Memory
Operation	R/W																																
Reset	$XXXXXXXX																																

The General Purpose Memory (**GPM**) is a 2Kilobyte RAM. It is provided for inter-process message passing or general purpose storage. It does not control any hardware. The **GPM** group is the only group within the *XCSR* that supports all transfer sizes supported by the PowerPC and all alignments within the 8 byte boundary.

On power up reset, the contents of the **GPM** are unknown. Software is required to initialize the **GPM**. Local reset does not change the contents of the **GPM**, therefore the **GPM** may be used for storing critical information when performing commanded reset cycles.

Performance 4

SDRAM Interface

The following table shows the performance summary for SDRAM when operating at 100 MHz using PC100 SDRAM with a CAS_latency of 2. The figure on the following page defines the times that are specified in the table.

Table 4-1. PowerPC (60x) Bus to SDRAM Estimated Access Timing at 100 MHz with PC100 SDRAM's

ACCESS TYPE	Access Time (tB1-tB2-tB3-tB4)	Comments
4-Beat Read after idle, SDRAM Internal-Bank Inactive	10-1-1-1	
4-Beat Read after idle, SDRAM Internal-Bank Active -Page Miss	12-1-1-1	
4-Beat Read after idle, SDRAM Internal-Bank Active -Page Hit	7-1-1-1	
4-Beat Read after 4-Beat Read, SDRAM Internal-Bank Active -Page Miss	5-1-1-1	
4-Beat Read after 4-Beat Read, SDRAM Internal-Bank Active -Page Hit	2.5-1-1-1	2.5-1-1-1 is an average of 2-1-1-1 half of the time and 3-1-1-1 the other half.
4-Beat Write after idle, SDRAM Internal-Bank Active or Inactive	4-1-1-1	
4-Beat Write after 4-Beat Write, SDRAM Internal-Bank Active -Page Miss	6-1-1-1	

Table 4-1. PowerPC (60x) Bus to SDRAM Estimated Access Timing at 100 MHz with PC100 SDRAM's (Continued)

ACCESS TYPE	Access Time (tB1-tB2-tB3-tB4)	Comments
4-Beat Write after 4-Beat Write, SDRAM Internal-Bank Active -Page Hit	3-1-1-1	3-1-1-1 for the second burst write after idle. 2-1-1-1 for subsequent burst writes.
1-Beat Read after idle, SDRAM Internal-Bank Inactive	10	
1-Beat Read after idle, SDRAM Internal-Bank Active -Page Miss	12	
1-Beat Read after idle, SDRAM Internal-Bank Active -Page Hit	7	
1-Beat Read after 1-Beat Read, SDRAM Internal-Bank Active -Page Miss	8	
1-Beat Read after 1-Beat Read, SDRAM Internal-Bank Active -Page Hit	5	
1-Beat Write after idle, SDRAM Internal-Bank Active or Inactive	5	
1-Beat Write after 1-Beat Write, SDRAM Internal-Bank Active -Page Miss	13	
1-Beat Write after 1-Beat Write, SDRAM Internal-Bank Active -Page Hit	8	

Notes

1. SDRAM speed attributes are programmed for the following: CAS_latency = 2, tRCD = 2 CLK Periods, tRP = 2 CLK Periods, tRAS = 5 CLK Periods, tRC = 7 CLK Periods, tDPL = 2 CLK

Periods, and the WDPL bit is set in the Synchronous DRAM Control register.

2. The Harrier is configured for “no external registers” on the SDRAM control signals. Refer to the section titled *Hardware Configuration* on page 2-133 for more information.

3. tB1, tB2, tB3, and tB4 are specified in the following figure.

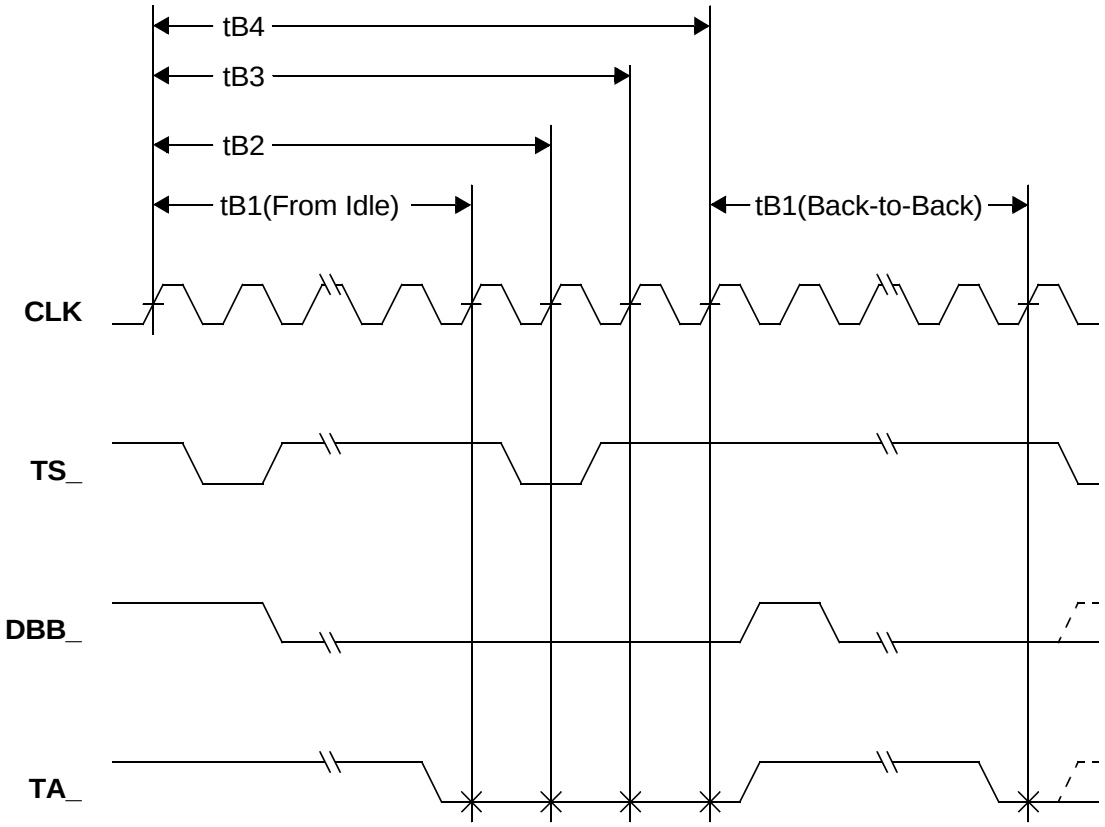


Figure 4-1. Timing Definitions for Table 4-1

Notes When the bus starts out idle, tB1 is the number of CLK periods from the rising edge of the CLK that drives TS_low, to the rising edge of CLK that samples the first TA_low.

