MPC8272ADSUG
3/2004
Rev. 0.1

# MPC8272ADS
# User Guide

freescale™
semiconductor

**Home Page:**
www.freescale.com
**email:**
support@freescale.com
**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescale.com
**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com
**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 2666 8080
support.japan@freescale.com
**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com
**For Literature Requests Only:**
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

freescale™
semiconductor

# Contents


| Paragraph Number | Title | Page Number |
|---|---|---|
| | **About This Book** | |
| | **Chapter 1 Overview** | |
| 1.1 | MPC8272ADS Specifications | 1-1 |
| 1.2 | MPC8272ADS Features | 1-2 |
| | **Chapter 2 Hardware Preparation and Installation** | |
| 2.1 | Unpacking Instructions | 2-1 |
| 2.2 | Hardware Preparation | 2-1 |
| 2.2.1 | Setting the $V_{DDL}$ Supply Voltage Level | 2-3 |
| 2.2.2 | Setting MODCK(1:3) for PLLs Multiplication Factor—SW5 (#6–#8) | 2-3 |
| 2.2.3 | Setting Hard Reset Configuration Source—JP9 | 2-4 |
| 2.2.4 | Setting Boot Source | 2-5 |
| 2.2.5 | Setting MODCKH(0:3) for PLL Multiplication Factors | 2-6 |
| 2.2.6 | Setting PCI_ARBITER for PCI Mode-Enabled | 2-6 |
| 2.2.7 | Setting PCI_DLL for PCI Mode-Enabled | 2-6 |
| 2.2.8 | Clock-In Source Selection | 2-7 |
| 2.2.9 | FCC1 Ethernet Port Mode—MII/RMII | 2-7 |
| 2.2.10 | FCC2 Ethernet Port Mode—MII/RMII | 2-8 |
| 2.2.11 | USB Speed Selection | 2-8 |
| 2.2.12 | USB Mode Selection | 2-8 |
| 2.2.13 | COP/JTAG Connection | 2-9 |
| 2.2.14 | Forced Parallel Port Connection | 2-9 |
| 2.2.15 | Power On/Off Switch | 2-9 |
| | **Chapter 3 Installation Instructions** | |

# Contents


| Paragraph Number | Title | Page Number |
|---|---|---|
| 3.1 | Installation Instructions | 3-1 |
| 3.1.1 | Host-Controlled Operation | 3-1 |
| 3.1.2 | Stand Alone Operation | 3-2 |
| 3.1.3 | COP/JTAG Connector—P21 | 3-3 |
| 3.1.4 | Terminal to MPC8272ADS RS-232 Connection | 3-3 |
| 3.1.5 | 10/100-Base-T Ethernet Ports Connection | 3-4 |
| 3.1.6 | Memory Installation | 3-4 |
| 3.1.6.1 | Flash Memory SIMM Installation | 3-4 |


## Chapter 4
## Operating Instructions


| | | |
|---|---|---|
| 4.1 | Controls and Indicators | 4-1 |
| 4.1.1 | Power-On RESET Switch—SW3 | 4-1 |
| 4.1.2 | ABORT Switch—SW6 | 4-1 |
| 4.1.3 | Soft Reset Switch—SW7 | 4-1 |
| 4.1.4 | Hard Reset Switches—SW6 & SW7 | 4-1 |
| 4.1.5 | Reset Configuration Switch—SW2 | 4-2 |
| 4.1.6 | Software Options Switch—SW1 | 4-2 |
| 4.1.7 | 3IDDL Measurement—JP1 | 4-2 |
| 4.1.8 | Thermal Sense Connector—JP7 | 4-2 |
| 4.1.9 | IDDH Measurement—JP14 | 4-3 |
| 4.1.10 | $V_{pp}$ Source Selector—JP4 | 4-3 |
| 4.1.11 | GND Bridges | 4-4 |
| 4.1.12 | Power O.K. Indicator—LD13 | 4-4 |
| 4.1.13 | 12V Indicator—LD14 | 4-4 |
| 4.1.14 | UTOPIA Multi PHY Indicator—LD28 | 4-4 |
| 4.1.15 | 5-V Indicator—LD15 | 4-4 |
| 4.1.16 | 3.3-V Indicator—LD16 | 4-4 |
| 4.1.17 | USB Power Indicator—LD5 | 4-4 |
| 4.1.18 | -12V Indicator—LD17 | 4-4 |
| 4.1.19 | RUN Indicator—LD18 | 4-5 |
| 4.1.20 | ATM ON Indicator—LD19 | 4-5 |
| 4.1.21 | Fast Ethernet Port 1 Enabled—LD20 | 4-5 |
| 4.1.22 | Fast Ethernet Port 2 Enabled—LD22 | 4-5 |
| 4.1.23 | RS232 Port 1 ON—LD23 | 4-5 |
| 4.1.24 | Fast Ethernet Port 1 Full Duplex Indicator—LD1 | 4-5 |
| 4.1.25 | RS232 Port 2 ON—LD24 | 4-5 |
| 4.1.26 | Fast Ethernet Port 1 100Base-Tx Indicator—LD2 | 4-6 |
| 4.1.27 | Ethernet Port 1 Tx/Rx Indicator—LD3 | 4-6 |
| 4.1.28 | General Purpose Led 2 Indicator—LD26 | 4-6 |
| 4.1.29 | Ethernet Port 1 LINK Indicator—LD4 | 4-6 |

# Contents


| Paragraph Number | Title | Page Number |
|---|---|---|
| 4.1.30 | Fast Ethernet Port 2 Full Duplex Indicator—LD6 | 4-6 |
| 4.1.31 | General Purpose Led 1 Indicator—LD25 | 4-6 |
| 4.1.32 | Fast Ethernet Port 2 100Base-Tx Indicator—LD7 | 4-6 |
| 4.1.33 | USB Enabled Indicator—LD21 | 4-6 |
| 4.1.34 | Ethernet Port 2 LINK Indicator—LD9 | 4-7 |
| 4.1.35 | Ethernet Port 2 Tx/Rx Indicator—LD8 | 4-7 |
| 4.1.36 | $V_{DDL}$ Indication—LD27 | 4-7 |
| 4.1.37 | Parallel Port connection—LD10 | 4-7 |
| 4.1.38 | External Debugger Connection Indicator—LD11 | 4-7 |


## Chapter 5
## Module Design


| | | |
|---|---|---|
| 5.1 | Reset and Reset Configuration | 5-1 |
| 5.1.1 | Power-On Reset | 5-1 |
| 5.1.2 | Power-On Reset Configuration | 5-1 |
| 5.1.3 | Hard Reset | 5-2 |
| 5.1.3.1 | COP/JTAG Port Hard Reset | 5-2 |
| 5.1.3.2 | Manual Hard Reset | 5-3 |
| 5.1.3.3 | Internal Sources Hard Reset | 5-3 |
| 5.1.3.4 | Hard Reset Configuration | 5-3 |
| 5.1.4 | Soft Reset | 5-7 |
| 5.1.4.1 | COP/JTAG Port Soft Reset | 5-8 |
| 5.1.4.2 | Manual Soft Reset | 5-8 |
| 5.1.4.3 | Internal Sources Soft Reset | 5-8 |
| 5.1.5 | PCI Bus Reset | 5-8 |
| 5.1.6 | Local Interrupter | 5-9 |
| 5.1.6.1 | ABORT Interrupt | 5-9 |
| 5.1.6.2 | ATM UNI Interrupt | 5-9 |
| 5.1.7 | Fast Ethernet PHY Interrupt | 5-9 |
| 5.1.8 | PCI Interrupt | 5-9 |
| 5.2 | Clock Generator | 5-13 |
| 5.2.1 | MPC8272 Clock | 5-13 |
| 5.2.2 | PCI Clock | 5-13 |
| 5.3 | Bus Configuration | 5-14 |
| 5.3.1 | Single PowerQUICC II Mode | 5-14 |
| 5.4 | Buffering | 5-14 |
| 5.5 | Chip-Select Generator | 5-15 |
| 5.6 | Synchronous DRAM (60X Bus) | 5-15 |
| 5.6.1 | SDRAM Programming | 5-16 |
| 5.6.2 | SDRAM Refresh | 5-17 |
| 5.7 | Flash Memory SIMM | 5-17 |

# Contents


5.7.1 Flash Programming Voltage ..... 5-19
5.8 E2PROM Memory ..... 5-19
5.9 PCI Bus ..... 5-20
5.10 Communication Ports ..... 5-22
5.10.1 ATM Port ..... 5-22
5.10.2 100/10 Base T Ports ..... 5-23
5.10.2.1 DM9161 Control ..... 5-23
5.10.3 RS232 Ports ..... 5-24
5.10.3.1 RS-232 Ports' Signal Description ..... 5-24
5.10.4 USB Port ..... 5-25
5.10.5 PC Parallel Port ..... 5-25
5.11 Board Control and Status Register - BCSR ..... 5-25
5.11.1 BCSR0 Board Control—Status Register 0 ..... 5-27
5.11.2 BCSR1 Board Control—Status Register 1 ..... 5-27
5.11.3 BCSR2 Board Control—Status Register 2 ..... 5-28
5.11.4 BCSR3 Board Control - Status Register 3 ..... 5-31
5.11.5 BCSR4 Board Control—Status Register 4 ..... 5-32
5.11.6 BCSR5 and BCSR7 Board Control—Status Register 3 and 5 ..... 5-33
5.12 COP/JTAG Port ..... 5-33


## Chapter 6
## Memory Map


6.1 Overview ..... 6-1
6.2 PowerQUICC II Register Programming ..... 6-5
6.2.1 System Initialization ..... 6-6
6.2.2 Memory Controller Register Programming ..... 6-7


## Chapter 7
## Physical Properties


7.1 Power Supply ..... 7-1
7.1.1 5-V Rail ..... 7-3
7.1.2 3.3-V Rail ..... 7-3
7.1.3 5-V Standby Rail ..... 7-3
7.1.4 $V_{DDH}$ Rail ..... 7-3
7.1.4.1 $V_{DDL}$ Bus ..... 7-3
7.1.4.2 12-V Rail ..... 7-4
7.1.4.3 -12-V Rail ..... 7-4
7.2 Connectors ..... 7-4
7.2.1 ATX Power Connector ..... 7-4
7.2.2 Fast Ethernet Port Connectors ..... 7-4

# Contents


| Paragraph Number | Title | Page Number |
|---|---|---|
| 7.2.3 | ATM 155 Port Connection | 7-5 |
| 7.2.4 | RS232 Port Connector | 7-5 |
| 7.2.5 | CPM Expansion Connector | 7-5 |
| 7.2.6 | COP/JTAG Port Connector | 7-5 |
| 7.2.7 | Logic Analyzer Connectors | 7-5 |
| 7.2.8 | Mach's In System Programming (ISP) Connector | 7-5 |
| 7.2.9 | PCI Connectors | 7-5 |
| 7.2.10 | System Expansion Connector | 7-6 |
| 7.2.11 | USB Connector | 7-6 |
| 7.2.12 | Parallel Port Connector | 7-6 |
| 7.3 | PCB Layout | 7-6 |

**Chapter 8**
**Support**

| | | |
|---|---|---|
| 8.1 | Interconnect Signals | 8-1 |
| 8.1.1 | P13—RS232 Ports 1 and 2 Connectors | 8-1 |
| 8.1.2 | P10 and P23 100/10 Base-T Ethernet Port Connector | 8-2 |
| 8.1.3 | P21—COP/JTAG Connector | 8-2 |
| 8.1.4 | P1—CPM Expansion Connector | 8-4 |
| 8.1.5 | P9, P11, P14, P22, P17, P5, P20, P18, P15, P24,P12 Logic Analyzer MICTOR Connectors | 8-13 |
| 8.1.6 | P26, P28, P29—PCI Connectors | 8-13 |
| 8.1.7 | P31 - ATX Power Supply Connector | 8-15 |
| 8.1.8 | P3,P4,P25—Mach/Lattice ISP Connector | 8-16 |
| 8.1.9 | P2—System Expansion Connector | 8-17 |
| 8.1.10 | P16, P19—USB Connectors | 8-22 |
| 8.2 | Programmable Logic Equations | 8-23 |
| 8.2.1 | U3—BCSR Code | 8-23 |
| 8.2.2 | U41—Power switch debounce | 8-53 |

**Appendix A**
**Revision History**

**Index**

# Contents


| Paragraph Number | Title | Page Number |
|---|---|---|

# Figures


| Figure Number | Title | Page Number |
|---|---|---|
| 1-1 | MPC8272ADS Block Diagram | 1-4 |
| 2-1 | MPC8272ADS Top Side Part Location Diagram | 2-2 |
| 2-2 | $V_{DDL}$ Trimmer—RP1 | 2-3 |
| 2-3 | SW5 Description | 2-4 |
| 2-4 | Hard Reset Configuration Source Selection—JP9 | 2-5 |
| 2-5 | SW2 Description | 2-6 |
| 2-6 | Clock Source Selection | 2-7 |
| 2-7 | FCC1 Ethernet Mode Selection | 2-7 |
| 2-8 | FCC2 Ethernet Mode Selection | 2-8 |
| 2-9 | USB Mode Selection | 2-9 |
| 2-10 | Forced Parallel Port Connection | 2-9 |
| 3-1 | Host-Controlled Operation Scheme—Command Converter | 3-1 |
| 3-2 | Host Controlled Operation Scheme—Parallel Port | 3-2 |
| 3-3 | Stand Alone Configuration | 3-2 |
| 3-4 | P21—COP/JTAG Port Connector | 3-3 |
| 3-5 | P1A/P1B—RS232 Serial Port Connector | 3-3 |
| 3-6 | Flash Memory SIMM Insertion | 3-4 |
| 4-1 | SW1 Description | 4-2 |
| 4-2 | JP7 Therm Connector | 4-3 |
| 4-3 | JP13 $V_{PP}$ Source Selection | 4-3 |
| 5-1 | PCI Host Configuration Registers | 5-7 |
| 5-2 | PCI Interrupt Routing Scheme | 5-10 |
| 5-3 | Main Clock Generator Scheme | 5-13 |
| 5-4 | PCI Clock Generator Scheme | 5-14 |
| 5-5 | 60x SDRAM Connection Scheme | 5-16 |
| 5-6 | FLASH SIMM Connection Scheme | 5-18 |
| 5-7 | E2PROM Connection Scheme | 5-20 |
| 5-8 | PCI Bus Scheme | 5-21 |
| 5-9 | RS232 Serial Ports Connector | 5-24 |
| 5-10 | Debug Station Connection Schemes | 5-33 |
| 5-11 | COP/JTAG Port Connector | 5-34 |
| 7-1 | MPC8272ADS Power Scheme | 7-2 |

# Figures

# Tables


| Table Number | Title | Page Number |
|---|---|---|
| i | Acronyms and Abbreviated Terms | xvii |
| 1-1 | MPC8272ADS specifications | 1-1 |
| 2-1 | MODCK(1:3) Encoding | 2-4 |
| 5-1 | BCSR/FLASH Hard Reset Configuration Word | 5-4 |
| 5-2 | E2PROM Hard Reset Configuration Word | 5-6 |
| 5-3 | PCI Interrupt Register Description | 5-10 |
| 5-4 | PCI Interrupt Mask Register Description | 5-12 |
| 5-5 | MPC8272ADS Chip Select Assignments | 5-15 |
| 5-6 | 100 MHz SDRAM Mode Register Programming | 5-17 |
| 5-7 | BCSR0 Description | 5-27 |
| 5-8 | BCSR1 Description | 5-27 |
| 5-9 | BCSR2 Description | 5-28 |
| 5-10 | FLASH Presence Detect (7:5) Encoding | 5-29 |
| 5-11 | FLASH Presence Detect (4:1) Encoding | 5-30 |
| 5-12 | EXTOOLI(0:3) Assignment | 5-30 |
| 5-13 | PowerQUICC II Board Version Encoding | 5-30 |
| 5-14 | PowerQUICC II Board Revision Encoding | 5-30 |
| 5-15 | External Tool Revision Encoding | 5-31 |
| 5-16 | BCSR3 Description | 5-31 |
| 5-18 | PCI Board Present Signal Definitions | 5-32 |
| 5-17 | BCSR4 Description | 5-32 |
| 5-19 | BCSR5 to BCSR7 Description | 5-33 |
| 5-20 | COP/JTAG Port Signals Description | 5-34 |
| 6-1 | MPC8272ADS Memory Map—Flash (or BCSR) as Boot Device | 6-1 |
| 6-2 | MPC8272ADS Memory Map—E2PROM as Boot Device | 6-3 |
| 6-3 | BCSR/Flash Power On Reset Configuration | 6-6 |
| 6-4 | E2PROM Power On Reset Configuration | 6-6 |
| 6-5 | SIU Register Programming | 6-7 |
| 6-6 | Memory Controller Initialization For 100MHz—Flash as Boot Device | 6-7 |
| 6-7 | Memory Controller Initialization For 100MHz—E2PROM as Boot Device | 6-8 |
| 6-8 | Memory Controller Initialization For 100MHz | 6-9 |
| 7-1 | Expansion Connectors Maximum Current Consumption | 7-2 |
| 7-2 | Maximum Power Consumption Per Add-In Card | 7-2 |
| 8-1 | P13 Connector | 8-1 |
| 8-2 | P10,P23 100/10 Base-T Ethernet Connector | 8-2 |

# Tables


| Table Number | Title | Page Number |
|---|---|---|
| 8-3 | P16—COP/JTAG Connector | 8-2 |
| 8-4 | P1—CPM Expansion Connector | 8-4 |
| 8-5 | P26, P28, P29—PCI Connectors | 8-13 |
| 8-6 | P31—ATX Power Supply Connector | 8-15 |
| 8-7 | P3—Lattice ISP Connector | 8-16 |
| 8-8 | P2—System Expansion Connector | 8-17 |
| 8-9 | P16, P19—USB Connectors | 8-22 |

# About This Book

The MPC8272 is a versatile communications processor that integrates on one chip a high-performance PowerPC™ RISC microprocessor, a very flexible system integration unit, encryption hardware, and many communications peripheral controllers that can be used in a variety of applications, particularly in communications and networking systems.

The primary objective of this manual is to describe the functionality of the MPC8272ADS board. It contains operational, functional and general information about the MPC8272ADS. This board is meant to serve as a platform for software and hardware development for the MPC8272 processor in a TEPBGA package (516 Pins in Hip7). The high-performance MPC8272 PowerQUICC II™ processor family contains a PowerPC™ core and addresses the needs of a wide variety of networking and communications applications.

Using its on-board resources and a debugger, a developer can download code, run it, set breakpoints, display memory and registers and connect proprietary hardware using the expansion connectors, to be incorporated into a desired system with the MPC8272 processor.

This board could also be used as a demonstration tool (for example, application software may be programmed either on or off-board into its Flash memory and run in exhibitions).

Although this book describes aspects regarding the PowerPC architecture that are critical for understanding the MPC8272 core, it does not contain a complete description of the architecture. Where additional information might help the reader, references are made to *Programming Environments Manual for 32-Bit Implementation of the PowerPC Architecture, REV 2*. Refer to “Architecture Documentation” for ordering information.

The information is subject to change without notice, as described in the disclaimers on the title page of this book. As with any technical documentation, it is the reader’s responsibility to use the most recent version of the documentation. Before using this manual, determine whether it is the latest revision and the existence of errata or addenda. To locate any published errata or updates for this document, refer to the worldwide web at www.motorola.com/semiconductors.

## Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products using the MPC8272 integrated processor. Readers should understand the following topics:

- Computer networking
- Basic principles of RISC processing
- Details of the PowerPC architecture

## Organization

The following list describes the major sections of this manual:

- Chapter 1, “Overview,” provides a high-level description of the MPC8272ADS, describing general operation, basic features, and a block diagram.
- Chapter 2, “Hardware Preparation and Installation,” provides unpacking instructions and information about setting various modes, switches, and other configuration on the MPC8272ADS.
- Chapter 3, “Installation Instructions,” provides information about host-controlled and stand alone operation, memory installation, and connection setup.
- Chapter 4, “Operating Instructions,” provides information about controls and indicators for the MPC8272ADS.
- Chapter 5, “Module Design,” provides information about MPC8272ADS reset and reset configuration, the clock generator, bus configuration, buffering, the chip-select generator, synchronous DRAM, Flash memory, and $E^2$PROM memory.
- Chapter 6, “Memory Map,” discusses PowerQUICC II register programming.
- Chapter 7, “Physical Properties,” provides information about the power supply, connectors, and PCB layout.
- Chapter 8, “Support,” provides information about interconnect signals and programmable logic equations.
- Appendix A, “Revision History,” provides information about the revision history of this document.

The book also contains an index.

## Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about the PowerPC architecture.

## MPC8272 Documentation

Supporting documentation for the MPC8272 can be accessed through the world-wide web at www.motorola.com/semiconductors. This documentation includes technical specifications, reference materials, and detailed applications notes.

## Architecture Documentation

Architecture documentation is organized in the following types of documents:

- Manuals—These books provide details about individual implementations of the PowerPC architecture and are intended to be used with the *Programming Environments Manual*. These include the *G2 Core Reference Manual* (Motorola order #: G2CORERM).
- Programming environments manuals—These books provide information about resources defined by the PowerPC architecture that are common to processors that implement the PowerPC architecture. The two versions include one that describes the functionality of the combined 32- and 64-bit architecture models and one that describes only the 32-bit model. The MPC8280 adheres to the 32-bit architecture definition.
  - *Programming Environments for 32-Bit Implementations of the PowerPC Architecture* (Motorola order #: MPCFPE32B)
- *The Programmer's Pocket Reference Guide for the PowerPC Architecture*: MPCPRGREF/D—This guide provides an overview of registers, instructions, and exceptions for 32-bit implementations.
- Application notes—These short documents contain useful information about specific design issues useful to programmers and engineers working with Motorola's processors.

For a current list of documentation, refer to www.motorola.com/semiconductors.

## Related Documentation

Motorola documentation is available from the sources listed on the back cover of this manual. The document order numbers are included in parentheses for ease in ordering:

- *Programming Environments Manual for 32-Bit Implementations of the PowerPC Architecture*, which describes resources defined by the PowerPC architecture.
- The *G2 Core Reference Manual*, which describes features for the embedded G2 processor core, a derivative of the original MPC603e PowerPC microprocessor design.
- The *Tsi107™ PowerPC Host Bridge User Manual*, published by Tundra Semiconductor Corporation (Tundra). See the Tundra website at www.tundra.com for more information.

- User’s manuals—These books provide details about individual implementations and are intended for use with the *Programming Environments Manual.*
- Addenda/errata to user’s manuals—Because some processors have follow-on parts, an addendum describes the additional features and functionality changes. These addenda are intended for use with the corresponding user’s manuals.
- Hardware specifications—Hardware specifications provide specific data about bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations. Separate hardware specifications are provided for each part described in this book.
- Technical summaries—Each device has a technical summary that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of a user’s manual for implementation.
- *The Programmer’s Reference Guide for the PowerPC Architecture*—This concise reference includes the register summary, memory control model, exception vectors, and the PowerPC instruction set.
- *The Programmer’s Pocket Reference Guide for the PowerPC Architecture*—This foldout card provides an overview of PowerPC registers, instructions, and exceptions for 32-bit implementations.
- Application notes—These useful short documents address specific design issues for programmers and engineers who work with Motorola processors.

Additional literature is published as new processors become available. For a current list of documentation, refer to http://www.motorola.com/semiconductors.

## Conventions

This document uses the following notational conventions:

| | |
|---|---|
| cleared/set | The words ‘set’ and ‘cleared’ are reserved for use with bits and fields, but not signals. A bit or field is *cleared* when it contains the value zero. A bit is *set* when it contains the value one. However, a field can be set to a value that contains zeros and ones. |
| asserted/negated | The words ‘asserted’ and ‘negated’ are reserved for use with signals and not bits. |
| **mnemonics** | Instruction mnemonics are shown in lowercase bold. |
| *italics* | Italics indicate variable command parameters, for example, **bcctr***x*. |
| | Book titles in text are set in italics. |
| | Internal signals are set in italics, for example, $\overline{qual\ BG}$. |
| 0x0 | Prefix to denote hexadecimal number |
| 0b0 | Prefix to denote binary number |

| | |
|---|---|
| **r**A, **r**B | Instruction syntax that identifies a source GPR |
| **r**D | Instruction syntax that identifies a destination GPR |
| **fr**A, **fr**B, **fr**C | Instruction syntax that identifies a source FPR |
| **fr**D | Instruction syntax that identifies a destination FPR |
| REG[FIELD] | Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register. |
| x | In some contexts, such as signal encodings, an unitalicized x indicates a 'don’t care' logic. |
| *x* | An italicized *x* indicates an alphanumeric variable. |
| *n* | An italicized *n* indicates an numeric variable. |
| ¬ | NOT logical operator |
| & | AND logical operator |
| \| | OR logical operator |
| 0 0 0 0 | Indicates reserved bits or bit fields in a register. Although these bits can be written to as ones or zeros, they are always read as zeros. |

# Acronyms and Abbreviations

Table i contains acronyms and abbreviations that appear in this document.

**Table i. Acronyms and Abbreviated Terms**

| | |
|---|---|
| MPC8272ADS | ADS board For MPC8272 processor |
| PQ2 | PowerQUICC II |
| MPC8275 | PowerQUICC II Hip7 in TEPBGA package |
| VOYAGER | MPC8260 - PowerQUICC 2 |
| PPC | PowerPC |
| PCI | Peripheral components interconnect |
| USB | Universal serial bus |
| CPM | Communication processor module |
| SDRAM | Synchronous dynamic random access memory |
| VADS | Voyager application development system |
| Kbyte | 1024 bytes |
| LSB | Least significant byte |
| lsb | Least significant bit |
| Mbyte | 1048576 bytes |

## Table i. Acronyms and Abbreviated Terms (continued)

| | |
|---|---|
| MPC8272ADS | ADS board For MPC8272 processor |
| DIMM | Dual in-line memory module |
| SIMM | Single in-line memory module |
| TBD | To be defined |
| UPM | User programmable machine |
| EVB | Evaluation board |
| GPCM | General purpose chip-select machine |
| GPL | General purpose line |
| BCSR | Board control and status register |
| FLASH | Non volatile reprogrammable memory. |
| ZIF | Zero input force |
| BGA | Ball grid array |
| ADI | Application development interface. |
| COP | Common on-chip processor |
| SAR | Segmentation And reassembly |
| UTOPIA | Universal test & operations interface for ATM |

# Chapter 1
# Overview

The primary objective of this manual is to describe the functionality of the MPC8272ADS board. It contains operational, functional and general information about the MPC8272ADS. This board is meant to serve as a platform for software and hardware development for the MPC8272 processor in a TEPBGA package (516 Pins in Hip7). The high-performance MPC8272 PowerQUICC II™ processor family contains a PowerPC™ core and addresses the needs of a wide variety of networking and communications applications.

Using its on-board resources and a debugger, a developer can download code, run it, set breakpoints, display memory and registers and connect proprietary hardware using the expansion connectors, to be incorporated into a desired system with the MPC8272 processor.

This board could also be used as a demonstration tool (for example, application software may be programmed either on- or off-board into its Flash memory and run in exhibitions).

## 1.1 MPC8272ADS Specifications

Table 1-1 shows the MPC8272ADS specifications.