When the bus is busy and TS_ is being asserted as soon as possible after Harrier asserts aack_ the back-to-back condition occurs. For back-to-back cycles, tB1 is the number of CLK periods from the rising edge of CLK that samples the last TA_ low of a data tenure to the rising edge of CLK that samples the first TA_ low of the next data tenure.

tB2 is the number of CLK periods from the rising edge of the CLK that samples the first TA_ low in a burst data tenure to the rising edge of CLK that samples the second TA_ low in that data tenure.

tB3 is the number of CLK periods from the rising edge of CLK that samples the second TA_ low in a burst data tenure to the rising edge of CLK that samples the third TA_ low in that data tenure.

tB4 is the number of CLK periods from the rising edge of CLK that samples the third TA_ low in a burst data tenure to the rising edge of CLK that samples the last TA_ low in that data tenure.

Xport Bus Interface

The following subsections describe the latency periods of Xport bound reads and provides background information.

Latency of Xport-bound Reads (Xport read bursting disabled)

Table 4-2. PowerPC (60x) Bus Performance for Xport Bus Bound Cycles

60x Transfer Size	Xport Channel Width (In bits)	Number of data beatson Xport Bus	Number of CLK periods required for 60x data Beat 1 when Device access time is:							
			50ns	60ns	70ns	80ns	90ns	100ns	120ns	150ns
1 byte	8,16, or 32	1	19	20	21	22	23	24	26	29
2 bytes	16 or 32									
3 bytes	32									
4 bytes	32									
2 bytes	8	2	25	27	29	31	33	35	39	45
3 bytes	16									
4 bytes	16									
5 bytes	32									
6 bytes	32									
8 bytes	32									
7 bytes	32									
3 bytes	8	3	31	34	37	40	43	46	52	61
5 bytes	16									
6 bytes	16									
4 bytes	8	4	37	41	45	49	53	57	65	77
7 bytes	16									
8 bytes	16									
5 bytes	8	5	43	48	53	58	63	68	78	93
6 bytes	8	6	49	55	61	67	73	79	91	109
7 bytes	8	7	55	62	69	76	83	90	104	125
8 bytes	8	8	61	69	77	85	93	101	117	141

Table 4-2. PowerPC (60x) Bus Performance for Xport Bus Bound Cycles

60x Transfer Size	Xport Channel Width (In bits)	Number of data beats on Xport Bus	Number of CLK periods required for 60x data Beat 1 when Device access time is:							
			50ns	60ns	70ns	80ns	90ns	100ns	120ns	150ns
32 bytes	32									
32 bytes	16	16	109	125	141	157	173	189	221	269
32 bytes	8	32	205	237	269	301	333	365	429	525

Background Information

CLK frequency is 100 MHz. For 32-byte transfers, data beats 2,3,4 require 1,1,1 CLKs. The numbers for data beat 1 come from the formula:

tDB1 = ***OVH*** + ((***DAT*** +1) x ***NXB***)

Where:

tDB1 is the number of CLK periods from TS_ to the first TA_.

OVH is the overhead (expressed in CLK periods) associated with any read cycle. ***OVH*** is 13 if the PowerPC master comes back to get the data just as it becomes ready. ***OVH*** increases by one for each CLK period that the PowerPC master is late.

DAT is the device access time (expressed in CLK periods). And,

NXB is the number of data beats required on the Xport bus for the transaction.

The following table shows ***NXB*** for the different port widths and PowerPC transfer sizes. Note that the table is for accesses that are aligned to the port size. Some accesses that are not aligned to port size add 1 beat to the number shown in the table

Table 4-3. Number of Xport Data Beats for Different PowerPC (60*x*) Transfer Sizes

60x Transfer Size	8-Bit Xport Channel Width	16-Bit Xport Channel Width	32-Bit Xport Channel Width
1 Byte	1 Beat	1 Beat	1 Beat
2 Bytes	2 Beats	1 Beat	1 Beat
3 Bytes	3 Beats	2 Beats	1 Beat
4 Bytes	4 Beats	2 Beats	1 Beat
5 Bytes	5 Beats	3 Beats	2 Beats
6 Bytes	6 Beats	3 Beats	2 Beats
7 Bytes	7 Beats	4 Beats	2 Beats
8 Bytes	8 Beats	4 Beats	2 Beats
32 Bytes	32 Beats	16 Beats	8 Beats

PowerPC to PCI Bridge

This section describes the differences between 32-bit and 64-bit outbound and inbound performance.

Table 4-4. Outbound Performance Matrix

Transaction	32-bit PCI			64-bit PCI			Clock Ratio
	Latency (PPC Clocks)		Continuous Bandwidth (MBytes/sec)	Latency (PPC Clocks)		Continuous Bandwidth (MBytes/sec)	
	Best Case	Bus Idle		Best Case	Bus Idle		
Burst Write	5-1-1-1	5-1-1-1	133	5-1-1-1	5-1-1-1	267	5:2
Burst Read	42-1-1-1	42-1-1-1	133	32-1-1-1	33-1-1-1	267	
Single Write	5	5	-	5	5	-	
Single Read	26	33	-	23	24	-	
Burst Write	5-1-1-1	5-1-1-1	267	5-1-1-1	5-1-1-1	533	3:2
Burst Read	28-1-1-1	33-1-1-1	267	22-1-1-1	24-1-1-1	533	
Single Write	5	5	-	5	5	-	
Single Read	19	24	-	18	24	-	
Burst Write	5-1-1-1	5-1-1-1	133	5-1-1-1	5-1-1-1	267	3:1
Burst Read	48-1-1-1	51-1-1-1	133	35-1-1-1	42-1-1-1	267	
Single Write	5	5	-	5	5	-	
Single Read	30	33	-	27	33	-	
Burst Write	5-1-1-1	5-1-1-1	133	5-1-1-1	5-1-1-1	267	2:1
Burst Read	35-1-1-1	42-1-1-1	133	27-1-1-1	33-1-1-1	267	
Single Write	5	5	-	5	5	-	
Single Read	23	24	-	22	24	-	
Burst Write	5-1-1-1	5-1-1-1	267	5-1-1-1	5-1-1-1	355	1:1
Burst Read	22-1-1-1	24-1-1-1	267	18-1-1-1	24-1-1-1	304	
Single Write	5	5	-	5	5	-	
Single Read	16	24	-	15	15	-	

- Bus Idle represents a PowerPC master continually retrying a transaction until the transaction is ready. In some cases, the timing of transaction ready may not line up with the timing of repetitive retry cycles.

- Best Case represents a mixture of transactions allowing the most effective alignment between a retried transaction and transaction ready.
- Assumes write posting enabled and write posting FIFO is initially empty.
- Assumes read ahead enabled and read ahead FIFO is initially empty.
- Does not include time taken to obtain grant for PowerPC bus. The count starts on the same clock period that TS_ is asserted.
- PowerPC bus is idle at the time of the start of the transaction. (i.e. no pipeline effects).
- All transactions are cache word aligned.
- PCI medium responder with zero wait states.
- One clock request/one clock grant PCI arbitration.
- Default FIFO threshold settings.
- Clock counts represent best case alignment between PCI and PowerPC clock domains. An exception to this is continuous bandwidth which reflects the average affects of clock alignment.

Inbound Performance

Transaction	32-bit PCI		64-bit PCI		Clock Ratio
	Latency (PCIClocks)	Continuous Bandwidth (MBytes/sec)	Latency (PCIClocks)	Continuous Bandwidth (MBytes/sec)	
Burst Write	3-1-1-1...	133	3-1-1-1...	267	5:2
Burst Read	9-1-1-1...	133	9-1-1-1...	267	
Single Write	3	-	3	-	
Single Read	9	-	9	-	

Transaction	32-bit PCI		64-bit PCI		Clock Ratio
	Latency (PCIClocks)	Continuous Bandwidth (MBytes/sec)	Latency (PCIClocks)	Continuous Bandwidth (MBytes/sec)	
Burst Write	3-1-1-1...	267	3-1-1-1...	533	3:2
Burst Read	13-1-1-1...	267	13-1-1-1...	533	
Single Write	3	-	3	-	
Single Read	13	-	13	-	
Burst Write	3-1-1-1...	133	3-1-1-1...	267	3:1
Burst Read	9-1-1-1...	133	9-1-1-1...	267	
Single Write	3	-	3	-	
Single Read	9	-	9	-	
Burst Write	3-1-1-1...	133	3-1-1-1...	267	2:1
Burst Read	11-1-1-1...	133	11-1-1-1...	267	
Single Write	3	-	3	-	
Single Read	11	-	11	-	
Burst Write	3-1-1-1...	267	3-1-1-1...	426	1:1
Burst Read	16-1-1-1...	267	16-1-1-1...	388	
Single Write	3	-	3	-	
Single Read	16	-	16	-	

- Assumes write posting enabled and write posting FIFO is initially empty.
- Assumes read ahead enabled and read ahead FIFO is initially empty.
- All transactions targeted to PowerPC memory space hosted by the Harrier SDRAM interface.
- Does not include time taken to obtain grant for PCI Bus. The count starts on the same clock period that FRAME_ is asserted.
- One clock request/one clock grant PowerPC bus arbitration.
- All transactions are cache word aligned.
- Default FIFO threshold settings.

- Clock counts represent best case alignment between PCI and PowerPC clock domains. An exception to this is continuous bandwidth which reflects the average affects of clock alignment.

Programming Considerations

5

Programming SDRAM Related Control Registers

The following subsections contain information that is helpful in programming a system that uses the Harrier ASIC.

Initializing SDRAM Related Control Registers

In order to establish proper SDRAM operation, software must configure control register bits that affect each SDRAM bank's speed, refresh period, size, base address, and enable. The SDRAM speed attributes are the same for all banks and are controlled by one register. The same is true for the refresh period. On the other hand, the size, base address and enable can be different for each bank and are controlled in bank specific registers.

SDRAM Speed Attributes

The SDRAM speed attributes come up from power-up reset initialized to the slowest settings capable by Harrier. This allows SDRAM accesses to be performed before the SDRAM speed attributes are known. An example of a need for this is when software requires some working memory that it can use while gathering and evaluating SDRAM device data from serial EEPROMs.

SDRAM Refresh Period

The SDRAM per row refresh period comes up from power-up reset initialized to 7.75us. For performance reasons, software should set the refresh period to the longest value that still meets the device manufacturer's requirements.

SDRAM Size

The SDRAM size control bits come up from power-up reset cleared to zero. Once software has determined the correct size for an SDRAM bank, it should set the bank's size bits to match. The value programmed into the size bits tells the Harrier how big the bank is (for map decoding), and how to translate that bank's PowerPC addresses to SDRAM addresses. Programming a bank's size to non-zero also allows it to participate in scrubbing if scrubbing is enabled.

I^2C EEPROMs

Most of the information needed to program the SDRAM speed attributes, refresh period, and size is provided by EEPROM devices that are connected to the Harrier's I^2C bus. The EEPROM's devices contain data in a specific format called Serial Presence Detect (SPD).

Board designers can implement one EEPROM for each of the Harrier's SDRAM banks or they can implement one EEPROM for several such banks. When using DIMMs, the board designer can use the EEPROM that is provided on the DIMM.

I^2C EEPROMs that are used for SPD can be wired to appear at one of 8 different device locations. Board designers should establish an I^2C EEPROM addressing scheme that allows software to know which I^2C address to use to find information for each SDRAM bank. For example, hardware could always place the I^2C EEPROM for SDRAM bank A at the first address, bank B at the second, etc. Whatever addressing scheme is used should also deal with cases where multiple banks are described by one I^2C EEPROM.

SDRAM Base Address and Enable

Each bank needs to be programmed for a unique base address that is an even multiple of its size. Once a bank's speed attributes, size, and base address is programmed, it can be enabled.

SDRAM Control Registers Initialization Example

The following is a possible sequence for initializing SDRAM control registers:

1. Get a small piece of SDRAM for software to use for this routine. (Optional)

   This routine assumes that SDRAM related control bits are still at the power-up-reset default settings. We will use a small enough piece of SDRAM that the address signals that are affected by SDRAM size will not matter.

   For each SDRAM bank:

   a. Set the bank's base address to some even multiple of 32Mbytes. (Refer to the previous section titled "SDRAM Bank (A, B, C, D, E, F, G, and H) Addressing Registers".)

   b. Set the bank's size to 4Mx16 and enable it. (Refer to the previous section titled "SDRAM Bank (A, B, C, D, E, F, G, and H) Addressing Registers".)

   c. Test the first 1Mbyte of the bank.

   d. If the test fails, disable the bank, clear its size to 0Mbytes, disable it and then repeat steps a-d with the next bank. If the test passes, go ahead and use the 1st 1M of the bank.

2. Using the I^2C bus, determine which memory banks are present.

   Using the addressing scheme established by the board designer, probe for SPDs to determine which banks of SDRAM are present. SPD byte 0 could be used to determine SPD presence. SPD Byte 5 indicates the number of SDRAM banks that belong to an SPD.

3. Obtain the CAS_ latency information for all banks that are present to determine whether to set or to clear the ***XCSR*.SDTC**.CL3 bit.