**Table 1-1. MPC8272ADS specifications**

| CHARACTERISTICS | SPECIFICATIONS |
|---|---|
| Power requirements (no other boards attached) | +5Vdc @ TBD A (Typ.), TBD A (Max.)<br>+3.3Vdc @ TBD A (Typ.), TBD A (Max.)<br>+12Vdc - @TBD A Max.<br>-12Vdc - @TBD A Max. |
| Microprocessor | MPC8272 running @ up to 100 MHz bus clock frequency. |
| Addressing<br>Total address range on PPC bus:<br>Total address range on Local bus:<br><br><br>Flash Memory SIMM (PPC bus)<br>Synchronous Dynamic RAM DIMM (PPC bus) | <br>4 GBytes (32 address lines)<br>256 KBytes External (18 address lines)<br>4 GBytes Internal (32 address lines internal decoding)<br><br>8 MByte, 32 bits wide expandable to 32 MBytes<br>64 MByte, 64 bits wide |

**Table 1-1. MPC8272ADS specifications (continued)**

| CHARACTERISTICS | SPECIFICATIONS |
|---|---|
| Operating temperature | 0°C - 70°C (room temperature) |
| Storage temperature | -25°C to 85°C |
| Relative humidity | 5% to 90% (non-condensing) |
| Dimensions:<br>Length<br>Width<br>Thickness | <br>12" (305 mm)<br>9" (229 mm)<br>0.063" (1.6 mm) |

## 1.2 MPC8272ADS Features

This section summarizes the features of the MPC8272ADS. Major features are the following:

- Supports MPC8272 (Hip7) processor
- 64-bit PowerQUICC II Communication Processor, running at up to 100MHz external bus frequency
- 64-MByte synchronous DRAM (soldered on-board), residing on 60X bus (PBI mode), controlled by SDRAM machine. Hard reset is applied by depressing BOTH Soft Reset & ABORT buttons.
- 8-MByte, 80-pin Flash SIMM, buffered from 60X bus. Supports up to 32 MBytes, controlled by GPCM, 5V/12V Programmable, with automatic Flash SIMM identification, using BCSR. Supports both On and OFF SIMM Flash reset.
- 5V/12V VPP (in-circuit programming voltage) for Flash SIMM—jumper selectable
- 64 KBytes $E^2$PROM, buffered from the 60x bus and controlled by the GPCM
- Board control and status register—BCSR, controlling boards' operation
- On-board COP/JTAG connector
- On-board logic to support direct connection to standard parallel port (EPP/SPP modes) in desktop PC for debug purposes using MetroWerks CodeWarrior tools
- Power-on reset option using JTAG
- Programmable power-on reset and hard-reset configuration using $E^2$PROM or Flash memory for the PowerQUICC II core
- PCI local bus is PCI Standard 2.2-compliant
- 3 PCI slots are available to host up to three masters/targets cards at 3.3 V only. Arbitration is supported by the on-chip arbiter.
- PCI bus supports 25–66 MHz at 3.3 V devices (determined by the user).
- Simple generic interrupt controller to handle the PCI interrupts (4 in each PCI slot)

- Module enable indications for all on-board modules
- High-density (MICTOR) logic analyzer connectors, carrying all 60x, PCI, and CPM signals, for fast logic analyzer connection
- 155 Mbps ATM UNI on FCC1 with optical I/F, connected to the PowerQUICC II via UTOPIA Level 2 I/F supporting 8 bit in single/multi PHY, using the PMC-SIERA 5384
- Two 100/10-Base-T Ports on FCC1 and FCC2 with T.P. I/F, MII/RMII controlled, using Davicom DM9161
- USB Port, USB 1.1 Standard Compliant, using Philips PDIUSBP11 USB transceiver. USB Port is with shutdown option and speed selectable - BCSR controlled. Host/slave selectable. Supports USB type-A and type-B connectors.
- Dual RS232 port residing on SCC1 & SCC4
- Module disable (that is, low-power mode) option for all communication transceivers —BCSR controlled, enabling use of communication ports, off-board by using the expansion connectors
- Dedicated PowerQUICC II communication ports expansion connectors for convenient tools' connection, carrying also necessary bus signals, for transceivers' M/P I/F connection. Use is done with 2 X 128 pin DIN 41612 receptacle connectors.
- External tools' identification & status read capability, via BCSR
- Separate power-on reset push button, soft / hard reset push button, and ABORT push button
- ATX power supply
- 1.3V to 1.7V (Hip7) internal logic operation voltage
- Software option switch provides 8 S/W options using BCSR.

Figure 1-1 shows the MPC8272ADS block diagram.

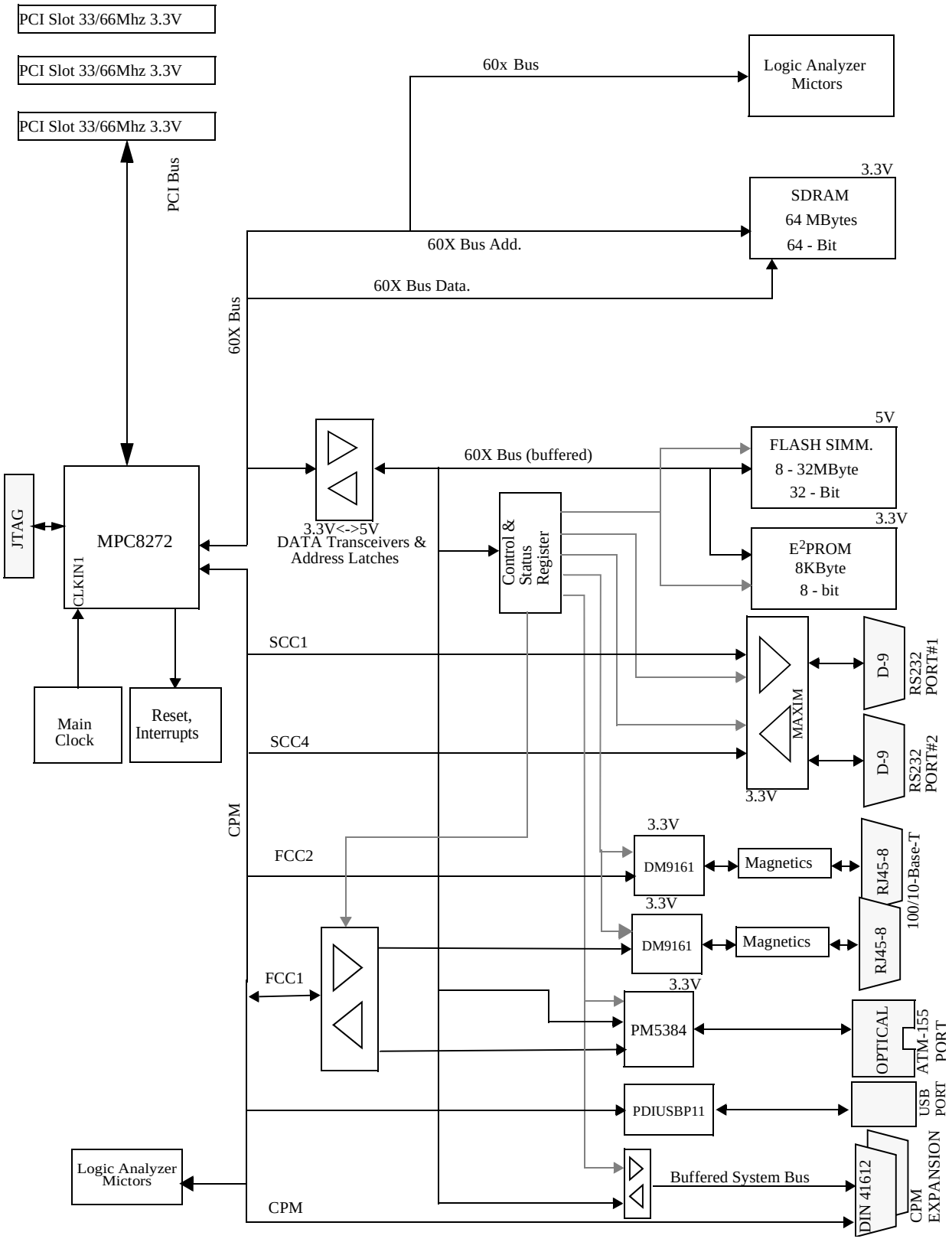



**Figure 1-1. MPC8272ADS Block Diagram**

# Chapter 2
# Hardware Preparation and Installation

This chapter provides unpacking instructions and hardware preparation for the MPC8272ADS.

## 2.1 Unpacking Instructions

If the shipping carton is damaged upon receipt, request that the carrier's agent is present during unpacking and inspection of equipment. Unpack the equipment from shipping carton. and refer to the packing list to verify that all items are included. Save the packing material for storing and reshipping the equipment.

**NOTE**

Avoid touching areas of integrated circuitry. Static discharge can damage the circuits of this product.

## 2.2 Hardware Preparation

To ensure that the desired configuration is selected to produce proper operation of the MPC8272ADS board, changes of the dip-switch settings may be required before installation. The location of the switches, indicators, dip-switches, and connectors is illustrated in Figure 2-1. Boards are factory-tested and shipped with dip-switch settings as described in the following tested graphs. Parameters can be changed for the following conditions:

The PowerQUICC II internal logic supply level within range ($V_{DDL}$) using potentiometer RP1.

- PowerQUICC II's MODCK(1:3). Determining core's and CPM's PLLs multiplication factor using dip-switches SW5(#6 - #8)
- PowerQUICC II's hard reset configuration word source - BCSR or memory (FLASH/EEPROM) using jumper JP9
- PowerQUICC II's boot code source EEPROM/FLASH using dip-switch SW2(1)
- PowerQUICC II's MODCKH(0:3) using SW5(1-4)
- PowerQUICC II's $\overline{\text{PCI_ARBITER}}$ using SW2(2)

- PowerQUICC II's PCI_DLL using SW2(3)
- Clock-in source—external or on-board clock oscillator—JP1
- FCC1 and FCC2 MII/RMII modes—using jumpers JP5 and JP10, respectively
- USB speed (12Mbits/s or 1.5Mbits/s) software controlled in BCSR
- USB host/slave mode—using jumper JP8
- PowerQUICC II's COP/JTAG connection—COP/JTAG connector (P21) or direct connection to PC parallel port (P27). Selected automatically by connecting parallel cable.
- Force PC parallel port (P27) connection using jumper JP12
- ATX power supply on/off switch using SW4

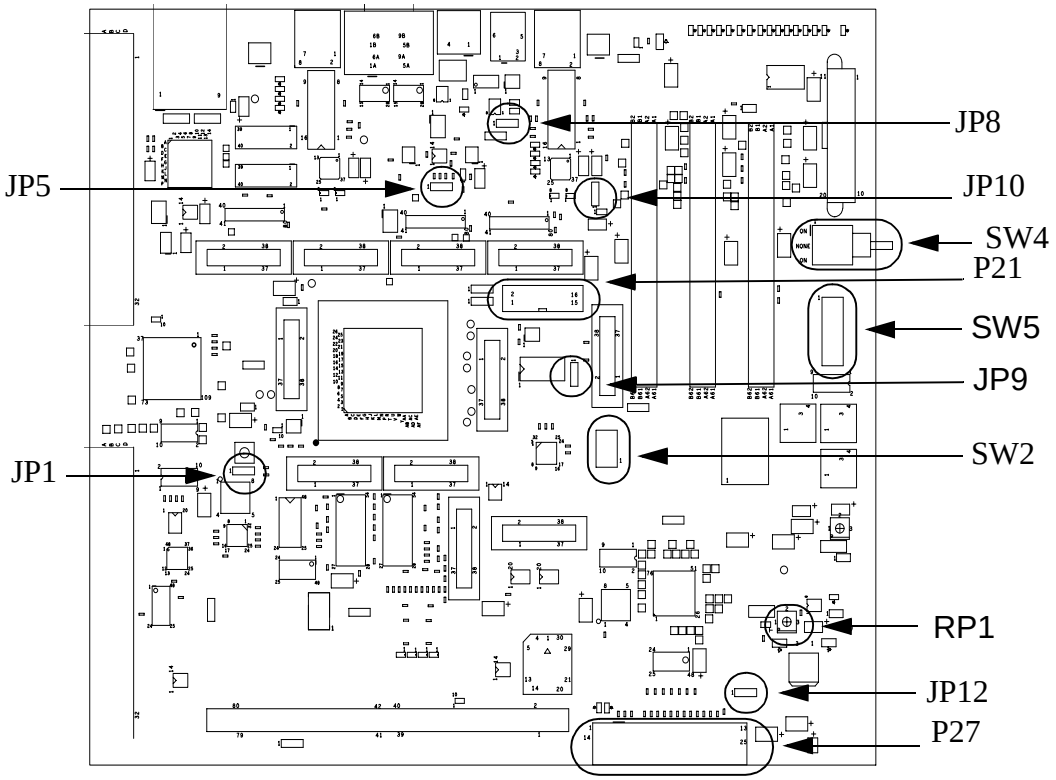



**Figure 2-1. MPC8272ADS Top Side Part Location Diagram**

### 2.2.1 Setting the $V_{DDL}$ Supply Voltage Level

The level of $V_{DDL}$ is tuned using RP1 and is in the range of 1.3 V to 1.7 V. $V_{DDL}$ may be measured upon JP13, using a DVM or any other high-input impedance voltage measuring device.

$V_{DDL}$ level is factory-set at the mid-range, but may be changed using RP1. Rotating RP1 CCW increases $V_{DDL}$ voltage up to range-high. Rotating it clockwise decreases $V_{DDL}$ down to range-low. LD27 provides visual indication for $V_{DDL}$ level. It illuminates brighter with the rise of $V_{DDL}$. $V_{DDL}$ changes Vs. RP1's rotation direction is shown in Figure 2-2.

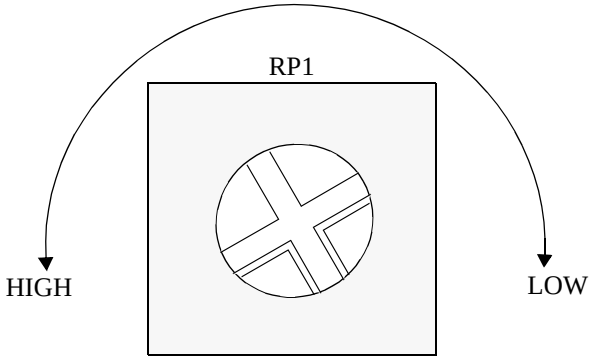



**Figure 2-2. $V_{DDL}$ Trimmer—RP1**

### 2.2.2 Setting MODCK(1:3) for PLLs Multiplication Factor—SW5 (#6–#8)

After 1K cycles and negation of the power-on reset signal, the MPC8272 samples the seven MODCK lines—the lower 3 on MODCK(1-3) and the upper four—MODCKH(0:3) field to establish the multiplication factors of the CPM's and core's PLLs. The levels on MODCK(1:3) lines are set using SW5 (see Figure 2-3), switches #6–#8. When an individual switch is at the OFF position, its associated MODCK line is pulled-high ('1'). When at the ON position, the associated MODCK is pulled-down ('0'). SW5 is shown in Figure 2-3, while the various combinations for SW5 (#6 - #8) and their associated MODCK(1:3) values are shown in Table 2-1.

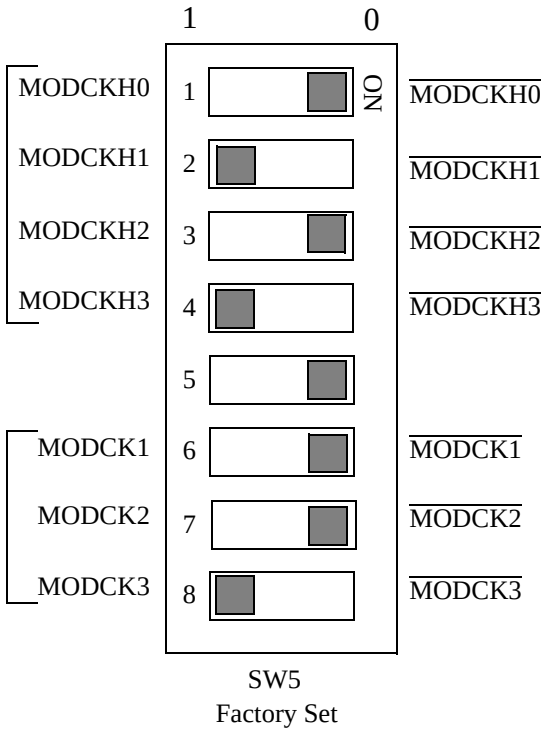



**Figure 2-3. SW5 Description**

**Table 2-1. MODCK(1:3) Encoding**

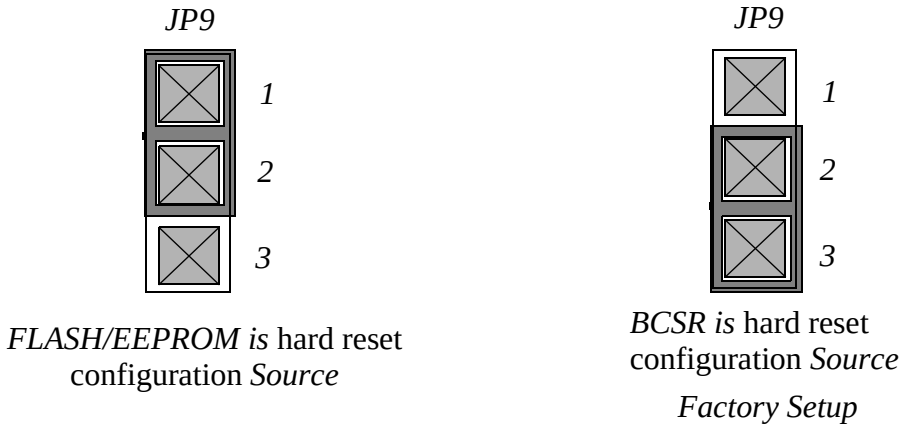
| MODCK(1:3) | Switch 6 | Switch 7 | Switch 8 |
|---|---|---|---|
| 0 | ON | ON | ON |
| 1 | ON | ON | OFF |
| 2 | ON | OFF | ON |
| 3 | ON | OFF | OFF |
| 4 | OFF | ON | ON |
| 5 | OFF | ON | OFF |
| 6 | OFF | OFF | ON |
| 7 | OFF | OFF | OFF |

## 2.2.3 Setting Hard Reset Configuration Source—JP9

The boot sequence that starts when $\overline{\text{HRESET}}$ is asserted can be from two sources:

- BCSR (default hard-reset configuration word—$\overline{\text{CS0}}$ is assumed to be assigned to the FLASH)
- Memories (FLASH/EEPROM—user controlled hard-reset configuration word)

When a jumper is placed between positions 1–2 of JP9, the hard reset configuration source is a memory (FLASH/EEPROM) as configured by switch SW6-1. When a jumper is set between positions 2–3 of JP7, the hard reset configuration source is the BCSR. See Figure 2-4.




*FLASH/EEPROM is* hard reset configuration *Source*

*BCSR is* hard reset configuration *Source*

*Factory Setup*

**Figure 2-4. Hard Reset Configuration Source Selection—JP9**

## 2.2.4 Setting Boot Source

The hard-reset configuration word, read by the MPC8272 while HRESET is asserted, may be taken from the following sources:

- Flash memory SIMM
- EEPROM
- BCSR

(In fact, eight hard-reset configuration words are read by a configuration master; however, only the first is relevant for a single MPC8272.)

For additional information about the contents of the hard-reset configuration word, see Section 5.1, “Reset and Reset Configuration.”

SW2#1 actually assigns $\overline{\text{CS0}}$ to the FLASH (default when booting from the BCSR) or to the EEPROM. When SW2 #1 is OFF, the hard-reset configuration word is taken from EEPROM. When it is ON, the hard-reset configuration word is taken from the Flash SIMM (see Figure 2-5).

.

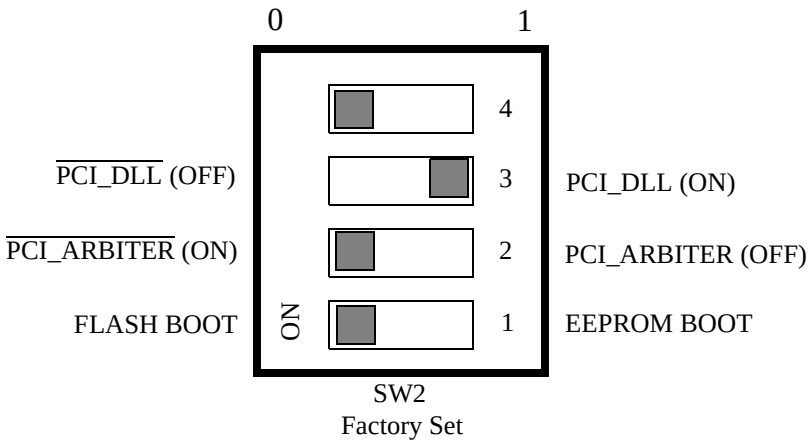



**Figure 2-5. SW2 Description**

## 2.2.5 Setting MODCKH(0:3) for PLL Multiplication Factors

Because the PCI mode in the PowerQUICC II (Hip4 and Hip7) is enabled, the MODCKH(0:3) lines are taken from SW5(#1 - #4) and the MODCKH(0:3) bits in the hard-reset configuration word are ignored. SW5(#1 - #4) set the upper 4 bits of the MODCK field during hard reset configuration acquisition. When an individual switch of SW4 #1 - #4 is at the OFF position, its corresponding MODCKH line is pulled-high (‘1’) during hard reset, while when at the ON position, pulled-down (‘0’) (see Figure 2-3).

## 2.2.6 Setting $\overline{\text{PCI_ARBITER}}$ for PCI Mode-Enabled

The settings of this line determine the operation of the PCI arbiter (when the PowerQUICC II is in PCI mode). When $\overline{\text{PCI_ARBITER}}$ is set low, the PCI arbiter in the PowerQUICC II is enabled. When set high, the PCI arbiter is disabled and an external arbiter can be used. When switch SW6 #2 is at the OFF position, its corresponding $\overline{\text{PCI_ARBITER}}$ line is pulled-high (‘1’ - disabled), while when at the ON position, pulled-down (‘0’ - enabled) (see Figure 2-5).

## 2.2.7 Setting PCI_DLL for PCI Mode-Enabled

The settings of this line determine the operation of the DLL for PCI mode-enabled. When PCI mode is enabled, the DLL must be enabled. When PCI_DLL is set low, the DLL is disabled. When set high, the DLL is enabled. When switch SW6 #3 is at the OFF position, its corresponding PCI_DLL line is pulled-high (‘1’ - enabled). When at the ON position, pulled-down (‘0’ - disabled) (see Figure 2-5).

## 2.2.8 Clock-In Source Selection

The main clock source can be selected between an external (off-board) source by connecting to P6 or an on-board clock oscillator. The selection is done by setting JP1. When a jumper is placed between positions 1–2 of JP1, the external clock source is enabled. When a jumper is placed between positions 2–3 of JP1, the on-board clock oscillator is enabled (see Figure 2-6).

.

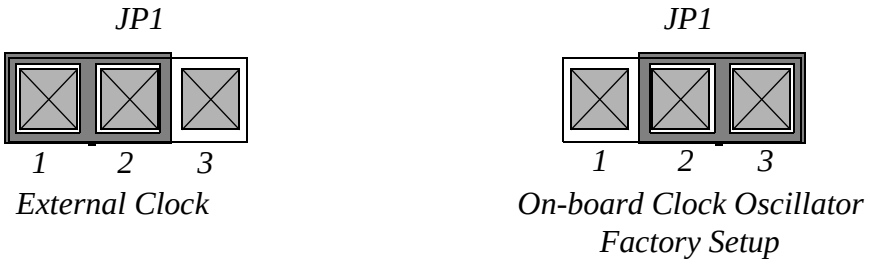



**Figure 2-6. Clock Source Selection**

## 2.2.9 FCC1 Ethernet Port Mode—MII/RMII

The Ethernet PHY on FCC1 is set by default to 100Base-Tx full duplex and can be configured to operate in MII or RMII interface. The selection is done by setting JP5. When a jumper is placed between positions 1–2 of JP5, the MII interface is enabled. When a jumper is placed between positions 2–3 of JP5, the RMII interface is enabled (see Figure 2-7).

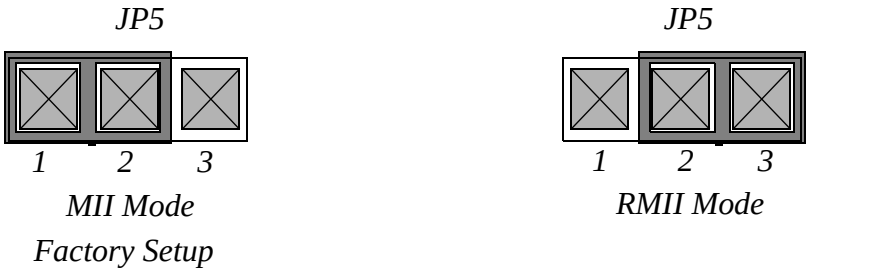



**Figure 2-7. FCC1 Ethernet Mode Selection**

**NOTE**

To cause a mode change, set JP5 while the board is powered-off.

## 2.2.10 FCC2 Ethernet Port Mode—MII/RMII

The Ethernet PHY on FCC2 is set by default to 100Base-Tx full duplex, and can be configured to operate in MII or RMII interface. The selection is done by setting JP10. When a jumper is placed between positions 1 - 2 of JP10, the MII interface is enabled. When a jumper is placed between positions 2–3 of JP10, the RMII interface is enabled (see Figure 2-8).




**Figure 2-8. FCC2 Ethernet Mode Selection**

**NOTE**

To cause the mode change, set JP10 while the board is powered-off.

## 2.2.11 USB Speed Selection

The USB port supports two speeds, 12 Mbits/s and 1.5 Mbits/s. The selection is software-controlled in the BCSR. At power-on reset, the default selection is 12Mbits/s.

## 2.2.12 USB Mode Selection

The USB port supports two modes, host and slave. The selection is done by setting JP8. When a jumper is placed between positions 1–2 of JP8, the slave mode is enabled. When a jumper is placed between positions 2–3 of JP8, the host mode is enabled (see Figure 2-9).




**Figure 2-9. USB Mode Selection**

## 2.2.13 COP/JTAG Connection

Either of two options can establish connection to the COP port of the PowerQUICC II: COP/JTAG connector or a parallel port of a PC. The COP/JTAG connector requires a command converter, and the second option connects directly to the parallel port of a PC and eliminates the need for one. The selection is done automatically. If a cable is connected to the parallel port in a PC, this connection has priority over the COP/JTAG connector.

## 2.2.14 Forced Parallel Port Connection

In some computers, because the parallel port does not comply fully with the parallel port standard, the connection of a cable to the parallel port connector (P27) does not automatically switch the COP/JTAG connection to the PC. Jumper JP12 is used to force the parallel port connection in such cases. When a jumper is placed between positions 1–2 of JP12, the parallel port connection mode is forced. When a jumper is placed between positions 2–3 of JP12, the parallel port connection mode is not forced (see Figure 2-10).

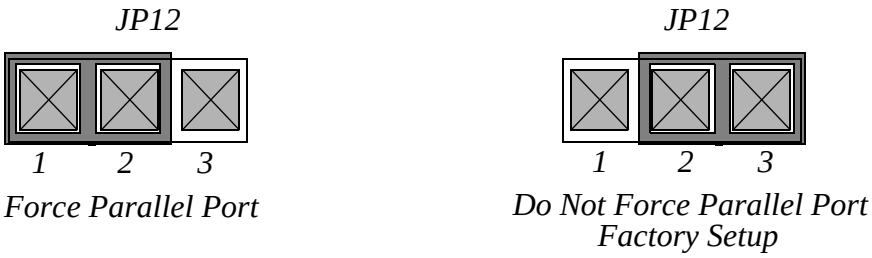



**Figure 2-10. Forced Parallel Port Connection**

## 2.2.15 Power On/Off Switch

Switching SW4 causes power-on reset or power-off.

# Chapter 3
# Installation Instructions

This chapter provides installation instructions for the MPC8272ADS.

## 3.1 Installation Instructions

When the MPC8272ADS has been configured as desired by the user, it can be installed according to the required working environment as follows:

- Host-controlled operation
- Stand-alone

### 3.1.1 Host-Controlled Operation

In this configuration, the MPC8272ADS is controlled by a host computer using the COP port, which is a subset of the JTAG port. This configuration allows for extensive debugging using on-host debugger. There are two options to connect to the COP port:

- The host is connected to the board by a COP controller (command converter) provided by a third party (see Figure 3-1).

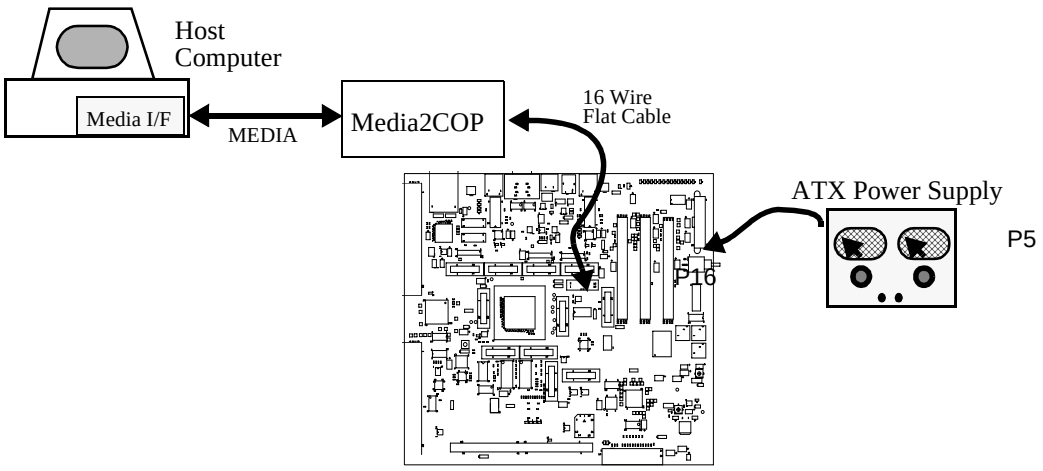



**Figure 3-1. Host-Controlled Operation Scheme—Command Converter**

- The host is connected to the board directly from the host's parallel port (see Figure 3-2).

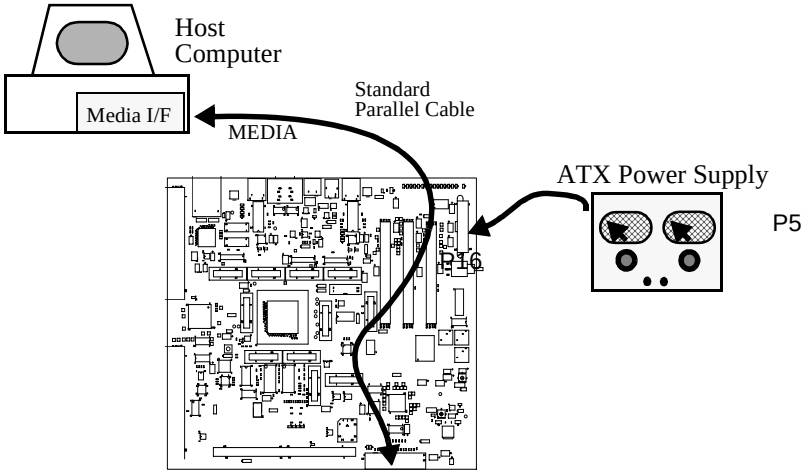



**Figure 3-2. Host Controlled Operation Scheme—Parallel Port**

## 3.1.2 Stand Alone Operation

In this mode (see Figure 3-3), the board is not controlled by the host by using the COP port. It may connect to the host by using one of its other ports (for example, the RS232 port), Fast Ethernet port, ATM155 port, and so on. Operating in this mode requires programming an application program into the board's Flash memory.

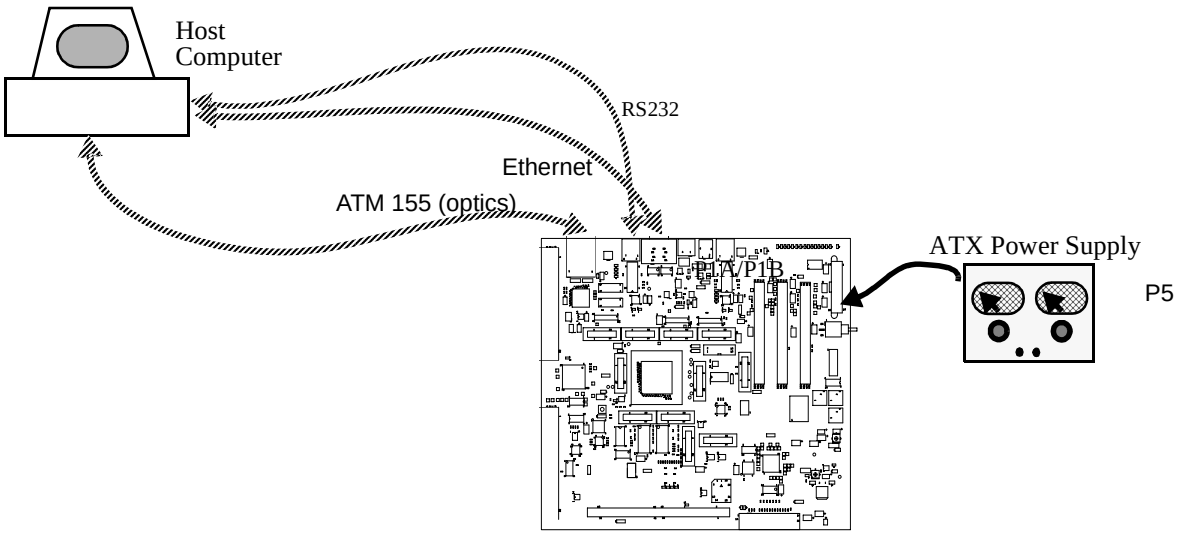



**Figure 3-3. Stand Alone Configuration**

### 3.1.3 COP/JTAG Connector—P21

The MPC8272ADS COP interface connector P21 is a 16-pin, male header connector. The connection between the MPC8272ADS and the COP controller is by a 16-line flat cable supplied with the COP controller board obtained from a third party developer. Figure 3-4 shows the pin configuration of the connector.

.

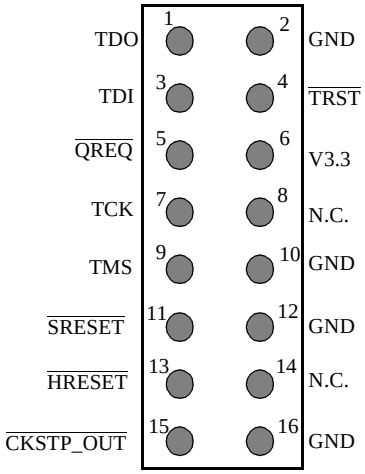



**Figure 3-4. P21—COP/JTAG Port Connector**

### 3.1.4 Terminal to MPC8272ADS RS-232 Connection

A serial (RS232) terminal or any other RS232 equipment may be connected to the RS-232 connectors P1A and P1B. The RS-232 connectors are 9-pin female D-type connectors arranged in a stacked configuration. P1B connected to SCC2 of the PowerQUICC II is the lower and P1A, connected to SCC1 of the PowerQUICC II, is the upper in the stack.

The connectors are arranged in a manner that allows for 1:1 connection with the serial port of an IBM-AT or compatibles, that is, by using a flat cable. The pinout that is identical for both P1A and P1B is shown in Figure 3-5.

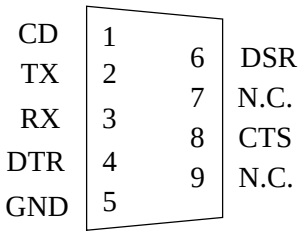



**Figure 3-5. P1A/P1B—RS232 Serial Port Connector**

### 3.1.5 10/100-Base-T Ethernet Ports Connection

The 10/100-Base-T port connectors P10 and P23 are an 8-pin, 90$^{o}$, receptacle RJ45 connector. A standard cable that has two RJ45/8 jacks on its ends connects the 10/100-Base-T ports to the network. The pinout of P10 and P23 is described in Table 8-2.

### 3.1.6 Memory Installation

The MPC8272ADS is supplied with one type of memory module, the Flash memory SIMM.

#### 3.1.6.1 Flash Memory SIMM Installation

To install a memory SIMM, remove it from its package, put it diagonally in its socket (U20), and raise it to a vertical position until the metal lock clips are locked (see Figure 3-6).

**CAUTION**

The memory SIMMs have alignment nibble near their # 1 pin. To avoid damage to both the memory SIMM and its socket, be sure to align the memory correctly before it is twisted.

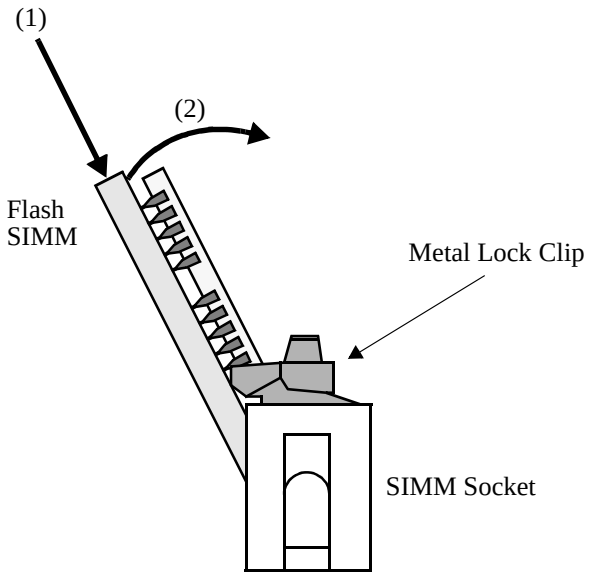



**Figure 3-6. Flash Memory SIMM Insertion**

# Chapter 4
# Operating Instructions

This chapter provides operating instructions for using the MPC8272ADS in host-controlled and stand-alone configurations. This information includes controls and indicators, memory map details, and software initialization of the board.

## 4.1 Controls and Indicators

The MPC8272ADS has the following switches and indicators.

### 4.1.1 Power-On RESET Switch—SW3

The power-on reset switch SW3 performs power-on reset to the MPC8272 as if the power was re-applied to the ADS. When the MPC8272 is reset that way, all configuration and all data residing in volatile memories is lost. After $\overline{\text{PORST}}$ signal is negated, the MPC8272 re-acquires the power-on reset and hard-reset configuration data from the hard-reset configuration source. (Flash | EEPROM | BCSR).

### 4.1.2 ABORT Switch—SW6

The ABORT switch ends program execution by issuing a level 0 interrupt to the MPC8272. If the ADS is in standalone mode, the user must provide a means for handling the interrupt, because the MPC8272ADS does not have a resident debugger. The ABORT switch signal is debounced and may be disabled by software.

### 4.1.3 Soft Reset Switch—SW7

The soft reset switch SW3 performs soft reset to the MPC8272 internal modules, maintaining MPC8272's configuration (clocks and chip-selects) and SDRAMs' contents. The switch signal is debounced and software cannot disable it.

### 4.1.4 Hard Reset Switches—SW6 & SW7

When BOTH switches SW6 and SW7 are depressed simultaneously, hard reset is generated to the MPC8272. When the MPC8272 is hard reset, all its configuration is lost except for

the hard-reset configuration word, which is acquired only once after PON-Reset. The loss includes data stored in the SDRAMs, and the MPC8272 must be re-initialized.

### 4.1.5 Reset Configuration Switch—SW2

SW2 is a 4-switch dip-switch. For its function see Section 2.2, “Hardware Preparation.”

### 4.1.6 Software Options Switch—SW1

SW1 is a 4-switch dip-switch that is connected over SWOPT(0:2) lines that are available at BCSR2. S/W options may be manually selected, according to SW1 state. SW1 is factory set to all ON (see Figure 4-1).

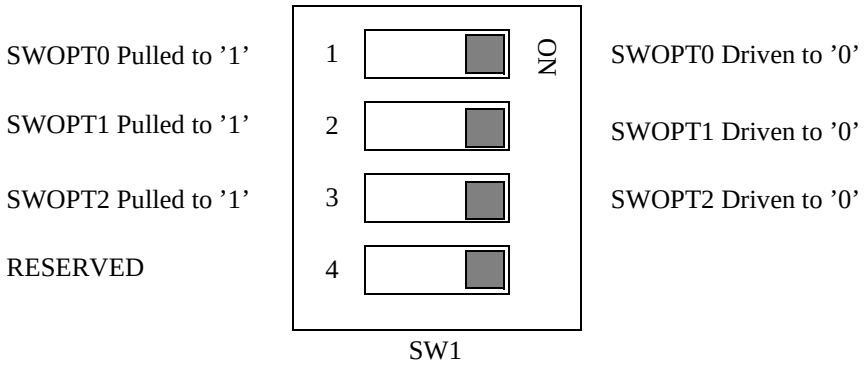



**Figure 4-1. SW1 Description**

### 4.1.7 3IDDL Measurement—JP1

JP13 resides in the IDDL main current flow. To measure IDDL, JP13 should be removed using a solder tool. A current meter should be connected instead with wires that are as short and thick as possible.

**Warning**

The job of removing JP13 and soldering the current meter connections instead is very delicate and should be done by a skilled technician.

If this process is done by unskilled hands or repeated more than 3 times, the MPC8272ADS can be permanently damaged.

### 4.1.8 Thermal Sense Connector—JP7

Two dedicated pins THERM(0:1) provide a way to take internal temperature measurements of the MPC8272. These pins should be connected to GND for normal operation. JP7 is

factory-set with a jumper on its 2–3 positions so that THERM1 is connected to GND (see Figure 4-2).

.

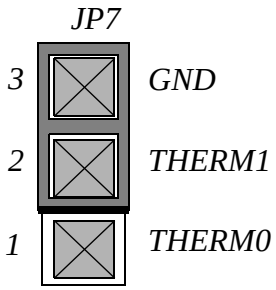



**Figure 4-2. JP7 Therm Connector**

## 4.1.9 IDDH Measurement—JP14

JP14 resides in IDDH's main current flow. To measure IDDH, JP14 should be removed using a solder tool and a current meter should be connected with wires that are as short and thick as possible.

**Warning**

The job of removing JP14 and soldering current meter connections instead is very delicate, and should be done by a skilled technician.

If this process is done by unskilled hands or repeated more than 3 times, the MPC8272ADS can be permanently damaged.

## 4.1.10 $V_{PP}$ Source Selector—JP4

JP4 selects the source for $V_{PP}$ programming voltage for the Flash SIMM. When a jumper is located between pins 2 and 3 of JP4, the $V_{PP}$ is connected to the VCC plane of the board, providing 5 V $V_{PP}$. When a jumper is located between positions 1 and 2 of JP4, $V_{PP}$ is drawn from the 12-V plane that provides 12V $V_{PP}$. JP4 options are shown in Figure 4-3.

.

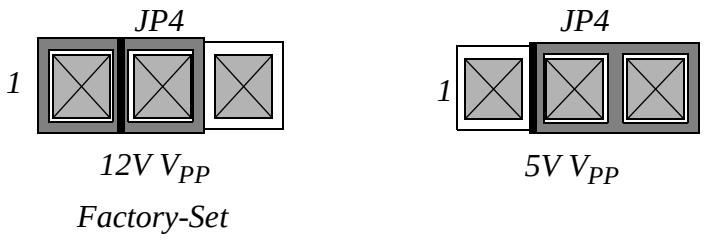



**Figure 4-3. JP13 $V_{PP}$ Source Selection**

### 4.1.11 GND Bridges

The MPC8272ADS has seven GND bridges that are meant to assist general measurements and the logic-analyzer connection.

**Warning**

When connecting to a GND bridge, use only insulated GND clips. Un-insulated clips may cause short-circuits that touch hot points around them. Failure to use insulated GND clips can permanently damage the MPC8272ADS.

### 4.1.12 Power O.K. Indicator—LD13

The green Power O.K. LED indicator lights if the ATX power supply is generating all the voltages.

### 4.1.13 12V Indicator—LD14

The green 12-V led LD14 indicates the presence of the +12-V supply on the board.

### 4.1.14 UTOPIA Multi PHY Indicator—LD28

The green multi PHY led LD28, indicates that the UTOPIA is in multiPHY mode. When this indicator is off, the UTOPIA is in single PHY mode.

### 4.1.15 5-V Indicator—LD15

The green 5V led LD15 indicates the presence of the +5 V supply on the board.

### 4.1.16 3.3-V Indicator—LD16

The green 3.3V led LD16 indicates the presence of the +3.3 V supply on the board.

### 4.1.17 USB Power Indicator—LD5

The green USB Power led LD5 indicates the presence of 5 V in the USB cable.

### 4.1.18 -12V Indicator—LD17

The green -12V led—LD17, indicates the presence of the -12V supply on the board.

### 4.1.19 RUN Indicator—LD18

When the green RUN led - LD18 is lit, it indicates that the MPC8272 is performing cycles on the PPC bus. When dark, the MPC8272 is either running internally or stuck.

### 4.1.20 ATM ON Indicator—LD19

When the yellow ATM ON led is lit, it indicates that the ATM-UNI transceiver PM5384 is enabled for communication. When it is dark, the ATM-UNI transceiver is disconnected from the MPC8272, enabling the use of its associated FCC1 pins off-board by using the expansion connectors or for Fast Ethernet.

ATM ON led is controlled by BCSR1.

### 4.1.21 Fast Ethernet Port 1 Enabled—LD20

When the yellow ETH1 ON led is lit, the fast Ethernet port 1 transceiver DM9161 is connected to FCC1. When it is dark, the DM9161 is in power-down mode and disconnected from FCC1, enabling the use of its associated FCC1 pins off-board by using the expansion connectors. BCSR1 controls the state of LD20.

### 4.1.22 Fast Ethernet Port 2 Enabled—LD22

When the yellow ETH2 ON led is lit, the fast Ethernet port 2 transceiver DM9161 is connected to FCC2. When it is dark, the DM9161 is in power-down mode and disconnected from FCC2, enabling the use of its associated FCC2 pins off-board by using the expansion connectors. BCSR1 controls the state of LD22.

### 4.1.23 RS232 Port 1 ON—LD23

When the yellow RS232 Port 1 ON led is lit, the RS232 transceiver is connected to P13A (upper DB9 connector), is active, and communication using that medium is allowed. When it is darkened, the transceiver is in shutdown mode and its associated SCC1 pins may be used off-board by using the expansion connectors.

### 4.1.24 Fast Ethernet Port 1 Full Duplex Indicator—LD1

When the Dm9161 on FCC1 is enabled and is in full duplex operation mode, the red led - LD1 lights.

### 4.1.25 RS232 Port 2 ON—LD24

When the yellow RS232 Port 2 ON led is lit, the RS232 transceiver is connected to P13B (lower DB9 connector), is active, and communication using that medium is allowed. When

darkened, it designates that the transceiver is in shutdown mode and its associated SCC4 pins may be used off-board by using the expansion connectors.

### 4.1.26 Fast Ethernet Port 1 100Base-Tx Indicator—LD2

When the DM9161 on FCC1 is enabled and is in 100 Mbps operation mode, the green led LD2 lights.

### 4.1.27 Ethernet Port 1 Tx/Rx Indicator—LD3

The green Ethernet transmit/receive LED indicator blinks whenever the Dm9161 on FCC1 is transmitting or receiving data by using the 10/100-Base-T port.

### 4.1.28 General Purpose Led 2 Indicator—LD26

This general-purpose red LED is user-controlled by BCSR0.

### 4.1.29 Ethernet Port 1 LINK Indicator—LD4

The yellow Ethernet twisted-pair link integrity LED indicator (LINK) lights to indicate good link integrity on the 10/100-Base-T port. When the link integrity fails, LD4 is off.

### 4.1.30 Fast Ethernet Port 2 Full Duplex Indicator—LD6

When the Dm9161 on FCC2 is enabled and is in full duplex operation mode, the red led LD6 lights.

### 4.1.31 General Purpose Led 1 Indicator—LD25

This general-purpose green LED is user-controlled by BCSR0.

### 4.1.32 Fast Ethernet Port 2 100Base-Tx Indicator—LD7

When the DM9161 on FCC2 is enabled and is in 100 Mbps operation mode, the green led LD7 lights.

### 4.1.33 USB Enabled Indicator—LD21

The yellow USB enable LED indicates that the USB transceiver is connected to the MPC8272.

### 4.1.34 Ethernet Port 2 LINK Indicator—LD9

The yellow Ethernet twisted pair link integrity LED indicator (LINK) lights to indicate good link integrity on the 10/100-Base-T port. When the link integrity fails, LD9 is off.

### 4.1.35 Ethernet Port 2 Tx/Rx Indicator—LD8

The green Ethernet transmit/receive LED indicator blinks whenever the Dm9161 on FCC2 is transmitting or receiving data using the 10/100-Base-T port.

### 4.1.36 $V_{DDL}$ Indication—LD27

The green $V_{DDL}$ indicator led LD27 is lit to indicate a $V_{DDL}$ power activity. Because $V_{DDL}$ level may vary, the LD27 illumination level also varies accordingly.

### 4.1.37 Parallel Port connection—LD10

The green parallel port connection LED indicates that the board is connected directly to the PC parallel port and the COP/JTAG connector (P21) is irrelevant.

### 4.1.38 External Debugger Connection Indicator—LD11

The green external debugger connection LED indicates that a command converter can be connected to the COP/JTAG connector (P21).

# Chapter 5
# Module Design

This chapter provides information about the functionality and design details of the various modules that constitute the MPC8272ADS.

## 5.1 Reset and Reset Configuration

The following are reset sources on the MPC8272ADS:

- Power-on reset
- Manual hard reset
- Manual soft reset
- PCI bus reset
- MPC8272 internal sources. For more information, see the *MPC8272 PowerQUICC II™ Family Reference Manual*.

### 5.1.1 Power-On Reset

The power-on reset to the MPC8272 initializes the processor state after power-on. A dedicated logic using Seiko S-80728AN-DR-T1, which is a voltage detector of 2.8V +/- 2.4%, asserts $\overline{\text{PORESET}}$ input to the PowerQUICC II for a period of ~2.5sec. This time period is sufficient to cover the $V_{DDL}$ stabilization also, powered by a different voltage regulator. It is assumed that the stabilization time for both linear regulators (see Section 7.1, “Power Supply”) are about the same. Furthermore, power-on reset may be generated manually by an on-board dedicated push-button (SW1). Power-on reset can also be generated by the JTAG logic, which is integrated with BCSR.

### 5.1.2 Power-On Reset Configuration

At the end of Power - On reset sequence, MODCK(1:3) are sampled by the MPC8272 to configure the various clock modes of the MPC8272 (core, cpm, bus, PCI...). Selection between the MODCK(1:3) combination options is done by means of dip-switches on the mother board while PCI_MODCKH(0:3) are obtained from the relevant dedicated dip-switches.

The configuration master is determined upon the rising edge of $\overline{\text{PORST}}$, according to the state of $\overline{\text{RSTCONF}}$ signal, driven low on this board, to set the MPC8272 as a configuration master.

After power-on reset negates, the hard-reset sequence starts, during which many other different options are configured. Among these options are additional clock configuration bits PCI_MODCKH(0:3), the most significant bits of the MODCK field, which determine additional options for the clock generator. Although these bits are sampled whenever the hard-reset sequence is entered, they are influential only once, after power-on reset. If a hard reset sequence is entered later, MODCKH(0:3), although sampled, are 'don’t care'.

The PCI_MODCK signal, which is sampled concurrently with the PCI_MODCK(0:3) pins, determines the PCI bus clock frequency (see Section 5.2.2, “PCI Clock”). When set high, it divides the PCI bus frequency by two. When reset low, the PCI bus frequency is as determined by the MODCK(1:3) and PCI_MODCKH(0:3) signals.

### 5.1.3 Hard Reset

Hard reset may be generated on the MPC8272ADS by the following sources:

- COP/JTAG port
- Manual hard reset
- MPC8272’s internal sources

Hard-reset, when generated, causes the MPC8272 to reset all its internal hardware except for PLL logic, re-acquires the Hard-reset configuration from its current source, and jumps to the Reset vector in the exception table. Since hard-reset resets also the refresh logic for dynamic RAMs, their content is lost as well.

$\overline{\text{HRESET}}$ when asserted, is extended internally by the MPC8272 for additional 512 bus clock cycles at the end of which the MPC8272 waits for 16 bus clock cycles and then re-checks the state of the $\overline{\text{HRESET}}$ line.

$\overline{\text{HRESET}}$ is an open-drain signal and must be driven with an open-drain gate by whichever external source is driving it. Otherwise, contention occurs over that line, which might cause permanent damage to either board logic or to the MPC8272 itself.

#### 5.1.3.1 COP/JTAG Port Hard Reset

To provide convenient hard-reset capability for a COP/JTAG controller, $\overline{\text{HRESET}}$ line appears at the COP/JTAG port connector. The COP/JTAG controller may directly generate hard-reset by asserting (low) this line.

### 5.1.3.2 Manual Hard Reset

To allow runtime hard-reset, when the COP controller is disconnected from the MPC8272ADS, and to support resident debuggers, manual hard is facilitated. Depressing both soft reset (SW7) and ABORT (SW6) buttons asserts the $\overline{\text{HRESET}}$ pin of the MPC8272, generating a hard reset sequence.

Because the $\overline{\text{HRESET}}$ line may be driven internally by the MPC8272, it must be driven to the MPC8272 with an open-drain gate. If off-board hardware connected to the MPC8272ADS is to drive $\overline{\text{HRESET}}$ line, it should do so with an open-drain gate, to avoid contention over this line.

When hard reset is generated, the MPC8272 is reset in a destructive manner, that is, the hard reset configuration is re-sampled and all registers (except for the PLL's) are reset, including memory controller registers. This reset causes loss of dynamic memory contents.

To save on board's real-estate, this button is not a dedicated one, but is shared with the soft-reset button and the ABORT button. When both are depressed, hard reset is generated.