   For each SDRAM bank that is present:

   a. Check SPD byte 18 to determine which CAS latencies are supported.

b. If a CAS_ latency of 2 is supported, then go to step 3. Otherwise a CAS_ latency of 3 is all that is supported for this bank.

c. If a CAS_ latency of 2 is supported, check SPD byte 23 to determine the CAS_latency_2 cycle time. If the CAS_latency_2 cycle time is less than or equal to the period of the system clock then this bank can operate with a CAS_ latency of 2. Otherwise a CAS_ latency of 3 is all that is supported for this bank.

   If any bank does not support a CAS_ latency of 2, then ***XCSR*.SDTC**.CL3 is to be set. If all of the banks **do** support a CAS_ latency of 2, then the ***XCSR*.SDTC**.CL3 bit is to be cleared.

   Do not update the ***XCSR*.SDTC**.CL3 bit at this point. You will use the information from this step later.

4. Determine the values to use for TRAS,TRP,TRCD, and TRC.

   The values to use for ***XCSR*.SDTC**.TRAS, ***XCSR*.SDTC**.TRP, ***XCSR*.SDTC**.TRCD, and ***XCSR*.SDTC**.TRC can be obtained from the SPD. ***XCSR*.SDTC**.TRAS determines the minimum tRAS time produced by the Harrier. ***XCSR*.SDTC**.TRP determines the minimum tRP time produced by the Harrier, etc. Each set of bits should accommodate the slowest bank of SDRAM. The SPD parameters are specified in nanoseconds and have to be converted to PowerPC clock periods for the Harrier.

   Use the table on the following page to convert SPD bytes 27, 29, and 30 to correct values for TRAS,TRP,TRCD, and TRC.

Do not actually update these bits in Harrier at this time. You will use the information from this step later.

Table 5-1. Deriving TRAS, TRP, TRCD and TRC Control Bit Values from SPD

Control Bits	Parameter	Parameter Expressed in CLK Periods	Possible Control Bit Values	
XCSR.SDTC.TRAS	tRAS (SPD Byte 30)	tRAS_CLK = tRAS/T (T = CLK Period in nanoseconds) See Notes 1, 2 and 9	0.0 < tRAS_CLK <= 4.0	TRAS =%00
			4.0 < tRAS_CLK <=5.0	TRAS =%01
			5.0 < tRAS_CLK <= 6.0	TRAS =%10
			6.0 < tRAS_CLK <= 7.0	TRAS =%11
			7.0 < tRAS_CLK	Illegal
XCSR.SDTC.TRP	tRP (SPD Byte 27)	tRP_CLK = tRP/T (T = CLK Period in nanoseconds) See Notes 3, 4 and 9	0.0 < tRP_CLK <= 2	TRP =%0
			2.0 < tRP_CLK <= 3	TRP =%1
			3 < tRP_CLK	Illegal
XCSR.SDTC.TRCD	tRCD (SPD Byte 29)	tRCD_CLK = tRCD/T (T = CLK Period in nanoseconds) See Notes 5, 6 and 9	0.0 < tRCD_CLK <= 2	TRCD =%0
			2.0 < tRCD_CLK <= 3	TRCD =%1
			3 < tRCD_CLK	Illegal
XCSR.SDTC.TRC	tRC (SPD Bytes 30 and 27)	tRC_CLK = (tRAS + tRP)/T (T = CLK Period in nanoseconds) See Notes 7, 8 and 9	0.0 < tRC_CLK <= 6.0	TRC =%110
			6.0 < tRC_CLK <= 7.0	TRC =%111
			7.0 < tRC_CLK <= 8.0	TRC =%000
			8.0 < tRC_CLK <= 9.0	TRC =%001
			9.0 < tRC_CLK <= 10.0	TRC =%010
			10.0 < tRC_CLK <= 11.0	TRC =%011
			11.0 < tRC_CLK	illegal

Notes

1. Use tRAS from the SDRAM bank that has the slowest tRAS.
2. tRAS_CLK is tRAS expressed in CLK periods.
3. Use tRP from the SDRAM bank that has the slowest tRP.
4. tRP_CLK is tRP expressed in CLK periods.
5. Use tRCD from the SDRAM bank that has the slowest tRCD.
6. tRCD_CLK is tRCD expressed in CLK periods.
7. Use tRC from the SDRAM bank that has the slowest tRC.
8. tRC_CLK is tRC expressed in CLK periods.
9. Remember that CLK is Harrier's PowerPC clock input pin.

5. Determine the size for each bank that is present.

   (Do not actually program the Harrier's size bits at this point. You use this information to program them later).

   Each bank's size can be determined using the following algoritm:

   a. Calculate the number of rows in each device using SPD byte 3. If the number of rows is *ROWS* and the value in SPD byte 3 is *R*, then

      $ROWS = 2^R$

   b. Calculate the number of columns in each device using SPD byte 4. If the number of columns is *COLUMNS* and the value in SPD byte 4 is *C*, then

      $COLUMNS = 2^C$

   c. Calculate the total number of addresses within each device. If the total number of addresses in a device is *A*, then

      $A = ROWS \times COLUMNS$

   d. Calculate the total number of locations in the bank using the results of step c and SPD byte 17. If the total number of locations in the bank is *L*, and the value in byte 17 is 4, then

      $L = A \times 4$

or

$L = 2^R \times 2^C \times 4$

(Note that Harrier only works if byte 17 is 4.)

e. Obtain the primary device width from SPD byte 13.

f. Determine the size bits based on the results of steps d and e using the following table:

Table 5-2. Programming the SDRAM SIZ Bits

Total Number of Locations within the Bank (L) [1]	Primary Device Width [2]	Bank Size [3]	Value to be programmed into the Bank's SIZE bits [4]
4M	16	32Mbytes	%0001
8M	8	64Mbytes	%0010
8M	16	64Mbytes	%0011
16M	4	128Mbytes	%0100
16M	8	128Mbytes	%0101
16M	16	128Mbytes	%0110
32M	4	256Mbytes	%0111
32M	8	256Mbytes	%1000
32M	16	256Mbytes	%1001
64M	4	512Mbytes	%1010
64M	8	512Mbytes	%1011
64M	16	512Mbytes	%1100
128M	4	1024Mbytes	%1101
128M	8	1024Mbytes	%1110
256M	4	2048Mbytes	%1111

Notes

1. Total Number of bank Locations(L) is 2^R x 2^C x 4 where R is the value in SPD byte 3 and C is the value in SPD byte 4.

2. Primary Device Width is from SPD byte 13.

3. Bank Size is the total number of bank locations (L) x 8 bytes.

4. Refer to the section titled "SDRAM Bank (A, B, C, D, E, F, G, and H) Addressing Registers for more information.

6. Determine the refresh rate from SPD byte 12 using the following table:

Table 5-3. Programming the SDRAM Refresh Period

SPD Byte 12 Value	Device Minimum Refresh Rate	Example Data Sheet Values	Value to be programmed into ***XCSR*.SDGC.**MXRR	Resulting Refresh Rate
$00	1 row per 15.6us	4096 rows, 64ms	%00	1 row per 15.5us
$01	1 row per 3.9us	16384 rows, 64ms	%10	1 row per 3.75us
$02	1 row per 7.8us	8192 rows, 64ms	%01	1 row per 7.75us
$03	1 row per 31.3us	-	%00	1 row per 15.5us
$04	1 row per 62.5us	-	%00	1 row per 15.5us
$05	1 row per 125us	-	%00	1 row per 15.5us
$06-$FF	N/A	N/A	N/A	N/A

7. Make sure software is no longer using SDRAM, and disable the bank that was being used.

8. Write to the SDRAM control registers.

   a. Program the SDRAM Timing Control Register using the information obtained in steps 3 and 4 and the fact that the ***XCSR*.SDTC**.WDPL bit should be set to 1. Be careful not to alter ***XCSR*.SDTC**.SDER.

   b. Program the ***XCSR*.SDGC**.MXRR bits in the SDRAM General Control Register using the information obtained in step 6. Note that ***XCSR*.SDGC**.DREF and ***XCSR*.SDGC**.RWCB should be cleared, ***XCSR*.SDGC**.ENRV and ***XCSR*.SDGC**.SWVT should

be programmed as desired, and ***XCSR*.SDGC**.DERC should be left set until software has initialized all of SDRAM.

c. Program the SDRAM Bank (A, B, C, D, E, F, G and H) Addressing Registers. Each bank's base address should be programmed so that it is an even multiple of its size. Only those banks that exist should be enabled. Also, only those that exist should be programmed with a non-zero size. (The size information was obtained in step 5.)

9. SDRAM is now ready to be initialized for use.

Optional Method for Sizing SDRAM

Generally SDRAM bank sizes can be determined by using SPD information. (Refer to the section titled "SDRAM Control Register Initialization Example".) Another method for accomplishing this is as follows:

1. Initialize the SDRAM interface control register bits to a known state.

   a. Make sure the PowerPC Clock Frequency Register matches the operating frequency.

   b. Make sure that the SDRAM Timing Control Register contains its power-up reset values. If not, make sure that the values match the actual characteristics of the SDRAM being used.

   c. Make sure that the Error Exception Enable Register does not enable SDRAM error exceptions.

   d. Make sure the following bits are initialized as follows:

   ***XCSR*.SDGC**.MXRR = 0,1,

   ***XCSR*.SDGC**.DREF = 0,

   ***XCSR*.SDGC**.RWCB = 0,

   ***XCSR*.SDGC**.DERC = 1,

   ***XCSR*.SDGC**.ENRV = 0,

   ***XCSR*.SDGC**.SWVT = 0,

***XCSR*.SDSC**.SCPA = $00.

(Refer to the sections titled "SDRAM General Control Register" and "SDRAM Scrub Control Register" for more information).

e. Make sure that no other resources (internal or external to Harrier) are set up to respond in the range $00000000 - $4000001F.

2. For each of Banks A - H:

a. Set the bank's base address to $00000000. (Refer to the section titled "SDRAM Bank (A, B, C, D, E, F, G and H) Addressing Registers.)

b. Enable the bank and make sure that the other seven banks are disabled. (Refer to the section titled "SDRAM Bank (A, B, C, D, E, F, G and H) Addressing Registers.)

c. Set the bank's size control bits. Start with the largest possible (2048Mbytes). (Refer to the section titled "SDRAM Bank (A, B, C, D, E, F, G and H) Addressing Registers.)

d. Write a unique 64-bit data pattern to each one of a specified list of addresses. The list of addresses to be written varies depending on the size that is currently being checked. The address lists are shown below.

Table 5-4. Address Lists for Different Bank Size Checks

Size	Addresses to Check	Notes
2048MB (256Mx4)	$00000000, $00010000,$40000000	
1024MB (128Mx8)	$00000000, $00008000,$20000000	
1024MB (128Mx4)	$00000000, $00010000	
512MB (64Mx16)	$00000000, $10000000	
512MB (64Mx8)	$00000000, $00008000,$10000000	1
512MB (64Mx4)	$00000000, $00008000,$10000000	1
256MB (32Mx16)	$00000000, $00004000,$08000000	2
256MB (32Mx8)	$00000000, $00004000,$08000000	2
256MB (32Mx4)	$00000000, $00008000	

Table 5-4. Address Lists for Different Bank Size Checks

Size	Addresses to Check	Notes
128MB (16Mx16)	$00000000, $04000000	
128MB (16Mx8)	$00000000, $00004000	3
128MB (16Mx4)	$00000000, $00004000	3
64MB (8Mx16)	$00000000, $00002000	4
64MB (8Mx8)	$00000000, $00002000	4
32MB (4Mx16)	$00000000, $00001000	5

Notes

1. 64Mx8 and 64Mx4 are the same. If the real size is either one of these, this algorithm will program for 64Mx8 regardless of whether the SDRAM size is 64Mx8 or 64Mx4. This is not a problem because the Harrier behaves identically when programmed for either size.

2. 32Mx16 and 32Mx8 are the same. The same idea that applies to 64Mx8 and 64Mx4 applies to them.

3. 16Mx8 and 16Mx4 are the same. The same idea that applies to 64Mx8 and 64Mx4 applies to them.

4. 8Mx16 and 8Mx8 are the same. The same idea that applies to 64Mx8 and 64Mx4 applies to them.

5. This is needed only to check for non-zero size.

e. Read back all of the addresses that have been written.

   If all of the addresses still contain exactly what was written, then the bank's size has been found. It is the size for which it is currently programmed.

   If any of the addresses do not contain exactly what was written, then the bank's memory size is less than that for which it is programmed. Sizing needs to continue for this bank by programming its control bits to the next smaller size and repeating steps d and e.

f. If no match is found for any size then the bank is unpopulated and has a size of 0MB. Its size should be programmed to 0.

Operation without Firmware

The Harrier includes features that allow a host processor on the PCI bus to initialize it, load the operating code into SDRAM and start the local processor without on board firmware. For example, a Processor PCI Mezzanine Card (PPMC) can be designed to operate without on board firmware when it is used in an application where their is a host processor on the PCI bus. In this application, the PPMC would be installed on a host board and the processor on the host board would initialize the PPMC, download the operating code and start the local processor.

When this mode is used, the processor is held in reset (HRESET_ is asserted) after the board reset signal is negated (Harrier's RST_ signal). This is accomplished by pulling up the XAD[26] signal. The host processor on the PCI bus must have access to the Harrier's PCI registers. This is accomplished by pulling down XAD[27] and XAD[28]. Refer to the section titled *Hardware Configuration* on page 2-133 for more information.