### 5.1.3.3 Internal Sources Hard Reset

The MPC8272 has internal sources that generate hard reset. Among these sources are the following:

- Loss of lock reset. When one of the PLLs (core, CPM), is out of lock, hard-reset is generated.
- Check-stop reset. When the core enters a check-stop state from some reason, a hard reset may be generated, depending on the CSRE bit in the RMR.
- Bus monitor reset. When the bus monitor is enabled and a bus cycle is not terminated, hard-reset is generated.
- S/W Watch Dog Reset. When the S/W watch-dog is enabled, and application software fails to perform its reset routine, it generates hard reset.
- COP/JTAG Reset (Internal). Hard reset may be forced by driving the $\overline{\text{HRESET}}$ line using the external pin's scan chain. Not useful for run time.

The MPC8272 asserts a reset line HARD or SOFT for a period 512 clock cycles after a reset source is identified. A hard reset sequence is followed by a soft reset sequence.

### 5.1.3.4 Hard Reset Configuration

When hard reset is applied to the MPC8272 (externally as well as internally), it samples the hard reset configuration word. This configuration may be taken from an internal default, in case RSTCONF is negated during HRESET asserted or taken from the Flash /E2PROM/BCSR (MS 8 bits of the data bus) (in general, from any device residing on CS0) in case RSTCONF signal is asserted along with HRESET. The default configuration word

can be taken from the E2PROM/BCSR in case the Flash has been tampered with. The selection between the BCSR, FLASH and the E2PROM as the source of the default configuration word is determined by a dedicated dip-switch and a jumper.

During hard reset sequence, the configuration master[1] reads the Flash (or $E^2$PROM or BCSR) memory at addresses 0, 8, 0x18, 0x20,... a byte each time, to assemble the 32 bit configuration word. A total of 64 bytes of data is read from D(0:7) to acquire 8 full configuration words for system that may have up to 8 MPC8272 chips.

The configuration word for a single[2] MPC8272 is stored in the Flash memory SIMM, in the $E^2$PROM or as default in the BCSR, while the other seven words are not initialized, as there are no additional MPC8272 on the MPC8272ADS. The default configuration word is shown in Table 5-1 for the FLASH and in Table 5-2 for the $E^2$PROM. PCI module configuration is 256 Bytes long and should start at address 0x100.

The two possible configuration words are the following:

- FLASH/BCSR is the boot device. $\overline{CS0}$ is assigned to the FLASH and $\overline{CS4}$ is assigned to the $E^2$PROM.
- $E^2$PROM is the boot device. $\overline{CS0}$ is assigned to the $E^2$PROM and $\overline{CS4}$ is assigned to the FLASH.

**Table 5-1. BCSR/FLASH Hard Reset Configuration Word**

| Field | Data Bus Bits | Prog Value [Bin] | Implication | Offset In Flash [Hex] | Value [Hex] |
|---|---|---|---|---|---|
| ERB | 0 | '0' | Internal Arbitration Selected. | 0 | 0C |
| EXMC | 1 | '0' | Internal Memory Controller. $\overline{CS0}$ active at system boot. | | |
| CDIS | 2 | '0' | Core Enabled. | | |
| EBM | 3 | '0' | '0' - Single PowerQUICC II Mode | | |
| BPS | 4:5 | 11 | 32 Bit Boot Port Size | | |
| CIP | 6 | '0' | Sets Core Initial Prefix MSR[IP]=1, so that system exception table is placed at address 0xFFF00100 regardless of FLASH memory size | | |
| ISPS | 7 | '0' | 64-bit internal space for external master accesses. In fact don't care on this board since external master is not supported. | | |

[1]In general, The MPC8272 for which $\overline{RSTCONF}$ is asserted along with $\overline{PORST}$ asserted or in particular, the MPC8272 residing on the MPC8272ADS.

[2]Although the MPC8272 as configuration master reads 8 configuration words, only the 1'st configuration word is influential.

## Table 5-1. BCSR/FLASH Hard Reset Configuration Word (continued)

| Field | Data Bus Bits | Prog Value [Bin] | Implication | Offset In Flash [Hex] | Value [Hex] |
|---|---|---|---|---|---|
| L2CPC | 8:9 | ‘01’ | $\overline{CI}$/BADDR(29)/$\overline{IRQ2}$ selected as $\overline{IRQ2}$<br>$\overline{WT}$/BADDR(30)/$\overline{IRQ3}$ selected as $\overline{IRQ3}$<br>$\overline{L2_HIT}$/$\overline{IRQ4}$ selected as $\overline{IRQ4}$<br>$\overline{CPU_BG}$/BADDR(31)/$\overline{IRQ5}$ as $\overline{IRQ5}$) | 8 | 72 |
| DPPC | 10:11 | ‘11’ | Data Parity Pin configuration as:<br>DP0 as EXT_BR2<br>DP1 as EXT_BG2<br>DP2 as EXT_DBG2<br>DP3 as EXT_BR3<br>DP4 as EXT_BG3<br>DP5 as $\overline{EXT_DBG3}$<br>DP6 as $\overline{IRQ6}$<br>DP7 as $\overline{IRQ7}$ | | |
| Reserved | 12 | ’0’ | Reserved. | | |
| ISB | 13:15 | ’010’ | IMMR initial value 0x0F000000, i.e., the internal space resides initially at this address. | | |
| BMS | 16 | ’0’ | Boot memory (Flash) at 0xFE000000. | 10 | 36 |
| BBD | 17 | ’0’ | $\overline{ABB}$/$\overline{IRQ2}$ pin is $\overline{ABB}$<br>$\overline{DBB}$/$\overline{IRQ3}$ pin is $\overline{DBB}$ | | |
| MMR | 18:19 | ’11’ | ‘11’ - Mask Masters Requests. Boot Master is MPC8272s 60x. | | |
| LBPC | 20:21 | ’01’ | ‘01’ - Local Bus pins function as PCI bus. | | |
| APPC | 22:23 | ’10’ | MODCK1/AP(1)/TC(0) functions as BKSEL0<br>MODCK2/AP(2)/TC(1) functions as BKSEL1<br>MODCK3/AP(3)/TC(2) functions as BKSEL2<br>IRQ7~/APE~ functions as IRQ7~<br>CS11~/AP(0) functions as CS11~ | | |
| CS10PC | 24:25 | ’01’ | CS10~/BCTL1/DBG_DIS~ functions as BCTL1 | 18 | 5A |
| ALD_EN | 26 | ’0’ | PCI Auto Load Enable. When high, PCI Bridge Configuration is done automatically from the FLASH/$E^2$PROM (CPM is configuration master - PPC core should be disabled) right after the Hard Configuration Word. When low, the PPC Core should configure the PCI Bridge. | | |
| PCI_MODCK | 27 | ’1’ | Determines PCI clock settings as set by PCI_MODCKH:<br>‘0’ - PCI clock set by PCI_MODCKH<br>‘1’ - PCI clock is divided according to PCI_MODCKH | | |
| MODCK_HI [1] | 28:31 | ‘1010’ | Determines the Core’s frequency out of power-up reset. | | |

[1] Applies only ONCE after power-up reset.

## Table 5-2. E$^2$PROM Hard Reset Configuration Word

| Field | Data Bus Bits | Prog Value [Bin] | Implication | Offset In Flash [Hex] | Value [Hex] |
|---|---|---|---|---|---|
| ERB | 0 | '0' | Internal Arbitration Selected. | 0 | 04 |
| EXMC | 1 | '0' | Internal Memory Controller. $\overline{CS0}$ active at system boot. | | |
| CDIS | 2 | '0' | Core Enabled. | | |
| EBM | 3 | '0' | '0' - Single PowerQUICC II Mode | | |
| BPS | 4:5 | '01' | 8 Bit Boot Port Size | | |
| CIP | 6 | '0' | Sets Core Initial Prefix MSR[IP]=1, so that system exception table is placed at address 0xFFF00100 regardless of FLASH memory size | | |
| ISPS | 7 | '0' | 64 bit internal space for external master accesses. In fact don't care on this board since external master is not supported. | | |
| L2CPC | 8:9 | '01' | $\overline{CI}$/BADDR(29)/$\overline{IRQ2}$ selected as $\overline{IRQ2}$<br>$\overline{WT}$/BADDR(30)/$\overline{IRQ3}$ selected as $\overline{IRQ3}$<br>$\overline{L2_HIT}$/$\overline{IRQ4}$ selected as $\overline{IRQ4}$<br>$\overline{CPU_BG}$/BADDR(31)/$\overline{IRQ5}$ as $\overline{IRQ5}$) | 8 | 72 |
| DPPC | 10:11 | '11' | Data Parity Pin configuration as:<br>DP0 as EXT_BR2<br>DP1 as EXT_BG2<br>DP2 as EXT_DBG2<br>DP3 as EXT_BR3<br>DP4 as EXT_BG3<br>DP5 as EXT_DBG3<br>DP6 as $\overline{IRQ6}$<br>DP7 as $\overline{IRQ7}$ | | |
| Reserved | 12 | '0' | Reserved. | | |
| ISB | 13:15 | '010' | IMMR initial value 0x0F000000, i.e., the internal space resides initially at this address. | | |
| BMS | 16 | '0' | Boot memory (E$^2$PROM) at 0xFE000000. | 10 | 36 |
| BBD | 17 | '0' | $\overline{ABB}$/$\overline{IRQ2}$ pin is $\overline{ABB}$<br>$\overline{DBB}$/$\overline{IRQ3}$ pin is $\overline{DBB}$ | | |
| MMR | 18:19 | '11' | Mask Masters Requests. Boot Master is 60x. | | |
| LBPC | 20:21 | '01' | Local Bus pins function as PCI bus. | | |
| APPC | 22:23 | '10' | MODCK1/AP(1)/TC(0) functions as BKSEL0<br>MODCK2/AP(2)/TC(1) functions as BKSEL1<br>MODCK3/AP(3)/TC(2) functions as BKSEL2<br>IRQ7~/APE~ functions as IRQ7~<br>CS11~/AP(0) functions as CS11~ | | |

**Table 5-2. E$^2$PROM Hard Reset Configuration Word (continued)**

| Field | Data Bus Bits | Prog Value [Bin] | Implication | Offset In Flash [Hex] | Value [Hex] |
|---|---|---|---|---|---|
| CS10PC | 24:25 | ’01’ | CS10~/BCTL1/DBG_DIS~ functions as BCTL1 | 18 | 5A |
| ALD_EN | 26 | ’0’ | PCI Auto Load Enable. When high, PCI Bridge Configuration is done automatically from the FLASH/E$^2$PROM (CPM is configuration source - PPC core should be disabled) right after the Hard Configuration Word. When low, the PPC Core should configure the PCI Bridge. | | |
| PCI_MODCK | 27 | ’1’ | Determines PCI clock settings as set by PCI_MODCKH:<br>‘0’ - PCI clock set by PCI_MODCKH<br>‘1’ - PCI clock is divided according to PCI_MODCKH | | |
| MODCK_HI [1] | 28:31 | ‘1010’ | Determines the Core’s frequency out of power-up reset. | | |

1 Applies only ONCE after power-up reset.

The PCI configuration registers which are set at hard reset sequence are shown in Figure 5-1

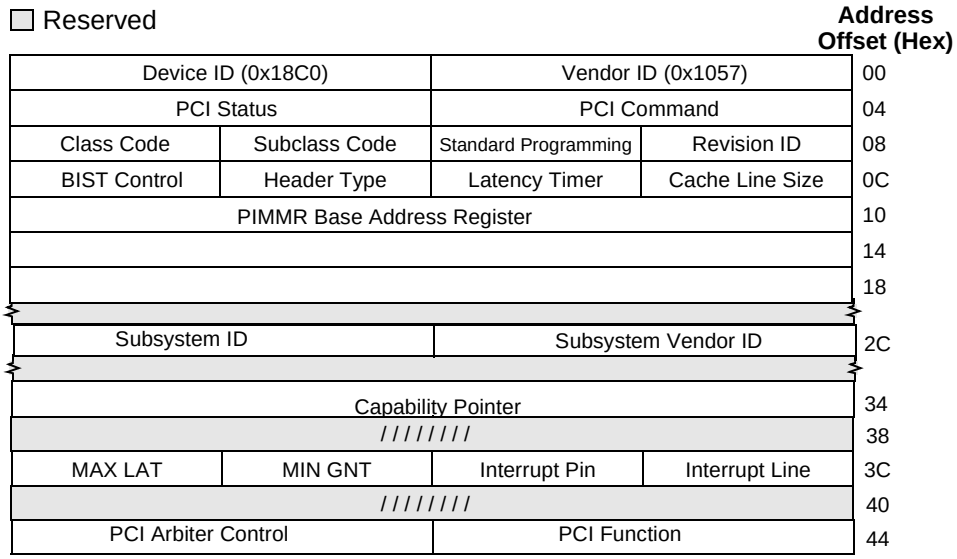



**Figure 5-1. PCI Host Configuration Registers**

## 5.1.4 Soft Reset

Soft reset may be generated on the board from the following sources:

- COP/JTAG port
- Manual soft reset

- Internal MPC8272 source.

A soft reset, when generated, causes the MPC8272 to reset its internal logic while keeping its hard-reset configuration and memory controller setup and then jumping to the Reset vector in the exception table. Since soft-reset does not reset the refresh logic for dynamic RAMs, their contents is preserved.

$\overline{\text{SRESET}}$ when asserted, is extended internally by the MPC8272 for an additional 512 bus clock cycles at the end of which, the MPC8272 waits for 16 bus clock cycles and then re-checks the state of the $\overline{\text{SRESET}}$ line.

$\overline{\text{SRESET}}$ is an open-drain signal and must be driven with an open-drain gate by every external source driving it. Otherwise, contention occurs over that line, which might cause permanent damage to either the boards' logic or to the MPC8272 itself.

#### 5.1.4.1 COP/JTAG Port Soft Reset

To provide convenient soft-reset capability for a COP/JTAG controller, $\overline{\text{SRESET}}$ line appears at the COP/JTAG port connector - P21. The COP/JTAG controller may directly generate Soft-reset by asserting (low) this line.

#### 5.1.4.2 Manual Soft Reset

To allow run-time Soft-reset, when the COP controller is disconnected from the MPC8272ADS and to support resident debuggers, a Soft Reset push-button is provided. When the Soft Reset push-button is depressed, the $\overline{\text{SRESET}}$ line is asserted to the MPC8272, generating a Soft Reset sequence.

Since the $\overline{\text{SRESET}}$ line may be driven internally by the MPC8272, it must be driven by an open-drain gate, to avoid contention over that line. If off-board H/W connected to the MPC8272ADS is to drive $\overline{\text{SRESET}}$ line, then, it should do so with an open-drain gate, this, to avoid contention over this line.

#### 5.1.4.3 Internal Sources Soft Reset

The only internal Soft-reset source is the COP/JTAG soft-reset, which may be generated using Public JTAG instructions to shift active-value ('0') to the $\overline{\text{SRESET}}$ pin via the boundary scan chain. This is not useful for run time.

### 5.1.5 PCI Bus Reset

The PCI Module in the MPC8272 can generate a reset signal dedicated for MPC8272 devices which reside on the PCI bus. This is a reset to the PCI bus which is initiated by the PCI bus Host - the MPC8272 on this board. This reset can also be initiated by a Soft PCI Reset by setting a dedicated bit in a PCI control register (consult the MPC8272 User Manual for details).

## 5.1.6 Local Interrupter

The following external interrupts are applied to the MPC8272 using its interrupt controller:

- ABORT (NMI)
- ATM UNI interrupt
- Fast Ethernet PHY interrupt
- PCI interrupt

### 5.1.6.1 ABORT Interrupt

The ABORT (NMI) is generated by a push-button. When this button is depressed, the $\overline{\text{IRQ0}}$ input to the MPC8272 is asserted. The purpose of this type of interrupt is to support the use of resident debugger if any is made available to the board. This interrupt is enabled by setting the MSR[EE] bit.

To support external (off-board) generation of an NMI, the $\overline{\text{IRQ0}}$ line is driven by an open-drain gate to allow for an external h/w, to drive this line also. If an external hardware indeed does so, it is compulsory that $\overline{\text{IRQ0}}$ is driven by an open-drain (or open-collector) gate.

### 5.1.6.2 ATM UNI Interrupt

To support ATM UNI (User Network I/F) event report by means of interrupt, the interrupt output of the UNI (INTB) is connected to $\overline{\text{IRQ5}}$ line of the MPC8272. This $\overline{\text{IRQ5}}$ input is shared with the Fast Ethernet PHY Interrupt. Since INTB of the UNI is an open-drain output, it is possible to connect additional (on and off-board) interrupt requesters on the same $\overline{\text{IRQ5}}$, provided that they drive $\overline{\text{IRQ5}}$ with open-drain gate as well. When an interrupt request appears in $\overline{\text{IRQ5}}$, it is necessary to check the source of the interrupt whether it's the ATM UNI or the Fast Ethernet PHY.

## 5.1.7 Fast Ethernet PHY Interrupt

To support the two fast Ethernet transceivers event reports by means of interrupt, the interrupt outputs of the DM9161 are connected to $\overline{\text{IRQ5}}$ line of the PowerQUICC II. This $\overline{\text{IRQ5}}$ input is shared with the ATM UNI Interrupt.

## 5.1.8 PCI Interrupt

Each PCI slot can generate up to four interrupts for a total of 12 (3 slot x 4 interrupts each). Each PCI expansion board can generate an interrupt at any given time. Since there is only one interrupt input available in the MPC8272, an interrupt controller is used. The interrupt controller receives all the possible interrupts from the PCI slots and generate one interrupt ($\overline{\text{IRQ2}}$) to the MPC8272.

A simple generic interrupt controller is implemented using a CPLD device. The interrupt controller is implemented as an interrupt register and an interrupt mask register. The interrupt controller has its' own dedicated chip-select line ($\overline{CS3}$). A simple priority scheme is devised to prioritize the interrupts from different slots. The PCI IRQ routing are according to Figure 5-2.

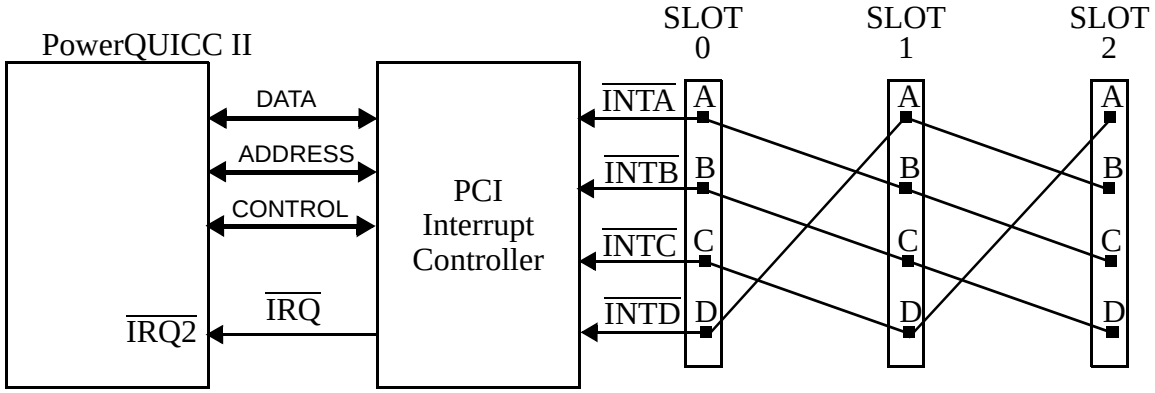



**Figure 5-2. PCI Interrupt Routing Scheme**

An interrupt request in any of the $\overline{INTx}$ lines, sets 3 interrupt bits in the PCI interrupt register (if not masked in the interrupt mask register) because every $\overline{INTx}$ line has three possible interrupt sources. It is up to the user to implement a polling process to verify the real interrupt source (by polling the interrupt pending bit in the PCI device) and clear the other two. The PCI interrupt register can be read at any time and accessed at offset 0x0 from $\overline{CS3}$ base address. The description of the PCI interrupt register is in Table 5-3.

**Table 5-3. PCI Interrupt Register Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 | PCI0_INTA | ***PCI Slot 0 $\overline{INTA}$***. PCI Slot 0 Interrupt A:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 1 | PCI0_INTB | ***PCI Slot 0 $\overline{INTB}$***. PCI Slot 0 Interrupt B:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 2 | PCI0_INTC | ***PCI Slot 0 $\overline{INTC}$***. PCI Slot 0 Interrupt C:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 3 | PCI0_INTD | ***PCI Slot 0 $\overline{INTD}$***. PCI Slot 0 Interrupt D:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |

**Table 5-3. PCI Interrupt Register Description (continued)**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 4 | PCI1_INTA | ***PCI Slot 1 $\overline{INTA}$***. PCI Slot 1 Interrupt A:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 5 | PCI1_INTB | ***PCI Slot 1 $\overline{INTB}$***. PCI Slot 1 Interrupt B:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 6 | PCI1_INTC | ***PCI Slot 1 $\overline{INTC}$***. PCI Slot 1 Interrupt C:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 7 | PCI1_INTD | ***PCI Slot 1 $\overline{INTD}$***. PCI Slot 1 Interrupt D:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 8 | PCI2_INTA | ***PCI Slot 2 $\overline{INTA}$***. PCI Slot 2 Interrupt A:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 9 | PCI2_INTB | ***PCI Slot 2 $\overline{INTB}$***. PCI Slot 2 Interrupt B:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 10 | PCI2_INTC | ***PCI Slot 2 $\overline{INTC}$***. PCI Slot 2 Interrupt C:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 11 | PCI2_INTD | ***PCI Slot 2 $\overline{INTD}$***. PCI Slot 2 Interrupt D:<br>'0' - no interrupt was requested<br>'1' - an interrupt was requested and waiting to be handled | 0 | R |
| 12-31 | Reserved | Un-implemented | | R/W |

Also available is an interrupt mask register that provides the user with the option to mask any of the possible PCI interrupt sources. It can be read or written at any time and accessed at offset 0x4 from $\overline{CS3}$ base address.The description of the PCI interrupt mask register is in Table 5-4.

## Table 5-4. PCI Interrupt Mask Register Description

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 | MPCI0_INTA | ***Mask PCI Slot 0 $\overline{INTA}$***. Mask PCI Slot 0 Interrupt A:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 1 | MPCI0_INTB | ***Mask PCI Slot 0 $\overline{INTB}$***. Mask PCI Slot 0 Interrupt B:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 2 | MPCI0_INTC | ***Mask PCI Slot 0 $\overline{INTC}$***. Mask PCI Slot 0 Interrupt C:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 3 | MPCI0_INTD | ***Mask PCI Slot 0 $\overline{INTD}$***. Mask PCI Slot 0 Interrupt D:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 4 | MPCI1_INTA | ***Mask PCI Slot 1 $\overline{INTA}$***. Mask PCI Slot 1 Interrupt A:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 5 | MPCI1_INTB | ***Mask PCI Slot 1 $\overline{INTB}$***. Mask PCI Slot 1 Interrupt B:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 6 | MPCI1_INTC | ***Mask PCI Slot 1 $\overline{INTC}$***. Mask PCI Slot 1 Interrupt C:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 7 | MPCI1_INTD | ***Mask PCI Slot 1 $\overline{INTD}$***. Mask PCI Slot 1 Interrupt D:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 8 | MPCI2_INTA | ***Mask PCI Slot 2 $\overline{INTA}$***. Mask PCI Slot 2 Interrupt A:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 9 | MPCI2_INTB | ***Mask PCI Slot 2 $\overline{INTB}$***. Mask PCI Slot 2 Interrupt B:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 10 | MPCI2_INTC | ***Mask PCI Slot 2 $\overline{INTC}$***. Mask PCI Slot 2 Interrupt C:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 11 | MPCI2_INTD | ***Mask PCI Slot 2 $\overline{INTD}$***. Mask PCI Slot 2 Interrupt D:<br>'0' - interrupt is available<br>'1' - interrupt is masked | 0 | R/W |
| 12-31 | Reserved | Un-implemented | | R/W |

## 5.2 Clock Generator

The two main clock circuits on board are the following:

- MPC8272 system clock
- PCI clock

### 5.2.1 MPC8272 Clock

The MPC8272 requires a single clock source as the main clock source. All MPC8272 60x bus timings are referenced to the main clock input - CLKIN1. The main clock input is in 1:1 ratio to the bus clock, with internal skew elimination (PLL). Use is done with 100MHz 3.3V clock oscillator, which is connected to a low inter-skew buffer (U10) to split the load between all various clock consumers on both boards.

Special care is taken to isolate and terminate the clock route between the on-board PLL and the MPC8272 to provide a 'clean' clock input for proper operation. The main clock scheme is shown in Figure 5-3.

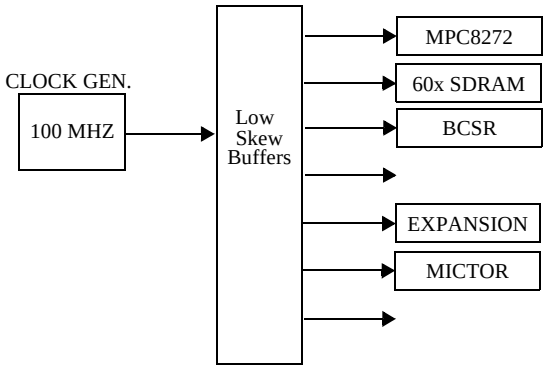



**Figure 5-3. Main Clock Generator Scheme**

### 5.2.2 PCI Clock

The PCI bus clock is derived internally from the main clock input CLKIN1. The generated PCI clock is output from a PCI-dedicated PLL named DLL. That clock output is feeding an on-board low-skew and fast clock distributor that distributes the PCI clock to all on-board PCI devices. One of the outputs is fed back to the PCI clock to the MPC8272 through CLKIN2 input. This clock input is driven to the DLL, which synchronizes the DLL output clock to the CLKIN2 input clock and maintains low skew between the DLL output and CLKIN2 input. All PCI bus timings are referenced to the CLKIN2 input clock. Special care was taken when the board layout was done to keep all copper traces from the clock distributor outputs at the same lengths, including the output that is fed back to CLKIN2. This design is in compliance with the PCI standard to achieve bus synchronization and low skew. The PCI clock scheme is shown in Figure 5-4.

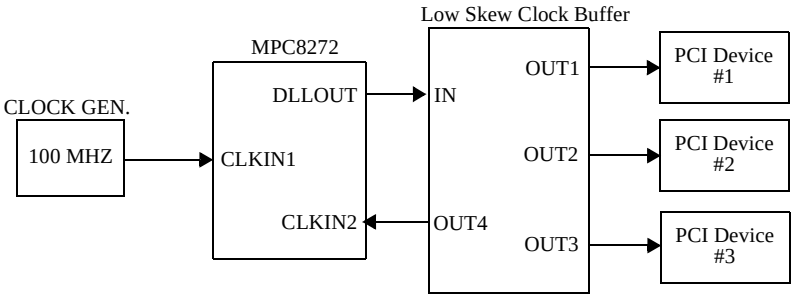



**Figure 5-4. PCI Clock Generator Scheme**

## 5.3 Bus Configuration

The MPC8272on the board is configured in one possible bus mode, which is single MPC8272 mode.

### 5.3.1 Single PowerQUICC II Mode

The MPC8272 is configured in single MPC8272 mode, assuming only one MPC8272 on the 60x bus with no support for external master access. This configuration allows for internal address multiplexing to occur, which makes external address multiplexers redundant and unused, improving SDRAM performance.

## 5.4 Buffering

To achieve optimal performance, it is necessary to reduce the capacitive load over the 60X bus as much as possible. The slower devices on the bus, that is, the Flash SIMM, $E^2$PROM, ATM UNI M/P interface, PCI interrupt controller and the BCSR are buffered, while the SDRAM is not buffered from the 60X bus.

Latches are provided over address and strobe (when necessary) lines while transceivers are provided for data. Use is done with 74ALVT buffers (by Philips) which are 3.3 V operated and 5-V tolerant[1], and provide bus hold to reduce pull-up/pull-down resistors count (as the MPC8272 requires). This type of buffer reduces noise on board due to reduced transition amplitude.

To reduce noise and reflections further, serial damping resistors are placed over SDRAM address and all MPC8272 strobe lines.

The data transceivers are open only if there is an access to a valid[2] buffered board address or during hard reset configuration[3]. Data conflicts are avoided in case an unbuffered

[1]Required for Flash, $E^2$PROM, Interrupt Controller and BCSR

[2]An address which is covered in a Chip-Select region, that controls a buffered device.

[3]To allow a configuration word stored in the Flash/$E^2$PROM memory to become active.

memory read or off-board memory is read, provided that it is not mapped to an address valid on board. It is the user's responsibility to avoid such errors.

## 5.5 Chip-Select Generator

The memory controller of the MPC8272 is used as a chip-select generator to access on-board (and off-board) memories, saving boards' area, reducing cost and power consumption, and increasing flexibility. To enhance off-board application development, memory modules (including the BCSRx) may be disabled by using BCSR[1] in favor of an external memory connected via the expansion connectors. That way, a CS line may be used off-board via the expansion connectors while its associated local memory is disabled.

When a CS region assigned to a buffered[2] memory is disabled by using BCSR, the local data transceivers are disabled during access to that region, avoiding possible[3] contention over data lines.

The MPC8272 chip-select assignments to the various memories / registers on the MPC8272ADS are shown in Table 5-5.

**Table 5-5. MPC8272ADS Chip Select Assignments**

| Chip Select: | Assignment | Bus | Timing Machine |
|---|---|---|---|
| $\overline{CS0}$ | Flash SIMM / E$^2$PROM [1] | 60X (Buffered) | GPCM |
| $\overline{CS1}$ | BCSR | 60X (Buffered) | GPCM |
| $\overline{CS2}$ | SDRAM | 60X (Main) | SDRAM Machine 1 |
| $\overline{CS3}$ | PCI Interrupt Controller | 60X (Buffered) | GPCM |
| $\overline{CS4}$ | E$^2$PROM / Flash SIMM[a] | 60X (Buffered) | GPCM |
| $\overline{CS5}$ | ATM UNI Microprocessor I/F | 60X (Main) | GPCM |
| $\overline{CS6}$ | Communication Tool M/P Interface $\overline{CS1}$. | 60X (Buffered) | GPCM/UPMx |
| $\overline{CS7}$ | Communication Tool M/P Interface $\overline{CS2}$. | 60X (Buffered) | GPCM/UPMx |

1 Selection is done by a dip-switch.

## 5.6 Synchronous DRAM (60X Bus)

To enhance performance, especially in higher operation frequencies, 64Mbytes of SDRAM are provided on board. The SDRAM is unbuffered from the MPC8272 60X bus. Use is

[1]After the BCSR is removed from the local memory map, there is no way to access it but to re-apply power to the MPC8272ADS.

[2]When an unbuffered CS region is being accessed, buffers do not open anyway.

[3]During read cycles.

done with four MT48LC8M16A2 by Micron or compatibles, which are each 2M X 16bit X 4banks.

The SDRAM's timing is controlled by SDRAM Machine #1 associated with 60X bus, via its assigned chip select lines (See Figure 5-5). The SDRAM Machine supports PBI (Page Bank Interleave) which increases the SDRAM throughput. The SDRAM connection scheme is shown in Figure 5-5.

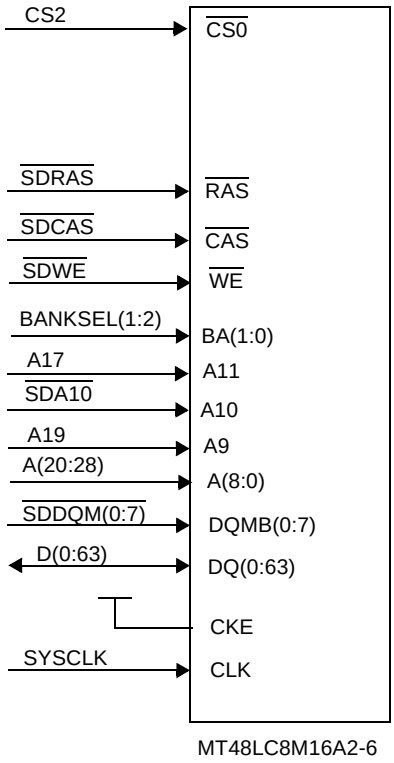



**Figure 5-5. 60x SDRAM Connection Scheme**

## 5.6.1 SDRAM Programming

After power-up, the SDRAM needs to be initialized by means of programming to establish its mode of operation. The SDRAM is programmed according to the following procedure:

1. Issue Precharge-All command.
2. Issue 8 CBR refresh commands.
3. Issue MODE-SET command.

An SDRAM is programmed by issuing a mode register set command. During that command, data is passed to the mode register through the SDRAM's address lines. This command is fully supported by the SDRAM machine of the MPC8272. Before that can occur, the SDRAM machine of the MPC8272 must be initialized.

Mode register programming values are shown in Table 5-6.

**Table 5-6. 100 MHz SDRAM Mode Register Programming**

| SDRAM Address Line [1] | SDRAM Mode Reg Field | Value | Meaning: |
|---|---|---|---|
| A11 (MSB) | Reserved | '0' | |
| A10 | Reserved | '0' | |
| A9 | Opcode | '0' / '1' | 0 - Burst read & burst write (copy-back data cache)<br>1 - Burst read & single write (write-through data cache) |
| A8 | Reserved | '0' | |
| A7 | Reserved | '0' | |
| A6 - A4 | CAS Latency | '011' | Data valid 3 clock cycles after CAS asserted |
| A3 | Burst Type | '0' | Sequential burst |
| A2 - A0 | Burst Length | '010' | 4 Operand burst length |

[1] Actually SDRAMs' A0 is connected to MPC8272s' A28 and so on.

### 5.6.2 SDRAM Refresh

The SDRAM is refreshed using its auto-refresh mode. Using the SDRAM machine one's periodic timer, an auto-refresh command is issued to the SDRAM every 8.2 µsec, so that all 4096 SDRAM rows are refreshed within specified 34 msec, while leaving an interval of ~30 msec of refresh redundancy within that window as a safety measure to cover for possible delays in bus availability for the refresh controller.

## 5.7 Flash Memory SIMM

The MPC8272ADS is provided with 8 Mbytes of 95-nsec Flash memory SIMM, the SM73228XG1JHBGO by Smart Modular Technology. This technology is composed of 4 LH28F016SCT-L95 chips by Sharp, arranged as 2M X 32 in a single bank. Support is given also to 16 Mbytes and 32 Mbytes SIMMs. The Flash SIMM resides on an 80-pin SIMM socket and is buffered from the 60X bus to reduce capacitive load over it.

To minimize use of the MPC8272's chip-select lines, only one chip-select line ($\overline{CS0}$ or $\overline{CS4}$ if the $E^2$PROM is using $\overline{CS0}$) is used to select the Flash as a whole while distributing chip-select lines among the module's internal banks is done by on-board programmable logic, according to the presence-detect lines of the Flash SIMM inserted to the MPC8272ADS.

The access time of the Flash memory provided with the MPC8272ADS is 95 nsec. However, devices with different delay are supported as well. By reading the delay section

of the Flash SIMM presence-detect lines (see Table 5-10), the debugger can establish (via register OR0 in case $\overline{CS0}$ is used or OR4 if $\overline{CS4}$ is used) the correct number of wait-states needed to access the Flash SIMM (considering 100MHz system clock frequency).

The control over the Flash is done with the GPCM and a dedicated $\overline{CS0}$ (or $\overline{CS4}$) region which controls the whole bank. During hard - reset initialization[1], the debugger or any application S/W for that matter, reads the Flash Presence-Detect lines via BCSR and determines how to program registers BR0 & OR0 (or BR4 & OR4), within which the size and the delay of the region are determined.The Flash module may be disabled / enabled at any time by writing '1' /'0' respectively to the $\overline{FlashEn}$ bit in BCSR1. The Flash connection scheme is shown in Figure 5-6.

.

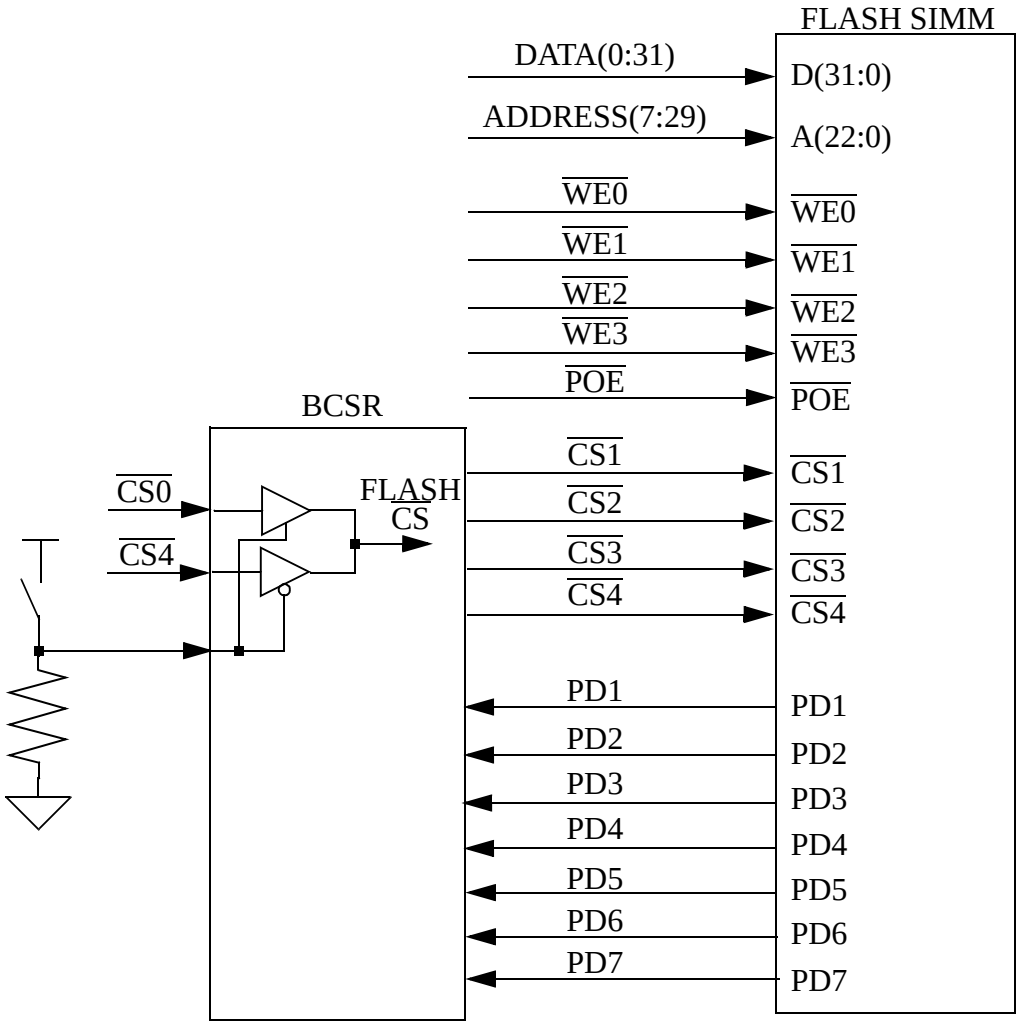



**Figure 5-6. FLASH SIMM Connection Scheme**

[1]i.e., initialization that follow the hard reset sequence at system boot.

As can be seen in Figure 5-6, the FLASH $\overline{CS}$ is distributed to four $\overline{CS}$ signals. The distribution depends on the size of the FLASH module installed - it is read by the BCSR using the PD(1-7) pins.

The Hard-Reset configuration word stored in the FLASH differs from the one stored in the $E^2$PROM in the BPS field which is the Boot Port Size - the $E^2$PROM is 8 bits while the FLASH is 32 bits.

### 5.7.1 Flash Programming Voltage

Support is given to 5V- and 12-V programmable modules. The selection between $V_{PP}$ voltage levels is done with a dedicated jumper. To avoid inadvertent programming or erasure of the Flash, Motorola recommends leaving the jumper open so that no $V_{PP}$ is applied to the Flash SIMM.

## 5.8 $E^2$PROM Memory

The MPC8272ADS is provided with 8KBytes of $E^2$PROM memory in a PLCC package. The $E^2$PROM resides on a socket in case it is desired to replace or re-program a different configuration for the board. The $E^2$PROM is used only for the purpose of supplying the Reset Configuration Word during power-on reset and for storing the PCI configuration data. It is used as a back-up for the Flash memory in case the Flash is not installed or the data it holds is incorrect. As a back-up, it holds the default Hard-Reset configuration word and the default PCI configuration. The Hard-Reset configuration word stored in the $E^2$PROM differs from the one stored in the FLASH in the BPS field which is the Boot Port Size - the $E^2$PROM is 8 bits while the FLASH is 32 bits. It uses a single chip-select, $\overline{CS0}$ or $\overline{CS4}$, which depends on the chip-select used by the Flash. The selection of the chip-select is done by a dip-switch. The $E^2$PROM connection scheme is shown in Figure 5-7.

The device used is ATMEL AT28HC64B, a 5-V Byte alterable $E^2$PROM, 150ns access time with byte-wide JEDEC pinout. Although the device is placed in a socket, it can be programmed onboard. In order to program the device onboard, it must be unlocked - it can be locked to prevent unauthorized alterations of its contents. The lock can be done by hardware or software. The hardware lock is done by write inhibit - the MPC8272 does not assert $\overline{WE}$ during write cycles (set in the BRx register). The software lock is achieved by writing a unique sequence to the device. To unlock, a different unique sequence has to be written

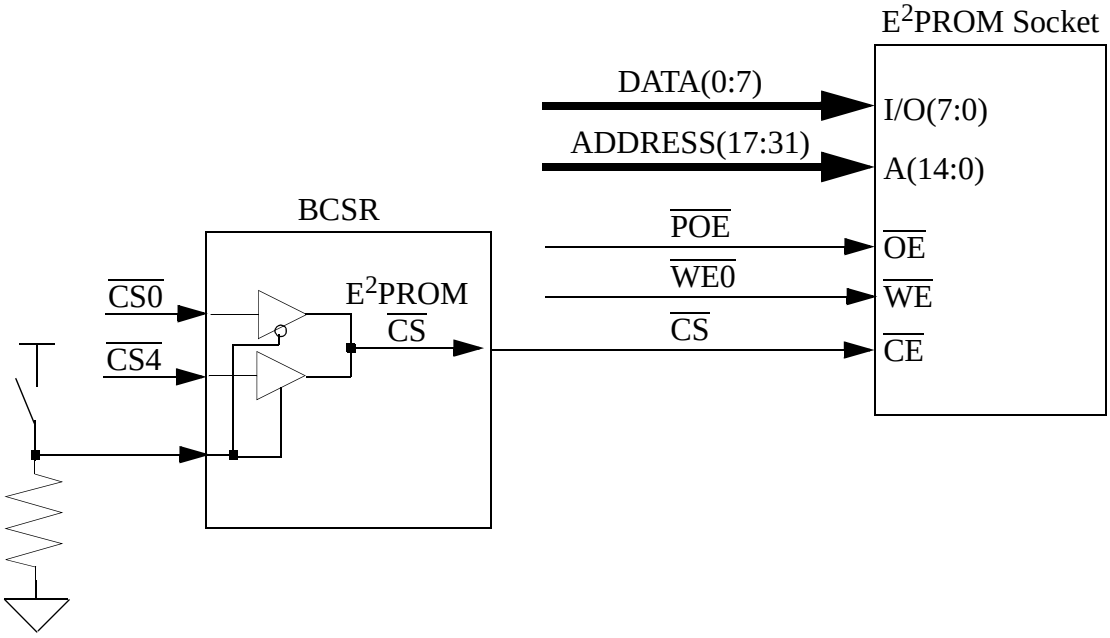



**Figure 5-7. E$^2$PROM Connection Scheme**

Additional address lines are connected to the socket according to the JEDEC format as an option to use E$^2$PROM up to 32 KByte.

## 5.9 PCI Bus

The MPC8272 has a PCI module which enables it to act as an host (Master) or a target. On this board, the MPC8272 serves only as a PCI host - a bridge between the PCI Bus and the PowerPC core.

The MPC8272 PCI Bridge is designed to connect the PowerPC processor and memory system to the PCI system bus, to which I/O components are connected. The PCI Bridge enables the MPC8272 to gluelessly bridge PCI masters and agents to a PowerPC system host. It uses a 32-bit multiplexed, address/data bus that can run from 25MHz up to 66MHz. The interface provides address and data parity with error checking and reporting. It also provides three physical address spaces: 32-bit address memory; 32-bit address I/O; and the PCI configuration space.

The MPC8272 also includes an on-chip arbiter which enables arbitration of up to three PCI masters. Only three PCI slots are supported on the MPC8272ADS because of the Arbiter capacity. Each slot can host either a PCI master or PCI target. The MPC8272 as a Bridge can support more PCI devices but that will require extra slots that can host PCI targets only. Therefore, to avoid dedicated slots for PCI targets, only three slots are implemented.

The PCI bridge is implemented on the MPC8272 Local Bus. Due to PCI Standard restrictions, no other application can reside on the local bus. The PCI bus can operate at frequencies of 25MHz up to 66MHz @ 3.3V only. The 3.3V restriction is due to the MPC8272 which is not 5V compliant. The PCI bus layout is shown in Figure 5-8 Special care was taken when the layout of the MPC8272ADS was done so that the PCI standard recommendations are followed strictly.

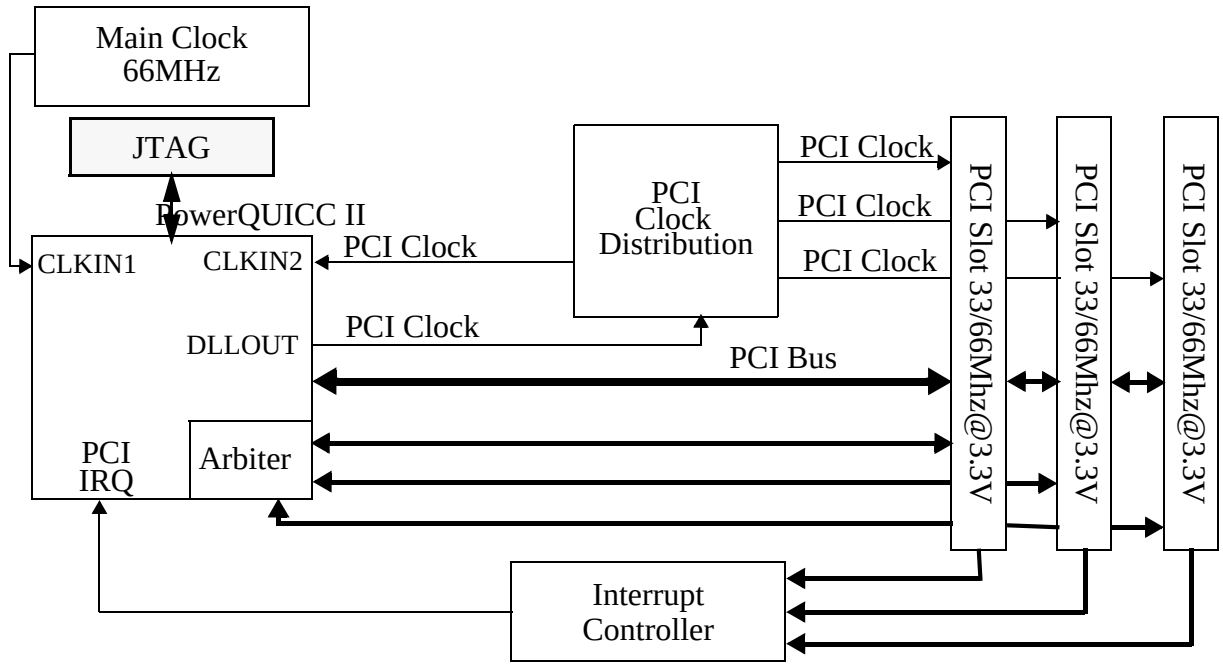



**Figure 5-8. PCI Bus Scheme**

The clock source for the MPC8272 is main clock 100MHz clock oscillator. The PCI Clock is derived internally from the main clock and output at DLLOUT. That clock is then distributed to each PCI device on the bus in a way that they are all synchronized (by keeping all clock traces the same length). The PCI Clock is also fed back to the MPC8272 for synchronization and skew elimination purposes.

An interrupt from any PCI slot is handled by a simple generic Interrupt Controller. Each slot can generate up to four interrupts for a total of twelve interrupts that the controller will support. It will be made of two register mapped in a dedicated $\overline{CS}$ region. One is an Interrupt Register (see Table 5-3) and the second is Interrupt Mask Register (see Table 5-4). A simple priority scheme is devised to allow the controller to support more than one interrupt concurrently.

# 5.10 Communication Ports

The MPC8272ADS has several communication ports, to allow convenient evaluation of the CPM features. Obviously, it is not possible to provide all types of communication interfaces supported by the CPM, but it is made convenient to connect any communication interface devices to the MPC8272 via the CPM Expansion connectors, residing on the edge of the board.

All CPM pins are visible on MICTOR connectors. In order to avoid long routes and stubs, bus muxing devices are used to direct the CPM signals to a communication element on-board or to the expansion connector. A signal that is used on-board, will not be visible in the expansion connector and vise-versa. The control is done by enabling/disabling the communication elements on-board.

The communication ports' interfaces provided on the MPC8272ADS are listed below:

1. 155 Mbps ATM UNI on FCC1 (switchable with Fast ethernet) with Optical interface, using the UTOPIA Level 2 interface - support for 8 bit in multi or single PHY.
2. Two 100/10-Base-T Ports on FCC1 (switchable with ATM) and FCC2 with T.P. interface, MII or RMII controlled.
3. Dual RS232 ports residing on SCC1 & SCC4.
4. USB port, 1.1 USB standard compliant, with speed control (12 or 1.5 Mbps) and mode control (Host or slave).

## 5.10.1 ATM Port

To support the MPC8272s' ATM controller, a 155.52Mbps User Network Interface (UNI) is provided on board, connected to FCC1 of the MPC8272 via UTOPIA I/F. Use is done with PM5384 S/UNI-155-ULTRA by PMC-SIERA. Although these transceivers are capable of supporting 51.84Mbps rate, support is given to 155.52Mbps only. The PHY supports UTOPIA level 2 which means support for 8 bit UTOPIA bus in single or multi PHY mode. The control over the mode of UTOPIA bus connection is done through BCSR3.

The control over the transceiver is done using the microprocessor interface of the transceiver, controlled by the MPC8272 memory controllers' GPCM.

The ATM transceiver may be enabled / disabled at any time by writing '0' /'1' respectively to the $\overline{\text{ATMEN}}$ bit in BCSRx. When $\overline{\text{ATMEN}}$ is negated, ('1') the microprocessor control port is also detached from the MPC8272 and its associated FCC may be used off-board via the expansion connectors.

The ATM transceiver reset input is driven by $\overline{\text{HRESET}}$ signal of the MPC8272, so that the UNI is reset whenever a hard-reset sequence occurs. The UNI may also be reset by either

asserting ATM_RST bit in BCSR1 (see Table 5-8) or by asserting (’1’) the RESET bit in the Master Reset and Identify / Load Meters register via the UNI microprocessor interface.

The UNI transmit and receive clocks are fed with a 19.44 MHz +/- 20 ppm, clock generator, 5 V powered, while the receive and transmit fifos’ clocks of the UTOPIA interface are provided by the MPC8272. The MPC8272 can provide the same clock for both UTOPIA transmit and receive or separate clocks for each, hard-configured[1].

The ATM SAR is connected to the physical medium by an optical interface. Use is done with HP’s HFBR 5805 optical interface, which operates at 1300 nm with up to 2 Km transmission range.

The ATM PHY is connected to $\overline{\text{IRQ5}}$ and generates an interrupt when an appropriate event occurs.

## 5.10.2 100/10 Base T Ports

Two fast Ethernet ports with T.P. (100-Base-TX) I/F is provided on the MPC8272ADS. These ports also support 10 Mbps ethernet (10-Base-T) via the same transceiver - the DM9161 by Davicom.

The DM9161 are connected to FCC1 and FCC2 of the MPC8272 via MII or RMII interface, which is used for both - devices’ control and data path. The initial configuration of the DM9161 on the MPC8272ADS is set by external resistors - 100Base-Tx Full Duplex in MII mode. The selection between MII/RMII for FCC1 and FCC2 is done by jumpers JP5 and JP10 respectively. The DM9161 must be set to MII or RMII while in power-down.

The DM9161 reset input is driven by either asserting the FETH_RST bit in BCSR1 (see Table 5-8) or by asserting a specific bit in an internal register by using MII I/F.

To allow external use of FCC1 and FCC2, their pins appear at the CPM expansion connectors and the ethernet transceiver may be Disabled / Enabled at any time via the MIIs’ MDIO port.

The DM9161 is able to interrupt the PowerQUICC II via $\overline{\text{IRQ5}}$ line. This line is shared also with the CPM expansion connectors. Therefore, any tool that is connected to $\overline{\text{IRQ5}}$, should drive these lines with an Open Drain buffer.

### 5.10.2.1 DM9161 Control

The DM9161 is controlled via the MII management[2] port which is a 2 wire interface: a clock (MDC) and a bidirectional data line (MDIO). This is in fact a bus, i.e., up to 32 devices may reside over it, while the protocol defines a 5-bit slave address field, which is compared against the slave address set to each device by hardware during device reset, according to the levels on some pins. On the board, the slave address is hard-set to b00000

[1] Using resistors.
[2] Also known as MII MDIO port.

for FCC1 and b00011 for FCC2. The MPC8272 interfaces this port using two PI/O pins: PC18 for MDIO and PC19 for MDC. There is no special support within the MPC8272 for the MDIO port and the protocol is implemented in S/W.

The MDIO port may interrupt a host in 2 ways: (a[1]) driving low the MDIO line during IDLE time or (b) using a dedicated interrupt line $\overline{\text{MDINT}}$. This line is connected to the MPC8272's $\overline{\text{IRQ5}}$ line, appearing also at the CPM expansion connectors.

Since $\overline{\text{IRQ5}}$ may also be driven by any tool, connected to the expansion connectors, it should be driven with an Open Drain buffer. $\overline{\text{IRQ5}}$ is pulled-up on the board.

## 5.10.3 RS232 Ports

To assist user's applications and to provide convenient communication channels with both a terminal and a host computer, two identical RS232 ports are provided on the MPC8272ADS, connected to SCC1 and SCC4 ports of the MPC8272. Use is done with MAX3241 transceiver which generates RS232 levels internally using a single 3.3V supply and has a standby mode. When the $\overline{\text{RS232EN1}}$ or $\overline{\text{RS232EN2}}$ bits in BCSR1 are asserted (low), the corresponding transceiver is enabled. When negated, the corresponding transceiver is in standby mode, within which the receiver outputs are tri-stated, enabling the use of the corresponding ports' pins off-board via the expansion connectors.