The host processor on the PCI bus programs Harrier's PCI base address registers to allow access to Harrier's PowerPC (60x) bus control registers and the SDRAM. The host processor on the PCI bus then initializes the SDRAM controller and the SDRAM. The SDRAM needs to be initialized with the correct CRCs before the ECC logic can be enabled. Also the ***XCSR*.SDGC**.ENRV bit must be set. Refer to the section titled *SDRAM General Control Register* on page 3-25 for more information. When this bit is set, the reset vector address range ($FFF0000- $FFFFFFFF) is mapped to the SDRAM bank at $00000000. The host processor on the PCI bus can then load the operation code into the SDRAM. The local processor is released by negating the ***XCSR*.BXCS**.P0H bit. Refer to the section titled *Bridge PowerPC Control and Status Register* on page 3-41 for more information. When the local processor starts executing code at $FFF00100, the hardware will translate the address to location $00000100 in the SDRAM.

The processor on the PCI bus can completely initialize the Harrier and the SDRAM, or it can initialize only the minimum required features to allow the local processor to start and then the local processor can complete the initialization.

Related Documentation

A

Motorola Computer Group Documents

The Motorola publications listed below are referenced in this manual. You can obtain paper or electronic copies of Motorola Computer Group publications by:

- Contacting your local Motorola sales office
- Visiting Motorola Computer Group's World Wide Web literature site, http://www.motorola.com/computer/literature.

Table A-1. Motorola Computer Group Documents

Document Title	Publication Number
PrPMC800 Processor PMC Installation and Use	PRPMC800A/IH
PrPMC800 Programmer's Reference Guide	PRPMC800A/PG
PPCBug Firmware Package User's Manual (Parts 1 and 2)	PPCBUGA1/UM PPCBUGA2/UM
PPCBug Diagnostics Manual	PPCDIAA/UM

To obtain the most up-to-date product information in pdf or html format, visit our web site at http://www.motorola.com/computer/literature.

Manufacturers' Documents

For additional information, refer to the following table for manufacturers' data sheets or user's manuals. As an additional help, a source for the listed document is provided. Please note that, while these sources have been verified, the information is subject to change without notice.

Table A-2. Manufacturers' Documents

Document Title and Source	Publication Number
PowerPC750™ RISC Microprocessor Technical Summary Motorola Literature Distribution Center Telephone: (800) 441-2447 or (303) 675-2140 FAX: (602) 994-6430 or (303) 675-2150 WebSite: http://e-www.motorola.com/webapp/DesignCenter/ E-mail: ldcformotorola@hibbertco.com	MPC750/D
PowerPC750™ RISC Microprocessor User's Manual MPC7410™ RISC Microprocessor User's Manual Literature Distribution Center for Motorola Semiconductor Products Telephone: (800) 441-2447 FAX: (602) 994-6430 or (303) 675-2150 WebSite: http://e-www.motorola.com/webapp/DesignCenter/ E-mail: ldcformotorola@hibbertco.com OR IBM Microelectronics Programming Evnironment Manual Web Site: http://www.chips.ibm.com/techlib/products/powerpc/manuals	MPC750UM/AD MPC7400UM/AD G522-0290-01

Table A-2. Manufacturers' Documents (Continued)

Document Title and Source	Publication Number
PowerPC™ Microprocessor Family: The Programming Environments Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 http://e-www.motorola.com/webapp/DesignCenter/ E-mail: ldcformotorola@hibbertco.com OR IBM Microelectronics Programming Evnironment Manual Web Site: http://www.chips.ibm.com/techlib/products/powerpc/manuals	MPCFPE/AD G522-0290-01
Intel 82559ER Fast Ethernet PCI Bus Controller with Integrated PHY — External Design Specification; Intel Corporation; http://developer.intel.com/design/network/datashts/738259.htm	73825902.pdf
3 Volt Intel Strata FLASH Memory, 28F128J3A Web Site: http://developer.intel.com/design/flcomp/prodbref/298044.htm	
TL 16C550C UART Texas Instruments P.O. Box 655303 Dallas, Texas 75265 Web Site: http://www.ti.com	SLLS177C
ATMEL Nonvolitile Memory Data Book Must request documentation at: http://www.atmel.com/atmel/support/	AT24Cxx AT93CV6
3 Volt Intel StrataFlash Memory, 28F128J3A Web Site: http://developer.intel.com/design/flcomp/prodbref/298044.htm	290667-005

Related Specifications

Table A-3 lists the product's related specifications. The appropriate source for the listed document is also provided. Please note that in many cases, the information is preliminary and the revision levels of the documents are subject to change without notice.

Table A-3. Related Specifications

Document Title and Source	Publication Number
IEEE - Common Mezzanine Card Specification (CMC) Institute of Electrical and Electronics Engineers, Inc. http://standards.ieee.org/catalog/	P1386, Draft 2.1
IEEE - PCI Mezzanine Card Specification (PMC) Institute of Electrical and Electronics Engineers, Inc. http://standards.ieee.org/catalog/	P1386.1, Draft 2.1
IEEE Standard for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Institute of Electrical and Electronics Engineers, Inc. http://standards.ieee.org/catalog/	IEEE 802.3
Peripheral Component Interconnect (PCI) Local Bus Specification, Revision 2.1 PCI Special Interest Group http://www.pcisig.com/	PCI Local Bus Specification
Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange (EIA-232-D) Electronic Industries Alliance http://www.eia.org/	TIA/EIA-232 Standard
PowerPC Reference Platform (PRP) Specification, Third Edition, Version 1.0, Volumes I and II International Business Machines Corporation http://www.ibm.com	MPR-PPC-RPU-02

Table A-3. Related Specifications (Continued)

Document Title and Source	Publication Number
PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture (CHRP), Version 1.0 Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 http://e-www.motorola.com/webapp/DesignCenter/ E-mail: ldcformotorola@hibbertco.com OR Morgan Kaufmann Publishers, Inc. Telephone: (415) 392-2665 Telephone: 1-800-745-7323 http://www.mkp.com/books_catalog/	ISBN 1-55860-394-8
VITA-32-199x Processor PMC Standard for Processor PMC Mezzanine Cards VITA Standards Organization http://www.vita.com/	VITA32 Draft 0.2

Index

Symbols

_ 2-132

Numerics

60x bus
    relation of address ranges to Xport chip select 2-95
60x slave
    relation to Xport bus master 2-96
8259 compatibility 2-71
8259 mode 2-81

A

INT 2-132
Access Timing (DRAM) 4-1
address offset
    Inbound Translation Function 2-28
Address Pipelining 2-7
address transfers
    responses to 2-5
Arbiers
    function described 1-5
arbiter
    PCI bus 2-115
arbiters
    internal PowerPC bus 2-113
ARTRY_ 2-5
assertion, definition xxii
asterisk (*) xxii