Nine pins, female D-Type stacked connector is used, configured to be directly (via a flat cable) connected to a standard IBM-PC like RS232 connector.

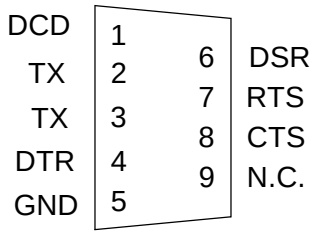



**Figure 5-9. RS232 Serial Ports Connector**

### 5.10.3.1 RS-232 Ports' Signal Description

In the list below, the directions 'I','O', and 'I/O' are relative to the MPC8272ADS board. (i.e.'I' means input to the MPC8272ADS)

- CD (O) - Data Carrier Detect. This line is always asserted by the MPC8272ADS.
- TX (O) - Transmit Data.
- RX (I) - Receive Data.
- DTR (I) - Data Terminal Ready. This signal is used by the software on the MPC8272ADS to detect if a terminal is connected to the board.

[1]Not supported on the board.

- DSR (O) - Data Set Ready. This line is always asserted by the MPC8272ADS.
- RTS (I) - Request To Send. This line is not connected in the MPC8272ADS.
- CTS (O) - Clear To Send. This line is always asserted by the MPC8272ADS.
-

### 5.10.4 USB Port

The USB port resides on the MPC8272ADS and is driven by the USB port of the MPC8272 through SCC3. A dedicated USB transceiver - the PDIUSBP11 by PHILIPS is provided, along with a tri-state buffer, separating this port from the MPC8272's USB port, this to allow Port disable option and off-board use of MPC8272 USB pins. The USB can be configured to Host or Slave mode by using JP8. Two USB connectors are supported - type-A and type-B.

To correctly support the 2 speed modes of the USB when in slave mode, detachable pull-up resistors (3.3V) are provided over D+ and D- lines of the USB, controlled by the USB_SPD bit of BCSR4. When USB_SPD is in low-speed level (low) D- is pulled-up while D+ remains floating. When USB_SPD bit is in high-speed level, D+ is being pulled-up and D- floats.

To correctly support Host mode, two pull down resistors are provided over D+ and D- lines of the USB, controlled by JP8 settings.

Also, 5V power will optionally be provided for the USB type-A connector, controlled by USB_VCC0 in BCSR4. When USB_VCC0 is driven low, a 5V supply will be connected to pin 1 of the USB type-A connector.

### 5.10.5 PC Parallel Port

A new feature to this board is the direct connection to a PC parallel port for the purpose of debugger connection (CodeWarrior). An on-board logic is used to interface to the parallel port and translate the signals to COP/JTAG format. The parallel port support both EPP and SPP modes of the parallel port in a PC. The direct connection eliminates the need for an external command converter. When connected to a PC's parallel port, the parallel port connection has automatic priority over the COP/JTAG connector interface.

## 5.11 Board Control and Status Register - BCSR

Most of the hardware options on the MPC8272ADS are controlled or monitored by the BCSR, which is a 32 bit wide read / write register file. The BCSR is accessed via the MPC8272s' memory controller (see Table 5-5) and in fact includes 8 registers: BCSR0 to BCSR7. Since the minimum block size for a CS region is 32KBytes and only A(27:29) lines are decoded by the BCSR for register selection, BCSR0 - BCSR7 are duplicated inside that region.

The following functions are controlled / monitored by the BCSR:

- PBI
- ATM port control which includes:
  - Transceiver enable / disable
  - Transceiver reset.
  - UTOPIA 8 bit
  - UTOPIA single/multi PHY
- Fast Ethernet Ports Control which includes:
  - Transceiver initial enable
  - Transceiver reset
- RS232 port 1 Enable / disable
- RS232 port 2 Enable / disable
- USB Port Control which includes:
  - Transceiver initial enable
  - USB speed
  - USB power
- Flash Size / delay identification.
- $\overline{CS0}$ assignment after hard reset to FLASH SIMM / $E^2$PROM.
- External (off-board) tools support that include:
  - Tool identification
  - Tool revision
  - Tool status Information
- S/W option identification.
- Board revision code
- Power-on reset by using JTAG (optional)
- PCI cards Present Detect and card type
- Local bus mode

Since part of the MPC8272ADSs' modules are controlled by the BCSR and since they may be disabled in favor of external hardware, the enable signals for these modules are presented at the CPM expansion connectors, so that off- board hardware may be mutually exclusive enabled with on-board modules.

### 5.11.1 BCSR0 Board Control—Status Register 0

The BCSR0 is a control register on the MPC8272ADS. It is accessed at **offset 0** from BCSR base address. It may be read or written at any time[1]. BCSR0 gets its defaults upon Power-On reset. BCSR0 fields are described in Table 5-7.

**Table 5-7. BCSR0 Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0-5 | Reserved | ***S*** | 0 | R/W |
| 6 | GPL LED 1 | ***General Purpose Led 1.*** General purpose Led to be used by the user. | 0 | R,W |
| 7 | GPL LED 2 | ***General Purpose Led 2.*** General purpose Led to be used by the user. | 0 | R,W |
| 8 - 31 | Reserved | Un-implemented | 0 | R |

### 5.11.2 BCSR1 Board Control—Status Register 1

The BCSR1 is a control register on the MPC8272ADS. It is accessed at **offset 4** from BCSR base address. It may be read or written at any time[2]. BCSR1 gets its defaults upon Power-On reset. The fields are described in Table 5-8

**Table 5-8. BCSR1 Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 | Conf_Word | ***Config_Source***. When asserted (**low**) Hard Reset Configuration Word is sourced from the BCSR. When negated, Hard Reset Configuration Word is sourced from the FLASH/EEPROM. The assignments selection is done via a dedicated jumper JP7. | 0 | R |
| 1 | FLASH_CS0 | ***FLASH $\overline{CS0}$***. When asserted (**low**) $\overline{CS0}$ is assigned to the FLASH SIMM and $\overline{CS4}$ is assigned to $E^2$PROM. When negated, $\overline{CS0}$ is assigned to the $E^2$PROM and $\overline{CS4}$ is assigned to the FLASH SIMM. The assignments selection is done via a dedicated jumper. | 0 | R |
| 2 | ATM_EN | ***ATM Port Enable***. When asserted (**low**) the ATM UNI chip (PM5384) connected to FCC1 is enabled for transmission and reception. When negated, the ATM transceiver is in standby mode, freeing all its i/f signals for off-board use via the expansion connectors. | 1 | R,W |
| 3 | ATM_RST | ***ATM Port Reset***. When asserted (**low**), the ATM port transceiver is in reset state. This line is driven also by $\overline{HRESET}$ signal of the MPC8272. | 1 | R,W |

[1]Provided that BCSR is not disabled.

[2]Provided that BCSR is not disabled.

**Table 5-8. BCSR1 Description (continued)**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 4 | FETHIEN1 | ***Fast Ethernet Port 1 Initial Enable***. When asserted (**low**) the DM9161's MII port, residing on FCC1, is enabled after Power-Up or after FETH_RST is negated. When negated (**high**), the DM9161's MII port is isolated after Power-Up or after FETH_RST is negated and all i/f signals are tri-stated. After initial value has been set, this signal has no influence over the DM9161 and MII isolation may be controlled via MDIO 0.10 bit. | 1 | R,W |
| 5 | FETH1_RST | ***Fast Ethernet port 1 Reset***. When active (**low**) the DM9161 is reset. This line is also driven by HRESET signal of the MPC8272. Since MDDIS pin of the DM9161 is driven low with this application, the negation of this signal causes all the H/W configuration bits to be sampled for initial values and device control is moved to the MDIO channel, which is the control path of the MII port. | 1 | R,W |
| 6 | RS232EN_1 | ***RS232 port 1 Enable***. When asserted (**low**) the RS232 transceiver for port 1, is enabled. When negated, the RS232 transceiver for port 1, is in standby mode and SCC1 pins are available for off-board use via the expansion connectors. | 1 | R,W |
| 7 | RS232EN_2 | ***RS232 port 2 Enable***. When asserted (**low**) the RS232 transceiver for port 2, is enabled. When negated, the RS232 transceiver for port 2, is in standby mode and SCC4 pins are available for off-board use via the expansion connectors. | 1 | R,W |
| 8 - 31 | Reserved | Un-implemented | 0 | R |

## 5.11.3 BCSR2 Board Control—Status Register 2

BCSR2 is a status register which is accessed at **offset 8** from the BCSR base address. Its a read- only register which may be read at any time[1]. BCSR2s' various fields are described in Table 5-9

**Table 5-9. BCSR2 Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 - 7 | TSTAT(0:7) | ***Tool Status (0:7)***. This field is reserved for external tool status report. The exact meaning of each bit within this field is tool unique and therefore will be documented separately per each tool. These signals are available at the System expansion connector. | - | R |
| 8 - 11 | TOOLREV(0:3) | ***TOOL Revision (0:3)***. This field may contain the revision code of an external tool connected to the PowerQUICC II. The various combinations of this field will be described per each tool users' manual. These signals are available at the System expansion connector. The revision option for the external tools are shown in Table 5-15 | | R |

[1]Provided that BCSR is not disabled.

## Table 5-9. BCSR2 Description (continued)

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 12 - 15 | EXTTOLI(0:3) | ***External Tools Identification***. These lines, which are available at the CPM expansion connectors, are intended to serve as tools' identifier. On-board S/W may check these lines to detect The presence of various tools (h/w expansions) at the CPM expansion connectors. For the external tools' codes and their associated combinations see Table 5-12 | - | R |
| 16 - 17 | SWOPT(0:1) [1] | ***Software Option (0:1)***. This field shows the state of a dedicated dip-switches providing an option to manually change a program flow. | 0 | R |
| 18 - 19 | Reserved | Un-implemented | 0 | R |
| 20 - 21 | BVERN(0:1) | ***Board Version Number (0:1)***. This field represents the version code, hard-assigned to the MPC8272ADS. See Table 5-13, for version encoding. | 11 | R |
| 22 - 23 | BREVN(0:1) | ***Board Revision Number (0:1)***. This field represents the revision code, hard-assigned to the MPC8272ADS. See Table 5-14, for revisions' encoding. | - | R |
| 24 | SWOPT2 | ***Software Option 2***. This is the LSB of the field. Shows the state of a dedicated dip-switch providing an option to manually change a program flow. | 0 | R |
| 25 - 27 | FLASH_PD(7:5) | ***Flash Presence Detect(7:5)***. These lines are connected to the Flash SIMM presence detect lines, which encode the Delay of Flash SIMM mounted on the Flash SIMM socket. For the encoding of FLASH_PD(7:5) see Table 5-10 | - | R |
| 28 - 31 | FLASH_PD(4:1) | ***Flash Presence Detect(4:1)***. These lines are connected to the Flash SIMM presence detect lines which encode the type of Flash SIMM mounted on the Flash SIMM socket. For the encoding of FLASH_PD(4:1) see Table 5-11 | - | R |

[1] There is additional bit to this field. See next on the same table.

## Table 5-10. FLASH Presence Detect (7:5) Encoding

| FLASH_PD(7:5) | FLASH DELAY [nsec] |
|---|---|
| 000 | Not Supported |
| 001 | 150 |
| 010 | 100/120 |
| 011 | 80/90 |
| 100 | 70 |
| 101 - 111 | Not Supported |

### Table 5-11. FLASH Presence Detect (4:1) Encoding

| FLASH_PD(4:1) | Flash TYPE / SIZE |
|---|---|
| 0000 | SM73288XG4JHBG0 - 32 MByte (4 banks of 4 X 2M X 8) by Smart Modular Technology. |
| 0001 | SM73248XG2JHBG0 - 16 MByte (2 banks of 4 X 2M X 8) by Smart Modular Technology. |
| 0010 | SM73228XG1JHBG0 - 8 MByte (1 bank of 4 X 2M X 8) by Smart Modular Technology. |
| 0011 - 1111 | Not Supported |

### Table 5-12. EXTOOLI(0:3) Assignment

| EXTTOOLI(0:3) | External Tool |
|---|---|
| 0 | T/ECOM - PowerQUICC II Communication tool |
| 1 | Reserved |
| 2 | T1 Circuit Emulation Tool |
| 3 - E | Reserved |
| F | Tool Non Existent |

### Table 5-13. PowerQUICC II Board Version Encoding

| Version Number (0:1) [Hex] | PowerQUICC II Board Version |
|---|---|
| 0 | PowerQUICC IIFADS-ZU |
| 1 | Reserved |
| 2 | MPC8272ADS |
| 3 | PowerQUICC II7e ADS |

### Table 5-14. PowerQUICC II Board Revision Encoding

| Revision Number (0:1) [Hex] | PowerQUICC II Board Revision |
|---|---|
| 0 | ENG (Engineering) |
| 1 | PILOT |
| 2 | A |
| 3 | Reserved |

**Table 5-15. External Tool Revision Encoding**

| TOOLREV(0:3) [hex] | External Tool Revision |
|---|---|
| 0 | ENGINEERING |
| 1 | PILOT |
| 2 | A |
| 3 - F | Reserved |

## 5.11.4 BCSR3 Board Control - Status Register 3

BCSR3 is a control register which is accessed at **offset 0xC** from the BCSR base address. Its a read- write register which may be read or written at any time[1]. BCSR3s' various fields are described in Table 5-16

**Table 5-16. BCSR3 Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 | USB_EN | ***USB Port Enable***. When asserted (**low**) the USB chip connected to SCC4 is enabled for transmission and reception. When negated, the USB transceiver is in standby mode and its associated buffers [1] are in tri-state mode, freeing all its i/f signals for off-board use via the expansion connectors. | 1 | R/W |
| 1 | USB_HI_SPEED | ***USB Hi Speed***. When asserted (**low**) the USB chip connected to SCC4 is set for hi speed (12 Mbps) transmission and reception. When negated, the USB transceiver is set to low speed (1.5 Mbps) transmission and reception | 0 | R/W |
| 2 | USBVCC0 | ***USB Port VCC EN***. When asserted (**high**), 5V power is applied to the USB Bus. When negated, power to the USB port is disconnected. | 0 | R/W |
| 3 | FETHIEN2 | ***Fast Ethernet Port 2 Initial Enable***. When asserted (**low**) the DM9161's MII port, residing on FCC3, is enabled after Power-Up or after FETH_RST is negated. When negated (**high**), the DM9161's MII port is isolated after Power-Up or after FETH_RST is negated and all i/f signals are tri-stated. After initial value has been set, this signal has no influence over the DM9161 and MII isolation may be controlled via MDIO 0.10 bit. | 1 | R/W |
| 4 | FETH2_RST | ***Fast Ethernet port 2 Reset***. When active (**low**) the DM9161 is reset. This line is also driven by HRESET signal of the PowerQUICC II. Since MDDIS pin of the DM9161 is driven low with this application, the negation of this signal causes all the H/W configuration bits to be sampled for initial values and device control is moved to the MDIO channel, which is the control path of the MII port. | 1 | R/W |
| 5 | ATM16 | ***ATM 16 bit UTOPIA***. When asserted (**low**) the UTOPIA is set for 16 bit. When negated (**high**), the UTOPIA is set for 8 bit. | 1 | R/W |
| 6 | ATM_SINGLE_PHY | ***ATM SINGLE PHY***. When asserted (**low**) the UTOPIA is set to Multi PHY. When negated (**high**), the UTOPIA is set for Single PHY. | 1 | R/W |

[1]Provided that BCSR is not disabled.

**Table 5-16. BCSR3 Description (continued)**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 7 | PCI_MODE | ***PCI_MODE***. When asserted (**low**) the Local Bus function is set to PCI. When negated (**high**), the Local Bus is set for Local Bus SDRAM. | | R |
| 8-31 | Reserved | un-implemented | | |

1 Required for voltage levels adaptation.

## 5.11.5 BCSR4 Board Control—Status Register 4

BCSR4 is a status register which is accessed at **offset 0x10** from the BCSR base address. Its a read- only register which may be read at any time[1]. BCSR4s' various fields are described in Table 5-17

**Table 5-17. BCSR4 Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 - 1 | PCI0_PRSNT(0:1) | ***PCI Slot 0 Present (0:1)***. This field holds a code that tells whether a PCI expansion board is plugged in PCI slot 0 and the total power requirements of the board according to the PCI spec. The different expansion board types are listed in Table 5-18 | 11 | R |
| 2 - 3 | PCI1_PRSNT(0:1) | ***PCI Slot 1 Present (0:1)***. This field holds a code that tells whether a PCI expansion board is plugged in PCI slot 1 and the total power requirements of the board according to the PCI spec. The different expansion board types are listed in Table 5-18 | 11 | R |
| 4 - 5 | PCI2_PRSNT(0:1) | ***PCI Slot 2 Present (0:1)***. This field holds a code that tells whether a PCI expansion board is plunged in PCI slot 2 and the total power requirements of the board according to the PCI spec. The different expansion board types are listed in Table 5-18 | 11 | R |
| 6 | M66EN | ***66MHz Enable***. This field shows if one of the expansion boards used is not capable of operating in 66MHz mode:<br>'1' - All expansion boards are 66MHz capable<br>'0' - One of the expansion boards is not 66MHz capable | 1 | R |
| 7 | PCI_MODCK | ***PCI_MODCK.*** This field shows the PCI bus clock settings. | - | R |
| 8-31 | Reserved | un-implemented | | |

**Table 5-18. PCI Board Present Signal Definitions**

| PCIx_PRSNT (0:1) [Hex] | Expansion Configuration |
|---|---|
| 0 | Expansion board present, 7.5W maximum |
| 1 | Expansion board present, 25W maximum |

[1]Provided that BCSR is not disabled.

| PCIx_PRSNT (0:1) [Hex] | Expansion Configuration |
|---|---|
| 2 | Expansion board present, 15W maximum |
| 3 | No expansion board present |

### 5.11.6 BCSR5 and BCSR7 Board Control—Status Register 3 and 5

BCSR5 to BCSR7 are additional control / status registers which may be accessed as a word at **offset 0x14 to 0x1C** from BCSR base address. These registers are not implemented. They may be read or written but with no valid data nor any effect on the board. The description of BCSR3 and BCSR5 is shown in Table 5-19

**Table 5-19. BCSR5 to BCSR7 Description**

| BIT | MNEMONIC | Function | PON DEF | ATT. |
|---|---|---|---|---|
| 0 - 31 | Reserved | Un Implemented | - | - |

## 5.12 COP/JTAG Port

The COP - Control Observation Port, is part of the PowerQUICC II's JTAG machine, implemented as a set of additional instructions and logic within the JTAG permissions. This port may be connected to a dedicated debug station[1], for extensive system debug.

There are several third party debug solutions on the market. These debug-stations may be connected to the host computer via either Ethernet, Parallel-Port, RS232 or any other media.

The debug station connection scheme is shown in Figure 5-10.

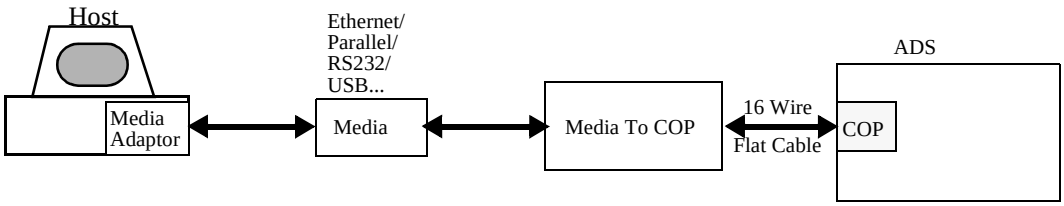



**Figure 5-10. Debug Station Connection Schemes**

To support debug station connection to the COP/JTAG port, a 16 pin generic header connector is provided on the MPC8272ADS, carrying the COP/JTAG signals as well as

[1]Not provided with the MPC8272ADS.

additional signals aiding in system debug. The pinout of this connector, which is a general Motorola recommendation for including a COP/JTAG port in a design, is shown in Figure 5-11 and detailed in Table 5-20.

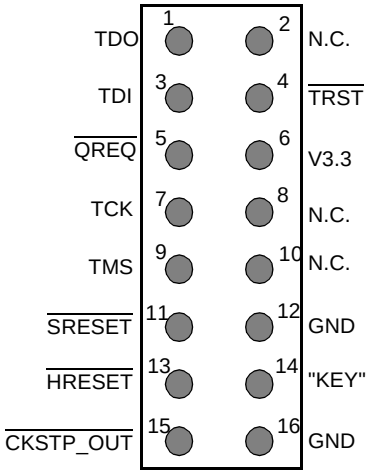



**Figure 5-11. COP/JTAG Port Connector**

**Table 5-20. COP/JTAG Port Signals Description**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| 1 | TDO | O | Transmit Data Out. This the JTAG's serial data output driven by Falling edge of TCK. |
| 2 | N.C. | - | Not Connected. |
| 3 | TDI | I | Transmit Data In. This is the JTAG serial data input, sampled by the PowerQUICC II on the rising edge of TCK. This line is pulled up internally by the PowerQUICC II. |
| 4 | $\overline{\text{TRST}}$ | I | Test port Reset (L). When this signal is active (Low), it resets the JTAG logic. This line is pull-down on the MPC8272ADS with a 1KΩ resistor, to provide constant reset of the JTAG logic. |
| 5 | $\overline{\text{QREQ}}$ | O | Quiescent Request (L). When asserted (low), this line indicates that the PowerQUICC II desires to enter low-power mode. This signal may be required by a debug station. |
| 6 | V3.3 | O | 3.3V power supply bus. |
| 7 | TCK | I | Test port Clock. This clock shifts in / out data to / from the PowerQUICC II JTAG port. Data is driven on the falling edge of TCK and is sampled both internally and externally on it's rising edge.<br>TCK is pulled up internally by the PowerQUICC II. |
| 8 | N.C. | - | Not Connected. |

**Table 5-20. COP/JTAG Port Signals Description (continued)**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| 9 | TMS | I | Test Mode Select. This signal qualified with TCK in a same manner as TDI, changes the state of the JTAG machine. This line is pulled up internally by the PowerQUICC II. |
| 10 | N.C. | - | Not Connected. |
| 11 | $\overline{\text{SRESET}}$ | I/O, O.D. | Soft Reset (L). This is the PowerQUICC II's soft reset which is in fact a non-maskable interrupt, making the PowerPC take the reset exception from the reset vector. This line may be driven by the PowerQUICC II as well during soft-reset sequence, for 512 system clocks. This line is pulled up on the MPC8272ADS with a 1KΩ resistor. When driven externally, it **MUST be driven with an Open Drain gate. Failure in doing so might result in permanent damage to the PowerQUICC II and / or to board logic.** |
| 12 | GND | O | Digital GND. Main GND plane. |
| 13 | $\overline{\text{HRESET}}$ | I/O, O.D. | PowerQUICC II's Hard Reset (L). When asserted by an external H/W, generates Hard-Reset sequence for the PowerQUICC II. During that sequence, asserted by the MPC for 512 system clocks. Pulled Up on the MPC8272ADS using a 1KΩ resistor. When driven by an external tool, **MUST be driven with an Open Drain gate. Failure in doing so might result in permanent damage to the PowerQUICC II and / or to board logic.** |
| 14 | N.C. | - | Not Connected. |
| 15 | $\overline{\text{XBR3}}$ ($\overline{\text{CKSTOP_OUT}}$) | I/O | Normally configured as $\overline{\text{XBR3}}$ which has no function with this connector. May be configured as $\overline{\text{CKSTOP_OUT}}$ - Check Stop Out (L). When asserted (Low) indicates that the PowerQUICC II core has entered a Check-Stop state. |
| 16 | GND | O | Digital GND. Main GND plane. |

# Chapter 6
# Memory Map

This chapter explains memory mapping in the MPC8272ADS.

## 6.1 Overview

All accesses to MPC8272ADS memory slaves are controlled by the MPC8272 memory controller. Therefore, the memory map is reprogrammable to suit the user. After the debug station performs hard reset, the debugger checks for existence, size, delay and type of the Flash memory SIMM mounted on board, decides the assignments of CS0 and CS4 ($E^2$PROM and Flash), and programs the memory controller accordingly. The SDRAM, $E^2$PROM, and Flash memory respond to all types of memory access (for example, problem / supervisory, program / data, and DMA).

This memory map is a recommended memory map, and because it is a 'soft' map, devices' addresses may be moved about the map for user convenience. Two memory maps depend on the device assigned to CS0 (regardless of the hard reset configuration word source). The memory address for the device assigned to CS0 is always the same as determined in the hard-reset configuration word. Because Flash and $E^2$PROM require different memory spaces, different memory maps are devised for each case. For details, see Table 6-1 and Table 6-2.

**Table 6-1. MPC8272ADS Memory Map—Flash (or BCSR) as Boot Device**

| Address Range | Memory Type | Device Name | | Port Size | Memory Size |
|---|---|---|---|---|---|
| 00000000 - 03FFFFFF | 60x SDRAM | 64 MByte | 64 MByte | 64 | 64 MBytes |
| 04000000 - 044FFFFF | Empty space | Optional 4MByte local bus SDRAM for legacy support | | — | 5 MBytes |

**Table 6-1. MPC8272ADS Memory Map—Flash (or BCSR) as Boot Device**

| Address Range | Memory Type | Device Name | Port Size | Memory Size |
|---|---|---|---|---|
| 04500000 - 04507FFF | BCSR(0:7) [1] | | 32 | 32 KBytes |
| 04500000 - 04507FE3 | BCSR0 | | | 4 Bytes |
| 04500004 - 04507FE7 | BCSR1 | | | 4 Bytes |
| 04500008 - 04507FEB | BCSR2 | | | 4 Bytes |
| 0450000C - 04507FEF | BCSR3 | | | 4 Bytes |
| 04500010 - 04507FF3 | BCSR4 | | | 4 Bytes |
| 04500014 - 04507FF7 | BCSR5 | | | 4 Bytes |
| 04500018 - 04507FFB | BCSR6 | | | 4 Bytes |
| 0450001C - 04507FFF | BCSR7 | | | 4 Bytes |
| 04508000 - 045FFFFF | Empty space | | - | ~1 MByte |
| 04600000 - 04607FFF [2] | ATM UNI proc. control | PMC5384 M/P I/F | 8 | 32 KBytes |
| 04608000 - 046FFFFF | Empty space | | - | ~1 MBytes |
| 04700000 [3] - 0471FFFF | PowerQUICC II Internal MAP [4] | | 32 | 128 KBytes |
| 04720000 - 0472FFFF | Empty space | | - | 64 KBytes |
| 04730000 - 04737FFF | PCI interrupt Controller | | 32 | 32 KBytes |
| 04738000 - 047FFFFF | Empty space | | - | ~800 KBytes |
| 04800000 - 04FFFFFF | PCI memory | Agents PIMMR (via PCI Direct) | | ~ 8 MBytes |
| 05000000 - 7FFFFFFF | Empty space | Tool Board is located at 60000000 and 70000000 | | ~ 2 GBytes |
| 80000000 - BFFFFFFF | PCI memory | PCI Agents GPL WIndows | 32 | 1 Gbyte |

## Table 6-1. MPC8272ADS Memory Map—Flash (or BCSR) as Boot Device

| Address Range | Memory Type | Device Name | | | Port Size | Memory Size |
|---|---|---|---|---|---|---|
| C0000000 - C1FFFFFF | Empty space | | | | - | 32 MBytes |
| C2000000[5] - C2007FFF | E$^2$PROM | ATMEL AT28HC64B | | | 8 | 32 KBytes |
| C2008000 - CFFFFFFF | Empty space | | | | | ~200 MBytes |
| D0000000 - D07FFFFF | Empty space | | | | | 8 MBytes |
| D0800000 - FDFFFFFF | Empty space | | | | | ~1 GBytes |
| FE000000[6] - FEFFFFFF | Flash SIMM | | | 32M SIMM - SM73288 | 32 | 32 MBytes |
| FF000000 - FF7FFFFF | | | 16M SIMM - SM73248 | | | |
| FF800000 - FFFFFFFF | | 8M SIMM - SM73228 | | | | |

1 The device appears repeatedly in multiples of its port-size (in bytes) X depth. That is, BCSR0 appears at memory locations 4700000, 4700020, 4700040..., while BCSR1 appears at 4700004, 4700024, 4700044... and so on.

2 The internal space of the ATM UNI control port is 256 bytes, however, the minimal block size that may be controlled by the GPCM is 32 KBytes.

3 Initially at h0F000000 - h0F00FFFF, set by hard reset configuration.

4 Refer to the *MPC8272 PowerQUICC II™ Family Reference Manual* for complete description of the internal memory map.

5 An 8 Kbyte device is used (16 Kbyte and 32 Kbyte devices can also be used) so it appears repeatedly in 8Kbyte multiples starting from C2000000.

6 Set by hard-reset configuration.

## Table 6-2. MPC8272ADS Memory Map—E$^2$PROM as Boot Device

| Address Range | Memory Type | Device Name | | Port Size | Memory Size |
|---|---|---|---|---|---|
| 00000000 - 03FFFFFF | SDRAM DIMM | 64 MByte | 64 MByte | 64 | 64 MBytes |
| 04000000 - 044FFFFF | Empty space | | | - | 5 MBytes |

## Table 6-2. MPC8272ADS Memory Map—E$^2$PROM as Boot Device

| Address Range | Memory Type | Device Name | Port Size | Memory Size |
|---|---|---|---|---|
| 04500000 - 04507FFF | BCSR(0:7) [1] | | 32 | 32 KBytes |
| 04500000 - 04507FE3 | BCSR0 | | | 4 Bytes |
| 04500004 - 04507FE7 | BCSR1 | | | 4 Bytes |
| 04500008 - 04507FEB | BCSR2 | | | 4 Bytes |
| 0450000C - 04507FEF | BCSR3 | | | 4 Bytes |
| 04500010 - 04507FF3 | BCSR4 | | | 4 Bytes |
| 04500014 - 04507FF7 | BCSR5 | | | 4 Bytes |
| 04500018 - 04507FFB | BCSR6 | | | 4 Bytes |
| 0450001C - 04507FFF | BCSR7 | | | 4 Bytes |
| 04508000 - 045FFFFF | Empty space | | - | ~1 MBytes |
| 04600000 - 04607FFF [2] | ATM UNI proc. control | PMC5384 M/P I/F | 8 | 32 KBytes |
| 04608000 - 046FFFFF | Empty space | | - | ~1 MBytes |
| 04700000 [3] - 0471FFFF | PowerQUICC II Internal MAP [4] | | 32 | 128 KBytes |
| 04720000 - 0472FFFF | Empty space | | - | 64 KBytes |
| 04730000 - 04737FFF | PCI interrupt controller | | 32 | 32 KBytes |
| 04738000 - 047FFFFF | Empty space | | - | ~800 KBytes |
| 04800000 - 04FFFFFF | PCI memory | Agents PIMMR (via PCI Direct) | | ~ 8 MBytes |
| 05000000 - 7FFFFFFF | Empty space | Tool Board is located at 60000000 and 70000000 | | ~ 2 GBytes |
| 80000000 - BFFFFFFF | PCI memory | PCI Agents GPL WIndows | 32 | 1 Gbytes |

**Table 6-2. MPC8272ADS Memory Map—E$^2$PROM as Boot Device**

| Address Range | Memory Type | Device Name | | | Port Size | Memory Size |
|---|---|---|---|---|---|---|
| C0000000 - C1FFFFFF | Empty space | | | | - | 32 MBytes |
| C2000000 - C2FFFFFF | Flash SIMM | | | 32M SIMM - SM73288 | 32 | 32 MBytes |
| C3000000 - C37FFFFF | | | 16M SIMM - SM73248 | | | |
| C3800000 - C3FFFFFF | | 8M SIMM - SM73228 | | | | |
| C4000000 - CFFFFFFF | Empty space | | | | | ~200 MBytes |
| D0000000 - D07FFFFF | Empty space | | | | | 8 MBytes |
| D0800000 - FFFFDFFF | Empty space | | | | | ~1 GBytes |
| FFF00000[5] - FFFFFFFF | E$^2$PROM | ATMEL AT28HC64B | | | 8 | 32 KBytes |

1 The device appears repeatedly in multiples of its port-size (in bytes) X depth. E.g., BCSR0 appears at memory locations 4700000, 4700020, 4700040..., while BCSR1 appears at 4700004, 4700024, 4700044... and so on.

2 The internal space of the ATM UNI control port is 256 bytes, however, the minimal block size that may be controlled by the GPCM is 32 KBytes.

3 Initially at h0F000000 - h0F00FFFF, set by hard reset configuration.

4 Refer to the *MPC8272 PowerQUICC II™ Family Reference Manual* for complete description of the PowerQUICC II internal memory map.

5 An 8-Kbyte device is used (16- and 32-Kbyte devices can also be used) so it appears repeatedly in 8 Kbyte multiples starting from FFF00000.

## 6.2 PowerQUICC II Register Programming

The PowerQUICC II provides the following functions on the MPC8272ADS:

- System functions that include the following:
  - PPC bus SDRAM controller
  - Chip-select generator
- Communication functions that include the following:
  - ATM SAR
  - Dual Fast Ethernet controller
  - UART for terminal or host computer connection
  - USB controller

The internal registers of the MPC8272 must be programmed after hard reset as described in the following sections. The addresses and programming values are in hexadecimal base. For more information about the this topic in the following sections of this chapter, see the *MPC8272 PowerQUICC II™ Family Reference Manual*.

## 6.2.1 System Initialization

The power-on reset configuration word is set in the BCSR or Flash or in the $E^2$PROM. There are two configuration words: one for the BCSR and Flash (when it is assigned to $\overline{CS0}$) and the other to the $E^2$PROM (when it is assigned to $\overline{CS0}$). The two configurations are detailed in Table 6-3 and Table 6-4, respectively.

**Table 6-3. BCSR/Flash Power On Reset Configuration [1]**

| Flash Address [hex] | Init Value[hex] | Description |
|---|---|---|
| 0 | 0C | Internal arbitration, Internal memory controller, Core enabled, Single PowerQUICC II, 32 Bit boot port size, Exceptions vectored to 0xFFFxxxxx, Internal space 64 bit slave for external master. |
| 8 | 72 | IRQ signals configured as BADDRx lines, DP(1:7) configured as IRQ I/F and IRQ(6:7),Initial internal space @ 0x0F000000 |
| 10 | 36 | Boot memory space @ 0xFE000000 - 0xFFFFFFFF, $\overline{ABB/IRQ2}$ pin is $\overline{ABB}$, $\overline{DBB/IRQ3}$ pin is $\overline{DBB}$, No masking on bus request lines, Local bus pins function as PCI, PCI is boot master, AP(1;3) configured as BNKSEL(0:2), $\overline{APE}$ configured as $\overline{IRQ5}$. |
| 18 | 5A | $\overline{CS10}$ configured as $\overline{BCTL1}$ |

[1] Programmed into the Flash ($E^2$PROM) memory in addresses 0x0, 0x8, 0x10 & 0x18

**Table 6-4. $E^2$PROM Power On Reset Configuration [1]**

| EEPROM Address [hex] | Init Value[hex] | Description |
|---|---|---|
| 0 | 04 | Internal arbitration, Internal memory controller, Core enabled, Single PowerQUICC II (60X Bus mode[b]), 8 Bit Boot size, Exceptions vectored to 0xFFFxxxxx, Internal space 64 bit slave for external master. |
| 8 | 72 | IRQ signals configured as BADDRx lines, DP(1:7) configured as IRQ I/F and IRQ(6:7),Initial internal space @ 0x0F000000 |
| 10 | 36 | Boot memory space @ 0xFE000000 - 0xFFFFFFFF, $\overline{ABB/IRQ2}$ pin is $\overline{ABB}$, $\overline{DBB/IRQ3}$ pin is $\overline{DBB}$, No masking on bus request lines, Local bus pins function as PCI, PCI is boot master, AP(1;3) configured as BNKSEL(0:2), $\overline{APE}$ configured as $\overline{IRQ7}$ and $\overline{CS11}$ as $\overline{CS11}$. |
| 18 | 5A | $\overline{CS10}$ configured as $\overline{BCTL1}$ |

[1] Programmed into the E$^2$PROM in addresses 0x0, 0x8, 0x10 & 0x18

**Table 6-5. SIU Register Programming**

| Register | Init Value[hex] | Description |
|---|---|---|
| RMR | 0001 | Check-Stop Reset enabled. |
| IMMR | 04700000 | Internal space @ 0x047000000 |
| SYPCR | FFFFFFC3 | Software watchdog timer count - FFFF, Bus-monitor timing FF, PPC Bus-monitor - Enabled, Local Bus-monitor - Enabled, S/W watch-dog - disabled, S/W watch-dog (if enabled) causes reset, S/W watch-dog (if enabled) - prescaled. |
| BCR | 100C0000 | Single PowerQUICC II, 1 wait-states on address tenure, No L2Cache, 1 clock hit delay (when L2cache available), 1-level Pipeline depth, Extended transfer mode enabled for PCC, Extended transfer mode disabled for Local Buses, Odd parity for PPC & Local Buses, External Master delay enabled, Internal space responds as 64 bit slave for external master (not relevant for this application). |

## 6.2.2 Memory Controller Register Programming

The memory controller on the MPC8272ADS is initialized to 100MHz operation, that is, the registers' programming is based on 100-MHz timing calculation. It also works for slower bus speeds, but the timing must be optimized). The two possible initialization for the memory controller are the following:

- Flash SIMM is assigned to $\overline{CS0}$ and E$^2$PROM is assigned to $\overline{CS4}$.
- Flash SIMM is assigned to $\overline{CS4}$ and E$^2$PROM is assigned to $\overline{CS0}$.

Both options are shown in Table 6-6 and Table 6-7, respectively.

**Table 6-6. Memory Controller Initialization For 100MHz—Flash as Boot Device**

| Reg. | Device Type | Bus | Init Value [hex] | Description |
|---|---|---|---|---|
| BR0 | SM73228XG1JHBG0 by Smart Modular Tech. | PPC | FF801801 | Base at FF800000, 32 bit port size, no parity, GPCM |
| | SM73248XG2JHBG0 by Smart Modular Tech. | | FF001801 | Base at FF000000, 32 bit port size, no parity, GPCM |
| | SM73288XG4JHBG0 by Smart Modular Tech. | | FE001801 | Base at FE000000, 32 bit port size, no parity, GPCM |
| OR0 | SM73228XG1JHBG0 by Smart Modular Tech. | | FF800876 | 8MByte block size, CS early negate, 11 w.s., Timing relax |
| | SM73248XG2JHBG0 by Smart Modular Tech. | | FF000876 | 16MByte block size, CS early negate, 11 w.s., Timing relax |
| | SM73288XG4JHBG0 by Smart Modular Tech. | | FE000876 | 32MByte block size, CS early negate, 11 w.s., Timing relax |

## Table 6-6. Memory Controller Initialization For 100MHz—Flash as Boot Device

| Reg. | Device Type | Bus | Init Value [hex] | Description |
|---|---|---|---|---|
| BR1 | BCSR | PPC | 04501801 | Base at 04500000, 32 bit port size, no parity, GPCM |
| OR1 | | | FFFF8010 | 32 KByte block size, all types access, 1 w.s. |
| BR2 | SDRAM MT48LC4M32B2 by MICRON | PPC | 00000041 | Base at 0, 64 bit port size, no parity, SDRAM machine 1 |
| OR2 | | | FE002EC0 | 32MByte block size, 4 banks per device, row starts at A7, 12 row lines, internal bank interleaving allowed, normal AACK operation |
| BR4 | $E^2$PROM | PPC | C2000801 | Base at C2000000, 8 bit port size, write protect disabled, no parity, GPCM |
| OR4 | AT28HC64B-70JC by Atmel | | FFFF8866 | 32 KByte block size, $\overline{CS}$ output half a clock after address, all types access, 6 w.s., Timing relax |
| BR5 | PM5384 - ATM UNI | PPC | 04600801 | Base at 04600000, 8 bit port size, no parity, GPCM on PPC bus. |
| OR5 | | | FFFF8E56 | 32K Byte block size, delayed CS assertion, early CS and WE negation for write cycle, relaxed timing, 7 w.s. for read, 8 for write, extended hold time after read. |
| BR3 | PCI Interrupt Controller | PPC | 04731801 | Base at 04730000, 32 bit port size, no parity, GPCM on PPC bus. |
| OR3 | | | FFFF8010 | 32 KByte block size, all types access, 1 w.s. |

## Table 6-7. Memory Controller Initialization For 100MHz—$E^2$PROM as Boot Device

| Reg. | Device Type | Bus | Init Value [hex] | Description |
|---|---|---|---|---|
| BR0 | $E^2$PROM | PPC | FFF00801 | Base at FFFFE000, 8 bit port size, write protect disabled, no parity, GPCM |
| OR0 | AT28HC64B-70JC by Atmel | | FFFF8866 | 32 KByte block size, $\overline{CS}$ output half a clock after address, all types access, 6 w.s., Timing relax |
| BR1 | BCSR | PPC | 04501801 | Base at 04500000, 32 bit port size, no parity, GPCM |
| OR1 | | | FFFF8010 | 32 KByte block size, all types access, 1 w.s. |
| BR2 | SDRAM MT48LC8M16A2 by MICRON | PPC | 00000041 | Base at 0, 64 bit port size, no parity, SDRAM machine 1 |
| OR2 | | | FE002EC0 | 32MByte block size, 4 banks per device, row starts at A7, 12 row lines, internal bank interleaving allowed, normal AACK operation |

## Table 6-7. Memory Controller Initialization For 100MHz—E$^2$PROM as Boot Device

| Reg. | Device Type | Bus | Init Value [hex] | Description |
|---|---|---|---|---|
| BR4 | SM73228XG1JHBG0 by Smart Modular Tech. | PPC | C3801801 | Base at C3800000, 32 bit port size, no parity, GPCM |
| | SM73248XG2JHBG0 by Smart Modular Tech. | | C3001801 | Base at C3000000, 32 bit port size, no parity, GPCM |
| | ASM73288XG4JHBG0 by Smart Modular Tech. | | C2001801 | Base at C2000000, 32 bit port size, no parity, GPCM |
| OR4 | SM73228XG1JHBG0 by Smart Modular Tech. | | FF800876 | 8MByte block size, CS early negate, 11 w.s., Timing relax |
| | SM73248XG2JHBG0 by Smart Modular Tech. | | FF000876 | 16MByte block size, CS early negate, 11 w.s., Timing relax |
| | SM73288XG4JHBG0 by Smart Modular Tech. | | FE000876 | 32MByte block size, CS early negate, 11 w.s., Timing relax |
| BR5 | PM5384 - ATM UNI | PPC | 04600801 | Base at 04600000, 8 bit port size, no parity, GPCM on PPC bus. |
| OR5 | | | FFFF8E56 | 32K Byte block size, delayed CS assertion, early CS and WE negation for write cycle, relaxed timing, 7 w.s. for read, 8 for write, extended hold time after read. |
| BR3 | PCI Interrupt Controller | PPC | 04731801 | Base at 04730000, 32 bit port size, no parity, GPCM on PPC bus. |
| OR3 | | | FFFF8010 | 32 KByte block size, all types access, 1 w.s. |

## Table 6-8. Memory Controller Initialization For 100MHz

| Reg. | Device Type | Bus | Init Value [hex] | Description |
|---|---|---|---|---|
| PSDMR | MT48LC8M16A2 (64 MByte) | PPC<br><br>Single PowerQUICC II Bus Mode | C34F36A3 | **Page** Based Interleaving, Refresh enabled, normal operation mode, address muxing mode 3, A14-A16 on BNKSEL, A7 on PSDA10, 8 clocks refresh recovery, 3 clocks precharge to activate delay, 3 clocks activate to read/write delay, 4 beat burst length, 2 clock last data out to precharge, 2 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 3 clocks CAS latency. |
| PSRT | PPC Bus SDRAM Supported | PPC | 13 | Divide MPTPR output by 20 (PSRT +1) Generates refresh every 8.2 µsec., while 15.6µsec. required. This will work also for 66MHz bus (12.4µsec). |
| MPTPR | All SDRAMs on board | | 2800 | Divide Bus clock by 41 (MPTPR+1) (decimal) |

# Chapter 7
# Physical Properties

This chapter explains aspects of the MPC8272ADS physical properties.

## 7.1 Power Supply

The board gets the power from the ATX power supply (which seats in an ATX chassis). All the power rails on the board are derived from the ATX power supply. The following power rails are included with the PowerQUICC II:

- $V_{DDH}$ (I/O)
- $V_{DDL}$ (Internal Logic)
- VCCSYN (CPM PLL)
- VCCSYN1 (core PLL)

The MPC8272ADS (see Figure 7-1) has the following power rails on it:

- VCC (5V) rail
- Stand By (5V) rail
- V3.3 (3.3V) rail
- $V_{DDL}$ (1.7V-2.5V) rail
- +12 V rail

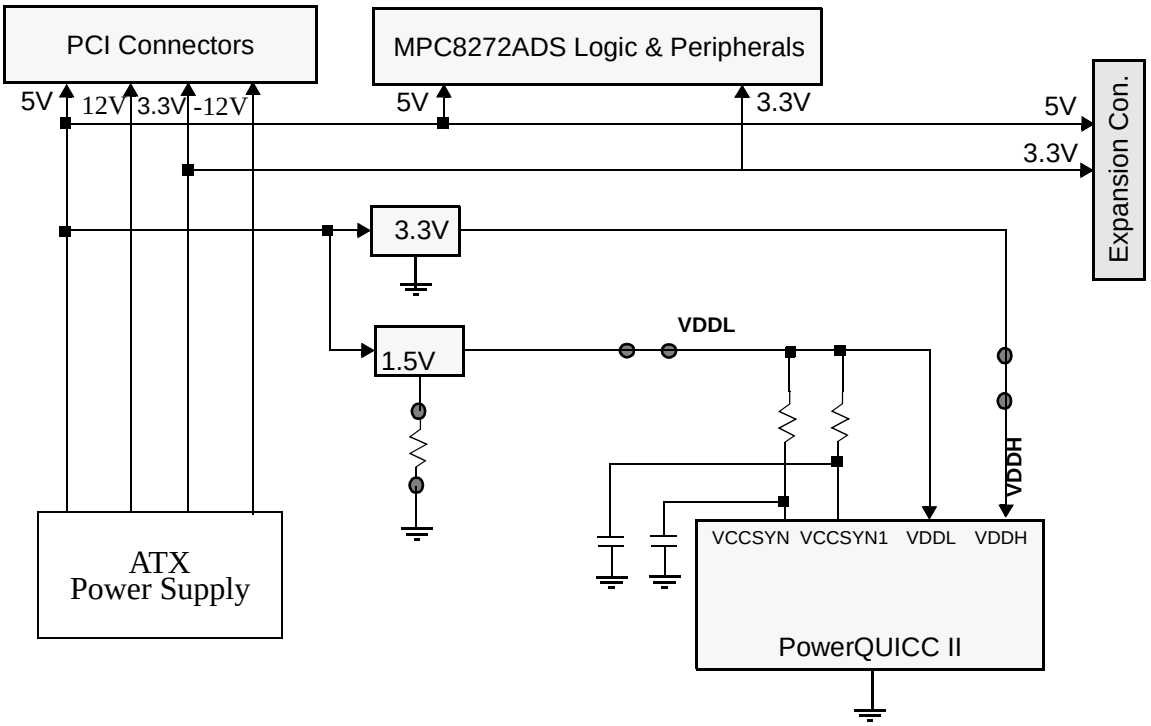



**Figure 7-1. MPC8272ADS Power Scheme**

To support off-board application development, the power buses are connected to the expansion connectors so that external logic may be powered directly from the board. The maximum current allowed to be drawn from the board on each bus also depends on the current drawn by the PCI bus. The figures are shown in Figure 7-1.

**Table 7-1. Expansion Connectors Maximum Current Consumption**

| Power Bus | Max. Current |
|---|---|
| VCC | TBD |
| V3.3 | TBD |

The PCI Standard specifies that each add-in card should consume maximum 25 W from all power sources combined. The maximum current consumption allowed per power source for a total of 25 W according to the PCI standard is shown in Table 7-2.

**Table 7-2. Maximum Power Consumption Per Add-In Card**

| Power Rail | Add-In Card |
|---|---|
| 5 V | 5A Max. (system dependent) |
| 3.3 V | 7.6A Max. (system dependent) |
| 12 V | 500mA |

**Table 7-2. Maximum Power Consumption Per Add-In Card (continued)**

| Power Rail | Add-In Card |
|---|---|
| -12V | 100mA |

### 7.1.1 5-V Rail

Some of the MPC8272ADS peripherals (not including the PCI add-in cards that should be 3.3 V *only* on the PCI interface, but can use 5 V for other components on-board) reside on the 5-V bus. Because the MPC8272 is not 5-V tolerant, buffering is provided between 5V peripherals and the MPC8272, protecting the MPC8272 from the higher voltage level.

### 7.1.2 3.3-V Rail

The MPC8272, SDRAM, PCI add-in cards, address and data buffers are powered by the 3.3 bus, which is produced from the ATX power supply.

### 7.1.3 5-V Standby Rail

The 5V standby power rail comes from the ATX Power Supply. Its only use is to power the logic required to support the power button in the front panel on the ATX chassis.

### 7.1.4 $V_{DDH}$ Rail

The MPC8272 $V_{DDH}$ power bus (3.3 V) is produced from the 5-V bus using a low-voltage drop linear voltage regulator MIC29501-3.3BU, made by Micrel.

A production option is made so that the level on this bus may be varied by means of trimming the potentiometer TR2. Although this option requires replacing some components, it allows the $V_{DDH}$ to be in the range of 3.0 V–3.6 V.

#### 7.1.4.1 $V_{DDL}$ Bus

The MPC8272 internal logic and the PLL are powered with a lower-voltage power source, whose voltage may be in the range of 1.3V–1.7V.

A trimming potentiometer does the fine tuning within a range.

Changing the voltage to the core logic of the MPC8272 obviously has an influence over the maximal speed of the core. The power-speed trade-off is that lower operation speeds may be obtained with lower voltage supply.

#### 7.1.4.2 12-V Rail

The 12-V bus from the ATX Power Supply supports the PCI slots and the VPP 12-V option from programming the FLASH.

#### 7.1.4.3 -12-V Rail

The -12V bus from the ATX Power Supply supports the PCI slots.

## 7.2 Connectors

The MPC8272ADS has connectors attached to serve the following functions:

- ATX power supply
- 100 / 10 - Base-T Ethernet ports
- ATM 155 Mbps port
- RS232 port 1
- RS232 port 2
- CPM expansion
- COP / JTAG
- Logic analyzer connectors
- Programmable logic in system programming (ISP)
- PCI connectors
- System expansion
- USB connector
- Parallel port connector

### 7.2.1 ATX Power Connector

The ATX power connector is a 20-lead, standard ATX power connector. The female part is soldered to the PCB and the plug is connected to the power supply to facilitate fast connection or disconnection of power.

### 7.2.2 Fast Ethernet Port Connectors

The Ethernet connector on the MPC8272ADS is a twisted-pair (100/10-Base-T) connector. Use is done with $90^{0}$ RJ45-8 connector.

### 7.2.3 ATM 155 Port Connection

The ATM 155 I/F to the media is optical rather than electrical. Use is done with HP's HFBR 5805 optical I/F, which is placed on the edge of the board for convenient connection.

### 7.2.4 RS232 Port Connector

The RS232 port connector is a stacked 9-pin, 90°, female D-Type connector that saves board space (made of two connectors for two ports).

### 7.2.5 CPM Expansion Connector

The CPM expansion connectors carries all CPM pins, that is, Port A to Port D signals. Use is done with DIN 41612, 128 pin T.H. PCB connector, residing on the board, allowing convenient vertical connection to off-board tools. Power supply pins are also provided through this connector.

### 7.2.6 COP/JTAG Port Connector

The debug port connector is a Motorola standard COP/JTAG connector for the 60X processors family. It is a generic 16-pin (2 X 8), male, SMD, 90° protected header connector.

### 7.2.7 Logic Analyzer Connectors

To support fast connection to HP's 16500 logic analyzers series for debugging purposes, a set of dedicated connectors is provided. Use is done with 38-pin, SMT, high density, matched impedance MICTOR connectors that AMP makes.

These connectors carry the unbuffered 60X signals and should be placed as near to the PowerQUICC II as possible to provide short PCB routes to yield better reflections and crosstalk immunity. They also carry the PCI bus signals. CPM signals also have connectors.

### 7.2.8 Mach's In System Programming (ISP) Connector

This is a 10 pin generic 0.100" pitch header connector, providing In System Programming capability for Vantis-made programmable logic on board.

### 7.2.9 PCI Connectors

A set of three standard PCI 3.3V keyed, 124 pin, 32-bit connectors is provided for connecting up to three PCI add-in cards.

### 7.2.10 System Expansion Connector

The system expansion connector is a 128-pin DIN 41612 connector, which provides a minimal system I/F required to interface to other tool-boards that may use the CPM expansion connector. This connector contains 16-bit (lower PPC bus) address lines, 16-bit (higher PPC bus) data lines, plus useful GPCM and UPM control lines.

### 7.2.11 USB Connector

The USB connectors are standard type-A and type-B USB connectors.

### 7.2.12 Parallel Port Connector

The parallel connector is a standard 25 pin D-Type male connector.

## 7.3 PCB Layout

The MPC8272ADS layout was done for high-frequency operation and follows the PCI standard layout recommendations closely. The following list of measures are taken to meet this design goal:

- Traces as short as possible
- Clock signals and sensitive strobe signals shielded and routed as a chain
- Multilayer PCB with ground and supply layers
- PCI signals lengths and impedance according to PCI standard Rev. 2.2

# Chapter 8
# Support

This chapter provides all information needed for support, maintenance and connectivity to the MPC8272ADS.

## 8.1 Interconnect Signals

The MPC8272ADS interconnects with external devices by using the following set of connectors:

1. P13—RS232 ports 1 and 2
2. P16, P19—USB connectors
3. P10 and P23—100 / 10 - Base-T Ethernet ports
4. P21—COP / JTAG
5. P1—CPM expansion
6. P9, P11, P14, P22, P17, P5, P20, P18, P15, P24,P12—Logic analyzer MICTOR connectors
7. P26, P28, P29—PCI slots connectors
8. P31—ATX power supply connector
9. P4,P25, P3—Mach/lattice and ALTERA In System Programming (ISP)
10. P2—System expansion
11. P27—Parallel port connector

### 8.1.1 P13—RS232 Ports 1 and 2 Connectors

P13 is a dual 9-pin D-Type connector as described in Table 8-1.