B

BAR
    read-write characteristics 2-28
BASE (Bank Base Address)
    SDRAM Addressing Registers 3-30
base address
    XCSR Register Group 3-3
Base Address Register (BAR)
    role in inbound translation function 2-27
binary number xxii
BRDFL_ 2-132
bridge
    control logic subdivisions 2-12
    inbound transactions 2-13
    outbound transactions 2-12
bus cycles
    types 2-36
Bus Hog 2-36
bus number 2-40
BXCS register
    role in copyback snarfing 2-35
    role with Bus Hog 2-36
byte write
    I2C 2-82
byte, definition xxiii

C

cache coherency 2-5
channel
    Xport plus 60x bus 2-95
CL3 (Cas Latency 3)
    SDRAM Timing Control Register (SDTC) 3-27
clock ratios
    supported 2-1
clock synchronization 2-2

clocking 2-1
CMP bit 2-82
compelled burst write transactions
    PPC Slave 2-18
CONFIG_ADDRESS 2-38
CONFIG_DATA 2-38
configuration cycles 2-38
Configuration Mechanism #1 2-38, 3-19
control bit, definition xxiii
conventions, manual xxii
Copy-back Snarfing
    controlling register (BXCS) 2-35
copyback snarfing 2-35
copy-back write cycles 2-35
counter
    as part of watchdog timer 2-120
Critical Word First (CWF) transfers 2-20
CSR's Readability 2-69
Current Address Read
    I2C 2-87
current task priority
    processor 2-69
Current Task Priority Level 2-81
cycle types
    SDRAM ECC 2-7

D

data
    types within Xport transactions 2-97
data parity
    PowerPC data 2-130
data transfers
    how done 2-5
decimal number xxii
decoding
    configuration devices 2-40
delayed transaction protocol
    PPC Slave 2-17
DERC (Disable Error Correction)
    SDRAM General Control Register (SDGC) 3-26
DEVSEL_ pin
    for mapping PCGS Register Group 2-25
direct delivery mode 2-72
distributed delivery mode 2-73
DMA Controller
    function described 1-5
Doorbell registers 2-66
double word, definition xxiii
DREF (Disable Refresh)
    SDRAM General Control Register (SDGC) 3-26

E

EEPROM
    use with SDRAM configuration 5-2
ENB (Bank Enable)
    SDRAM Addressing Registers 3-31
ENRV (Enable Reset Vector)
    SDRAM General Control Register (SDGC) 3-26
EOI register
    MPIC 2-81
EOS (Error On Scrub)
    SDRAM Single-bit Error Status Register (SDSES) 3-34
EREADY 2-132
Error Diagnostics
    function described 1-6
error diagnostics
    Harrier registers used 2-127
error exceptions
    how reported 2-71
error handling
    Inbound Function 2-36
error logging
    SDRAM errors 2-10
error reporting
    SDRAM data 2-8
errors
    outbound 2-24
ESB (Error Scrub Bank
    SDRAM Single-bit Error Status Register (SDSES) 3-34

INDEX

ESB (Error Scrub Bank)
SDRAM Multi-bit Error Status (SDMES) 3-37
ESYN (Error Syndrome)
SDRAM Single-bit Error Status (SDSES) 3-34
exceptions
error and functional 2-122

F

false, definition xxiii
features
of Harrier ASIC 1-1
Flash
compatibility to Hawk mode 2-109
four-beat Reads/Writes
SDRAM accesses 2-6
functional exceptions
how reported 2-71

H

half-word, definition xxiii
hardware configuration 2-133
Harrier
description 1-1
device number/configuration space connection 3-21
features 1-1
operation without firmware 5-13
resource table 3-1
Hawk Compatibility Mode 2-109
hexadecimal character xxii
host/slave
Harrier to EEPROMs 2-82

I

I2C
bus data transfer 2-82
Current Address Read 2-87
data transfer registers 2-82
Page Write 2-89
Sequential Read
Sequential Read
I2C 2-91
I2C Clock Prescaler (I2PSx) Register 2-82
I2C Control (I2COx) Register 2-82
I2C Controller
function described 1-5
I2C interface 2-82
I2C Receiver Data (I2RDx) Register 2-82
I2C serial data (SDAx) 2-82
I2C Status (I2STx) Register 2-82
I2C Transmitter Data (I2COx) Register 2-82
I2C Transmitter Data (I2TDx) Register 2-82
I2COx Register 2-83
Current Address Read 2-87
Page Write 2-89
Random Read 2-85
Sequential Read 2-91
I2O Message Passing 2-60
I2O/Generic Message Passing
function described 1-5
I2RDx Register
Current Address Read 2-87
Random Read 2-85
Sequential Read 2-91
I2STx Register 2-83
Current Address Read 2-87
Page Write 2-89
Random Read 2-85
Sequential Read 2-91
I2TDx Register 2-83
Current Address Read 2-87
Page Write 2-89
Random Read 2-85
Sequential Read 2-91
IDSEL lines
in configuration lines 2-39
role in configuration cycles 2-39
IMU Enable 2-64
IMU Interrupts 2-65
IMU Queue Structure 2-63
Inbound FIFO

role, components 2-32
inbound tansaction
key elements 2-25
inbound transaction
defined 2-25
Inbound Translation Function
address offset 2-28
initializing
SDRAM control registers 5-3
In-Service Register (ISR) 2-76
Interprocessor Interrupts (IPI) 2-70
Interrupt Acknowledge Cycles 2-41
interrupt acknowledge register
MPIC 2-81
interrupt delivery modes 2-72
interrupt events
nesting 2-70
Interrupt Pending Register (IPR) 2-75
Interrupt Request Register (IRR) 2-76
Interrupt Router 2-76
Interrupt Selector (IS) 2-75
interrupt source priority 2-69
interrupts
for DMA Controller 2-58
IOP Agent ID 2-65
IOP Message Unit 2-61
IRQ0_
active state 2-77
ITATx registers
role in read-ahead mode 2-35
role in write-posting mode 2-34

L

latency requirements 2-31

M

manual terminology xxii
manufacturers' documents A-2
map decode
PPC 2-14
masters
PowerPC bus 2-113
Memory and I/O Cycles 2-37
message passing 2-60
Message Passing Registers 2-67
Motorola Computer Group documents A-1
MPIC
block diagram 2-73
changing I/O interrupt configuration 2-80
EOI register 2-81
external interrupt service 2-78
features, architecture 2-68
function described 1-5
interprocessor interrupts 2-80
interrupt acknowledge register 2-81
operational characteristics 2-80
programming notes 2-78
reset state 2-79
MXRR (Multiply the Refresh Rate)
SDRAM General Control Register 3-25

N

naming conventions
PowerPC/PCI 2-11
negation, definition xxii

O

OTATx registers
role in memory and I/O cycles 2-37
OTOFx register
role in I/O address translation 2-37
role in memory and I/O handling 2-37
Outbound FIFO 2-19
outbound functions
PowerPC-to-PCI bus 2-14
Outbound Translation Function
attributes 2-15
Outbound Translation Function 3 3-19

P

Page Write
I2C 2-89
Passive Slave

role 2-23
PCFS Register Group
role in inbound transactions 2-25
PCI Arbiter 2-115
PCI bus #0 2-40
PCI bus cycles
types 2-36
PCI Bus Errors
as outbound error type 2-24
PCI Configuration Space (PCFS) Register Group 3-21
PCI Master
role 2-19
PCI Message Passing (PMEP) Register Group 3-23
PCI Message Passing Register Group
role in inbound transactions 2-30
PCI Slave
types of transactions accepted 2-30
PCLK
relation to XCLK 2-2
PIAC register
role in interrupt acknowledge cycles 2-41
PLL
as part of clocking mechanism 2-1
operational characteristics 2-2
PMEP
contents 3-23
PowerPC Address Bus Timer
function described 1-6
PowerPC address bus timer 2-129
PowerPC Address Space
role in inbound transactions 2-27
PowerPC bus
internal arbiters 2-113
PowerPC bus masters 2-113
PowerPC Bus Slave
relationship to SDRAM 2-5
PowerPC clock
relation to PCI clock 2-2
PowerPC Control and Status Register Group 3-3
PowerPC Multi-Processor Interrupt Controller Register Group 3-14
PowerPC Parity
function described 1-6
PowerPC to PCI Bridge 3-38, 4-8
function described 1-4
inbound performance 4-9
operating characteristics 2-11
PowerPC to PCI Configuration Space (XCFS) Register Group 3-19
PPC Decode 2-14
PPC Master
transfer modes, role 2-33
PPC Slave
as passive slave 2-23
delayed transaction protocol 2-17
role 2-16
PPMC signals 2-132
pre-scaler
use with timers 2-72
prescaler
as part of watchdog timer 2-120
priority
current task (processor) 2-69
priority scheme
for PowerPC bus masters 2-114
PRK Encoding
PCI bus 2-119
program visible registers 2-75
programming
watchdog timers 2-120
programming considerations 5-1

R

Random Read
I2C 2-85
I2COx Register 2-85
read-ahead mode 2-35
related documentation A-1
related specifications A-4

INDEX

reset signals 2-131
reset state
MPIC 2-79
resources
Harrier 3-1
rule sets
Interrupt Router 2-76
RWCB (Read/Write Checkbits)
SDRAM General Control Register (SDGC) 3-26

S

SCCNT (Scrub Counter)
SDRAM Scrub Control Register (SDSC) 3-32
SCON_ 2-132
SCPA (Scrub Prescaler Adjust)
SDRAM Scrub Control Register (SDSC) 3-32
scrub cycles
SDRAM Scrub Control 2-10
SCWE (Scrub Write Enable)
SDRAM Scrub Control Register (SDSC) 3-32
SDER (SDRAM External Registers or Buffers)
SDRAM Timing Control Register (SDTC) 3-29
SDMEA (SDRAM Multi-bit Error Address)
SDRAM Multi-bit Error Address Register (SDMEA) 3-37
SDRAM
control registers 5-1
initializing 5-1
interface characteristics 2-4
performance summary 4-1
refresh period 5-1
setting base address 5-2
size configuration 5-2
speed attributes 5-1
SDRAM accesses
four-beat Reads/Writes 2-6
single-beat Reads/Writes 2-6
SDRAM Bank Addressing Registers 3-30
SDRAM control registers
initializing 5-3
SDRAM Controller
operating characteristics 2-6
SDRAM ECC 2-7
SDRAM General Control Register (SDGC) 3-25
SDRAM interface
function described 1-4
SDRAM Multi-bit Error Address Register (SDMEA) 3-37
SDRAM Multi-bit Error Status Register (SDMES) 3-36
SDRAM Scrub Address Counter (SDSA) 3-33
SDRAM Scrub Control Register (SDSC) 3-32
SDRAM Single-bit Error Status Register (SDSEA) 3-36
SDRAM Single-bit Error Status Register (SDSES) 3-34
SDRAM sizing
optional method 5-3, 5-9
SDRAM Timing Control Register 3-27
SDSA (SDRAM Scrub Address
SDRAM Scrub Address Counter (SDSA) 3-33
SDSEA (SDRAM Single-bit Error Address)
SDRAM Single-bit Error Address Register (SDSEA) 3-36
SDTC Register 3-27
SECNT (Single-bit Error Count)
SDRAM Single-bit Error Status Register (SDSES) 3-34
serial port
interface device 2-94
Serial Presence Detect (SPD)
used to configure SDRAM 5-2
single word, definition xxiii
single-beat Reads/Writes

SDRAM accesses 2-6
SIZE (Bank Size)
SDRAM Addressing Registers 3-30
sizing SDRAM
optional method 5-3, 5-9
snarfing
defined 2-35
special cycles 2-40
spread I/O address translation 2-38
spurioous vector generation 2-70
status bit, definition xxiii
store gathering mode 2-20
SWVT (Swap Vector table)
SDRAM General Control Register (SDGC) 3-26

T

TDPL (SDRAM timing paramter tDPL)
SDRAM Timing Control Register (SDTC) 3-29
terminations
types of (DMA transfers) 2-57
time-out function
PPC bus 2-129
timers 2-72
Timing (DRAM Access) 4-1
transfer throttling
DMA activity 2-59
transfers
address 2-5
data 2-5
TRAS (SDRAM timing parameter tRAS)
SDRAM Timing Control Register (SDTC) 3-28
TRC (SDRAM timing parameter tRC)
SDRAM Timing Control Register (SDTC) 3-27
TRCD (SDRAM timing parameter tRCD)
SDRAM Timing Control Register (SDTC) 3-29
TRP (SDRAM timing parameter tRP)
SDRAM Timing Control Register (SDTC) 3-29
true, definition xxiii

U

UART
as serial port device 2-94
function described 1-5

W

Watchdog Timers
function described 1-5
watchdog timers
control of 2-120
described 2-120
programming 2-120
WDPL (Wait on tDPL)
SDRAM Timing Control Register (SDTC) 3-28
word boundaries
in I/O address translations 2-38
word, definition xxiii
write-posting mode 2-34

X

XCLK
relation to PCLK 2-2
XCSR 3-3
XCSR Register Group 3-3
Xport
function described 1-5
interface to expansion bus 2-94
multiple data beats 2-99
Xport Block Diagram 2-95
Xport Bus
transaction examples 2-97
Xport bus
address mapping 2-106
byte mapping 2-110
XAD mapping 2-108
Xport bus interface 4-4
Xport Bus Master

INDEX

60x bus slave 2-95
Xport Bus master 2-96
attributes 2-96
Xport Bus transactions
phases 2-96
Xport chip select
relation to 60s address ranges 2-95