**Table 8-1. P13 Connector**

| Pin No. | Signal Name | Description |
|---|---|---|
| 1 | CD | Carrier detect output from the MPC8272ADS |
| 2 | TX | Transmit data output from the MPC8272ADS |
| 3 | RX | Receive data input to the MPC8272ADS |

**Table 8-1. P13 Connector (continued)**

| Pin No. | Signal Name | Description |
|---|---|---|
| 4 | DTR | Data terminal ready input to the MPC8272ADS |
| 5 | GND | Ground signal of the MPC8272ADS |
| 6 | DSR | Data set ready output from the MPC8272ADS |
| 7 | N.C. | No connect |
| 8 | CTS | Clear to send output from the MPC8272ADS |
| 9 | N.C. | No connect |

## 8.1.2 P10 and P23 100/10 Base-T Ethernet Port Connector

P10 or P23 is a RJ-45 type connector for twisted pair Ethernet as described in Table 8-2.

**Table 8-2. P10,P23 100/10 Base-T Ethernet Connector**

| Pin No. | Signal Name | Description |
|---|---|---|
| 1 | TPTX | Twisted-pair transmit data positive output from the MPC8272ADS |
| 2 | TPTX~ | Twisted-pair transmit data negative output from the MPC8272ADS |
| 3 | TPRX | Twisted-pair receive data positive input to the MPC8272ADS |
| 4 | N.C. | Not connected, Bob Smith terminated on the MPC8272ADS |
| 5 | | |
| 6 | TPRX~ | Twisted-pair receive data negative input to the MPC8272ADS |
| 7 | N.C. | Not connected, Bob Smith terminated on the MPC8272ADS |
| 8 | | |

## 8.1.3 P21—COP/JTAG Connector

P21 is a Motorola standard COP / JTAG connector for the 60X processors family. It is a 16 pin-protected header connector as described in Table 8-3.

**Table 8-3. P16—COP/JTAG Connector**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| 1 | TDO | O | Transmit data output. This the PowerQUICC II's JTAG serial data output driven by Falling edge of TCK. |
| 2 | GND | O | Digital GND. Main GND plane. |
| 3 | TDI | I | Transmit data in. This is the JTAG serial data input of the ADS, sampled on the rising edge of TCK. |

## Table 8-3. P16—COP/JTAG Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| 4 | TRST# | I | Test port reset~ (L). When this signal is active (Low), it resets the JTAG logic of the PowerQUICC II. This line is pull-down on the ADS with a 1KΩ resistor, to provide constant reset of the JTAG logic. |
| 5 | QREQ# | O | Quiescent request (L). When asserted (low), this line indicates that the PowerQUICC II desires to enter low-power mode. This signal may be required by a debug station. |
| 6 | 3v3 | O | 3.3V power supply bus. |
| 7 | TCK | I | Test port clock. This clock shifts in / out data to / from the JTAG logic. Data is driven on the falling edge of TCK and is sampled both internally and externally on it's rising edge.<br>TCK is pulled up internally by the PowerQUICC II. |
| 8 | N.C. | - | Not connected. |
| 9 | TMS | I | Test Mode Select. This signal qualified with TCK in a same manner as TDI, changes the state of the JTAG machine. This line is pulled up internally by the PowerQUICC II. |
| 10 | GND | O | Digital GND. Main GND plane. |
| 11 | SRESET# | I/O, O.D. | Soft Reset (L). This is the PowerQUICC II's soft reset which is in fact a non-maskable interrupt, making the PPC take the reset exception from the reset vector. This line may be driven by the PowerQUICC II as well during soft-reset sequence, for 512 system clocks. This line is pulled up on the ADS with a 1KΩ resistor. When driven externally, it **MUST be driven with an Open Drain gate. Failure to do so may result in permanent damage to the PowerQUICC II and / or to ADS logic.** |
| 12 | GND | O | Digital GND. Main GND plane. |
| 13 | HRESET# | I/O, O.D. | PowerQUICC II's Hard Reset (L). When asserted by an external H/W, generates Hard-Reset sequence for the PowerQUICC II. During that sequence, asserted by the PowerQUICC II for 512 system clocks. Pulled Up on the ADS using a 1KΩ resistor.<br>When driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so may result in permanent damage to the PowerQUICC II and / or to ADS logic.** |
| 14 | N.C. | - | Not Connected. |
| 15 | XBR3#<br>(CKSTOP_OUT#) | I/O | Normally configured as XBR3# which has no function with this connector. May be configured as CKSTP_OUT# - Check Stop Out (L). When asserted (Low) indicates that the PowerQUICC II core has entered a Check-Stop state. |
| 16 | GND | O | Digital GND. Main GND plane. |

### 8.1.4 P1—CPM Expansion Connector

P1 is an 128-pin, $90^0$, DIN 41612 connector that allows for convenient expansion of the MPC8272 serial ports. This connector contains all CPM pins plus power supply pins to provide for easy tool connection as described in Table 8-4.

**Table 8-4. P1—CPM Expansion Connector**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| A1 | RS_RXD1 (PD31 [1]) | I/O, T.S. | When RS232 port #1 is enabled, this signal is the receive data line for that port. When this port is disabled, this signal is tristated and may be used to any available alternate function for PD31. |
| A2 | RS_TXD1 (PD30) | I/O, T.S. | When RS232 port #1 is enabled, this signal is the transmit data line for that port. When this port is disabled, this signal may be used to any available alternate function for PD30. |
| A3 | ATMRXADR3(PD29) | I/O, T.S | ATM Multi-PHY Rx Address 3. When the ATM port Multi-PHY function is enabled, this line is connected to the receive PHY Address of the PM5384 ATM UNI. When this port is disabled, this signal is tristated and may be used for any available function of PD29. |
| A4 | | | |
| A5 | | | |
| A6 | | | |
| A7 | USBRXD (PD25) | I/O, T.S. | USB RxD Line. When the USB port is enabled, this line is connected to the receive data of the USB tranceiver. When this port is disabled, this signal is tristated and may be used for any available function of PD25. |
| A8 | USBTN (PD24) | I/O, T.S. | USB TN Line. When the USB port is enabled, this line is connected to the transmit negative line of the USB tranceiver. When this port is disabled, this signal is tristated and may be used for any available function of PD24. |
| A9 | USBTP (PD23) | I/O, T.S. | USB TP Line. When the USB port is enabled, this line is connected to the transmit positive line of the USB tranceiver. When this port is disabled, this signal is tristated and may be used for any available function of PD23. |
| A10 | | | |
| A11 | | | |
| A12 | | | |
| A13 | ATMTXADR4(PD19) | I/O, T.S | ATM Multi-PHY Tx Address 4. When the ATM port Multi-PHY function is enabled, this line is connected to the transmit PHY Address of the PM5384 ATM UNI. When this port is disabled, this signal is tristated and may be used for any available function of PD19. |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| A14 | ATMRXADR4(PD18) | I/O, T.S | ATM Multi-PHY Rx Address 4. When the ATM port Multi-PHY function is enabled, this line is connected to the receive PHY Address of the PM5384 ATM UNI. When this port is disabled, this signal is tristated and may be used for any available function of PD18. |
| A15 | ATMRXPTY (PD17) | I/O, T.S. | ATM Receive Parity Line. When the ATM port is enabled, this line is connected to the receive parity of the PM5384 ATM UNI. When this port is disabled, this signal is tristated and may be used for any available function of PD17. |
| A16 | ATMTXPTY (PD16) | I/O, T.S. | ATM Transmit Parity Line. When the ATM port is enabled, this line is connected to the transmit parity of the PM5384 ATM UNI. When this port is disabled, this signal may be used for any available function of PD16. |
| A17 | I2CSDA(PD15) | I/O, T.S. | This signal is connected to the serial $I^2C$ data line. This line may be used off-board as an $I^2C$ data line for external $I^2C$ device. |
| A18 | I2CSCL(PD14) | I/O, T.S. | This signal is connected to the serial $I^2C$ clock line. This line may be used off-board as an $I^2C$ clock line for external $I^2C$ device. |
| A19 | | | |
| A20 | | | |
| A21 | | | |
| A22 | | | |
| A23 | | | |
| A24 | | | |
| A25 | ATMTXADR3(PD7) | I/O, T.S | ATM Multi-PHY Tx Address 3. When the ATM port Multi-PHY function is enabled, this line is connected to the transmit PHY Address of the PM5384 ATM UNI. When this port is disabled, this signal is tristated and may be used for any available function of PD7. |
| A26 | | | |
| A27 | | | |
| A28 | | | |
| A29 | ATMRCLKDIS | I | ATM Receive Clock Out Disable. When active (H), the ATMRCLK output, on pin C29 of this connector, is Tri-stated. When either not connected or driven low, ATMRCLK on pin C29, is enabled. This provides compatibility with ENG revision of T/ECOM communication tools. |
| A30 | EXPVCC | O | 5V Supply. Connected to ADS's 5V VCC plane. Provided as power supply for external tool. |
| A31 | | | |
| A32 | | | |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| B1 | ATMTXEN# (PA31) | I/O, T.S. | ATM Transmit Enabled (L). When this signal is asserted (Low), while the ATM port is enabled and ATMTFCLK is rising, an octet of data, ATMTXD(7:0), is written into the transmit FIFO of the PM5384. When the ATM port is disabled, this line may be used for any available function of PA31. |
| | FETH1COL (PA31) | I/O, T.S. | Fast-Ethernet Port 1 Collision Detected (H). When this signal is asserted (High) by the DM9161, while the ethernet port is enabled, it indicates a Collision state over the line. When the DM9161 is in Full-Duplex mode, this line is inactive.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of the PA31. |
| B2 | ATMTCA (PA30) | I/O, T.S. | ATM Transmit Cell Available (H). When this signal is asserted (High), while the ATM port is enabled, it indicates that the transmit FIFO of the PM5384 is empty and ready to except a new cell. When negated, it may show either that the transmit FIFO is Full or close to Full, depending on PM5384 internal programming.<br>When the ATM port is disabled, this line may be used for any available function of PA30. |
| | FETH1CRS (PA30) | I/O, T.S. | Fast-Ethernet 1 Carrier Sense (H). When this signal is asserted (High), while the Ethernet port is enabled and the DM9161 is in half-duplex mode, it indicates that either the transmit or receive media are non-idle. When the DM9161 is in either full-duplex or repeater operation, it indicates that the receive medium is non-idle.<br>When the Ethernet port is disabled, this line may be used for any available function of PA30. |
| B3 | ATMTSOC (PA29) | I/O, T.S. | ATM Transmit Start Of Cell (H). When this signal is asserted (High) by the PowerQUICC II, while the ATM port is enabled, it indicates to the PM5384 the start of a new ATM cell over ATMTXD(7:0), i.e., the 1’st octet is present there.<br>When the ATM port is disabled, this line may be used for any available function of PA29. |
| | FETH1TXER (PA29) | I/O, T.S. | Fast-Ethernet $^{2}$ 1 Transmit Error (H). When the Ethernet port is enabled, this signal will be asserted (High) by the PowerQUICC II when an error is discovered in the transmit data stream. When the port is operation at 100 Mbps, the DM9161 responds by sending invalid code symbols on the line.<br>When the Ethernet port is disabled, this line may be used for any available function of PA29. |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| B4 | ATMRXEN# (PA28) | I/O, T.S. | ATM Receive Enable (L). When this signal is asserted (Low), while the ATM port is enabled and ATMRFCLK [3] goes high, on octet of data is available at the PM5384's ATMRXD(7:0) lines.<br>When negated while ATMRFCLK goes high data on ATMRXD(7:0) is invalid, however driven.<br>When the ATM port is disabled, this line may be used for any available function for PA28. |
| | FETH1TXEN (PA28) | I/O, T.S. | Fast-Ethernet 1 Transmit Enable (H). The PowerQUICC II will assert (High) this line, to indicate data valid on the FETHTXD(3:0) lines.<br>When the Fast-Ethernet port is disabled, this line may be used for any available function of PA28. |
| B5 | ATMRSOC (PA27) | I/O, T.S. | ATM Receive Start Of Cell (H). When this signal is asserted (High), while the ATM port is enabled, it indicates, that the 1'st octet of data for the received cell is available at the PM5384's ATMRXD(7:0) lines. This line is updated over the rising edge of ATMRFCLK.<br>When the ATM port is disabled, this line is tristated and may be used for any available function for PA27. |
| | FETH1RXDV (PA27) | I/O, T.S. | Fast-Ethernet 1 Receive Data Valid (H). When this signal is asserted (High) while the Fast Ethernet port is enabled and FETHRXCK goes high, it indicates that data is valid on the MII Receive Data lines - FETHRXD(3:0).<br>When the Fast Ethernet port is disabled, this line is tristated and may be used for any available function go PA27. |
| B6 | ATMRCA (PA26) | I/O, T.S. | ATM Receive Cell Available (H). When this signal is asserted (High), while the ATM port is enabled and ATMRFCLK goes high, it indicates that the PM5384's receive FIFO is either full or that there are 4 empty bytes left in it - PM5384 internal programming dependent.<br>When the ATM port is disabled, this line is tristated and may be used for any available function of PA26. |
| | FETH1RXER (PA26) | I/O, T.S. | Fast-Ethernet 1 Receive Error (H). When this signal is asserted (High) by the DM9161, while the Ethernet port is enabled and FETH1RXCK goes high, it indicates that the port is receiving invalid data symbols from the network.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of PA26. |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| B7 | ATMTXD0 (PA25) | I/O, T.S. | ATM Transmit Data (7[4]:0). When the ATM port is enabled, this bus carries the ATM cell octets, written to the PM5384's transmit FIFO. This bus is considered valid only when ATMTXEN# is asserted and are sampled on the rising edge of ATMTFCLK.<br>When the ATM port is disabled, these lines may be used for any available respective function. |
| B8 | ATMTXD1 (PA24) | | |
| B9 | ATMTXD2 (PA23) | | |
| B10 | ATMTXD3 (PA22) | | |
| B11 | ATMTXD4 (PA21) | | |
| | FETH1TXD3 (PA21) | | |
| B12 | ATMTXD5 (PA20) | | |
| | FETH1TXD2 (PA20) | | |
| B13 | ATMTXD6 (PA19) | | |
| | FETH1TXD1 (PA19) | | |
| B14 | ATMTXD7 (PA18) | | |
| | FETH1TXD0 (PA18) | | |
| B15 | ATMRXD7 (PA17) | I/O, T.S. | ATM Receive Data (7[4]:0). When the ATM port is enabled, this bus carries the cell octets, read from the PM5384 receive FIFO. This lines are updated on the rising edge of ATMRFCLK[3].<br>When the ATM port is disabled, these lines are tristated and may be used for any available respective function. |
| | FETH1RXD0 (PA17) | | |
| B16 | ATMRXD6 (PA16) | | |
| | FETH1RXD1 (PA16) | | |
| B17 | ATMRXD5 (PA15) | | |
| | FETH1RXD2 (PA15) | | |
| B18 | ATMRXD4 (PA14) | | |
| | FETH1RXD3 (PA14) | | |
| B19 | ATMRXD3 (PA13) | | |
| B20 | ATMRXD2 (PA12) | | |
| B21 | ATMRXD1 (PA11) | | |
| B22 | ATMRXD0 (PA10) | | |
| B23 | PA9 | I/O, T.S. | PowerQUICC II's Port A (9:0). Parallel I/O or dedicated CPM lines. May be used for any of their available functions. |
| B24 | PA8 | | |
| B25 | | | |
| B26 | | | |
| B27 | | | |
| B28 | | | |
| B29 | PC27 | | |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| B30 | PC26 | | |
| B31 | | | |
| B32 | | | |
| C1 | | | |
| C2 | | | |
| C3 | | | |
| C4 | | | |
| C5 | PB27 | | |
| C6 | PB26 | | |
| C7 | PB25 | | |
| C8 | PB24 | | |
| C9 | RS_RXD2 (PD22) | I/O, T.S. | When RS232 port #2 is enabled, this signal is the receive data line for that port. When this port is disabled, this signal is tristated and may be used to any available alternate function for PD22. |
| C10 | RS_TXD2 (PD21) | I/O, T.S. | When RS232 port #2 is enabled, this signal is the transmit data line for that port. When this port is disabled, this signal may be used to any available alternate function for PD21. |
| C11 | PC9 | | |
| C12 | PD20 | | |
| C13 | | | |
| C14 | | | |
| C15 | FETH1RXDV (PB30) | I/O, T.S. | Fast-Ethernet 1 Receive Data Valid (H). When this signal is asserted (High) while the Fast Ethernet port is enabled and FETHRXCK goes high, it indicates that data is valid on the MII Receive Data lines - FETHRXD(3:0).<br>When the Fast Ethernet port is disabled, this line is tristated and may be used for any available function go PB30. |
| C16 | FETH1RXER (PB28) | I/O, T.S. | Fast-Ethernet 1 Receive Error (H). When this signal is asserted (High) by the DM9161, while the Ethernet port is enabled and FETHRXCK goes high, it indicates that the port is receiving invalid data symbols from the network.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of PB28. |
| C17 | FETH1TXER (PB31) | I/O, T.S. | Fast-Ethernet [5] 1 Transmit Error (H). When the Ethernet port is enabled, this signal will be asserted (High) by the PowerQUICC II when an error is discovered in the transmit data stream. When the port is operation at 100 Mbps, the DM9161 responds by sending invalid code symbols on the line.<br>When the Ethernet port is disabled, this line may be used for any available function of PB31. |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| C18 | FETH1TXEN (PB29) | I/O, T.S. | Fast-Ethernet 1 Transmit Enable (H). The PowerQUICC II will assert (High) this line, to indicate data valid on the FETHTXD(3:0) lines.<br>When the Fast-Ethernet port is disabled, this line may be used for any available function of PB29. |
| C19 | FETH1COL (PB27) | I/O, T.S. | Fast-Ethernet Port 1 Collision Detected (H). When this signal is asserted (High) by the DM9161, while the ethernet port is enabled, it indicates a Collision state over the line. When the DM9161 is in Full-Duplex mode, this line is inactive.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of the PB27. |
| C20 | FETH1CRS (PB26) | I/O, T.S. | Fast-Ethernet 1 Carrier Sense (H). When this signal is asserted (High), while the Ethernet port is enabled and the DM9161 is in half-duplex mode, it indicates that either the transmit or receive media are non-idle. When the DM9161 is in either full-duplex or repeater operation, it indicates that the receive medium is non-idle.<br>When the Ethernet port is disabled, this line may be used for any available function of PB26. |
| C21 | FETH2RXD3 (PB18) | I/O, T.S. | Fast Ethernet 2 Receive Data (3:0). This is the MII receive data bus. The DM9161 drives these lines according to rising edge of FETH2RXCK.<br>When the ethernet port is disabled, these lines are tristated and may be used for any available respective parenthesized function. |
| C22 | FETH2RXD2 (PB19) | | |
| C23 | FETH2RXD1 (PB20) | | |
| C24 | FETH2RXD0 (PB21) | | |
| C25 | FETH2TXD0 (PB22) | I/O, T.S. | Fast Ethernet 2 Transmit Data (3:0). This is the MII transmit data bus. The PowerQUICC II drives these lines according to rising edge of FETH2TXCK.<br>When the ethernet port is disabled, these lines may be used for any available respective function. |
| C26 | FETH2TXD1 (PB23) | | |
| C27 | FETH2TXD2 (PB24) | | |
| C28 | FETH2TXD3 (PB25) | | |
| C29 | ATMRCLK | O, T.S. | ATM Receive Clock. A divide by 8 of the ATM line clock recovered by the ATM receive logic. Provided to assist Circuit Emulation Tool. Enabled only when pin A29 of this connector is either not connected or driven low. Otherwise, Tri-stated. |
| C30 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| C31 | | | |
| C32 | | | |
| D1 | PC31 | I/O, T.S. | PowerQUICC II's Port C (31:22) Parallel I/O lines. May be used to any of their available functions. |
| D2 | PC30 | | |
| D3 | PC29 | | |
| D4 | PC28 | | |
| D5 | | | |
| D6 | | | |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| D7 | PC25 | | |
| D8 | USBCLK | | USB Clock Line. When the USB port is enabled, this line is connected to the USB clock line. When this port is disabled, this signal is tristated and may be used for any available function of PC24. |
| D9 | RS_CD1# (PC23) | I/O, T.S. | RS232 Port 1 Carrier Detect (L). Connected via RS232 transceiver to RS232 DTR1# input, allowing detection of a connected terminal to this port. This line is simply a PI/O input line to the PowerQUICC II.<br>When RS232 Port 1 is disabled, this line is tristated and may be used for any available function of PC23. |
| D10 | FETH1TXCK (PC22) | I/O, T.S. | Fast-Ethernet 1Transmit Clock. When the Ethernet port is enabled, this clock (25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps) is normally extracted from the received data and driven to the PowerQUICC II to qualify out coming transmit data. In Slave mode (not used with this application) this clock should be input to the DM9161.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of PC22 |
| D11 | ATMTFCLK (PC21) | I/O, T.S. | ATM Transmit FIFO Clock. Upon the rising edge of this clock (driven by the PowerQUICC II), while the ATM port is enabled, the cell octets are written to the PM5384's transmit FIFO. This clock samples ATMTXD(7:0), ATMTXPTY, ATMTXEN# and ATMTSOC. When the ATM port is disabled, this line may be used for any available function of PC21. |
| | FETH1RXCK (PC21) | I/O, T.S. | Fast-Ethernet 1Receive Clock. When the Ethernet port is enabled, this clock (25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps) is extracted from the received data and driven to the PowerQUICC II to qualify incoming receive data.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of PC21 |
| D12 | USBOE (PC20) | | USB OE Line. When the USB port is enabled, this line is connected to the OE line of the USB tranceiver. When this port is disabled, this signal is tristated and may be used for any available function of PC20. |
| D13 | | | |
| D14 | | | |
| D15 | FETH2RXCK (PC17) | I/O, T.S. | Fast-Ethernet 1Receive Clock. When the Ethernet port is enabled, this clock (25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps) is extracted from the received data and driven to the PowerQUICC II to qualify incoming receive data.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of PC17 |

## Table 8-4. P1—CPM Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| D16 | FETH2TXCK (PC16) | I/O, T.S. | Fast-Ethernet 2 Transmit Clock. When the Ethernet port is enabled, this clock (25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps) is normally extracted from the received data and driven to the PowerQUICC II to qualify out coming transmit data. In Slave mode (not used with this application) this clock should be input to the DM9161.<br>When the Ethernet port is disabled, this line is tristated and may be used for any available function of PC16 |
| D17 | PC15 | | |
| D18 | PC14 | | |
| D19 | PC13 | I/O, T.S. | PowerQUICC II's Port C 13 Parallel I/O line. May be used to any of its available functions. |
| D20 | PC12 | | |
| D21 | USBRP (PC11) | | USB RP Line. When the USB port is enabled, this line is connected to the receive positive line of the USB tranceiver. When this port is disabled, this signal is tristated and may be used for any available function of PC11. |
| D22 | FETHMDC (PC19) | I/O, T.S. | Fast-Ethernet Port Management Data Clock. This slow clock (S/W generated) qualifies the management data I/O to read / write the DM9161's internal registers.<br>When the Ethernet port is disabled, this line may be used for any available function of PC19. |
| D23 | FETHMDIO (PC18) | I/O, T.S. | Fast-Ethernet Port Management Data I/O. This signal serves as bidirectional serial data line, qualified by FETHMDC, to allow read / write the DM9161's internal registers.<br>When the Ethernet port is disabled, this line may be used for any available function of PC18. |
| D24 | USBRN (PC10) | | USB RN Line. When the USB port is enabled, this line is connected to the receive negative line of the USB tranceiver. When this port is disabled, this signal is tristated and may be used for any available function of PC10. |
| D25 | PC7 | | |
| D26 | PC6 | | |
| D27 | PC5 | | |
| D28 | PC4 | | |
| D29 | | | |
| D30 | RS_CD2# (PC8) | I/O, T.S. | RS232 Port 2 Carrier Detect (L). Connected via RS232 transceiver to RS232 DTR2# input, allowing detection of a connected terminal to this port. This line is simply a PI/O input line to the PowerQUICC II.<br>When RS232 Port 2 is disabled, this line is tristated and may be used for any available function of PC8. |

**Table 8-4. P1—CPM Expansion Connector (continued)**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| D31 | PC1 | | |
| D32 | PC0 | | |

1 The functions in parenthesis, are PowerQUICC II's parallel I/Os.
2 For that matter, both 100-Base-T and 10-Base-T.
3 Normally connected to ATMTFCLK on the ADS.
4 MS bit.
5 For that matter, both 100-Base-T and 10-Base-T.

### 8.1.5 P9, P11, P14, P22, P17, P5, P20, P18, P15, P24,P12 Logic Analyzer MICTOR Connectors

These connectors made by AMPare are 38-pin, SMT, high density, and matched impedance. They contain the MPC8272 60X bus, 60X system and memory controller signals, unbuffered. The pinout of these connectors is shown in the schematics. For signal description of these connectors, see the *MPC8272 PowerQUICC II" Family Reference Manual*.

### 8.1.6 P26, P28, P29—PCI Connectors

These are 2 X 62 , 3.3-V keyed, 32 bit PCI connectors. The pinout of each connector is available in Table 8-5.

For signal descriptions for these connectors, see the PCI v2.2 standard.

**Table 8-5. P26, P28, P29—PCI Connectors**

| Pin Number | Side B | Comments | Side A | Comments |
|---|---|---|---|---|
| 1 | -12V | Not connected | TRST# | |
| 2 | TCK | | +12V | |
| 3 | Ground | | TMS | |
| 4 | TDO | | TDI | |
| 5 | +5V | | +5V | |
| 6 | +5V | | INTA# | |
| 7 | INTB# | Not Connected | INTC# | Not Connected |
| 8 | INTD# | Not Connected | +5V | |
| 9 | PRSNT1# | Connected to GND | Reserved | Not Connected |
| 10 | Reserved | Not Connected | +3.3V(I/O) | |

## Table 8-5. P26, P28, P29—PCI Connectors (continued)

| Pin Number | Side B | Comments | Side A | Comments |
|---|---|---|---|---|
| 11 | PRSNT2# | Connected to GND | Reserved | Not Connected |
| 12 | CONNECTOR KEY | 3.3 volt key | CONNECTOR KEY | 3.3 volt key |
| 13 | CONNECTOR KEY | 3.3 volt key | CONNECTOR KEY | 3.3 volt key |
| 14 | Reserved | Not Connected | 3.3Vaux | Not Connected |
| 15 | Ground | | RST# | |
| 16 | CLK | | +3.3V (I/O) | |
| 17 | Ground | | GNT# | |
| 18 | REQ# | | Ground | |
| 19 | +3.3V (I/O) | | PME# | Not Connected |
| 20 | AD[31] | | AD[30] | |
| 21 | AD[29] | | +3.3V | |
| 22 | Ground | | AD[28] | |
| 23 | AD[27] | | AD[26] | |
| 24 | AD[25] | | Ground | |
| 25 | +3.3V | | AD[24] | |
| 26 | C/BE[3]# | | IDSEL | |
| 27 | AD[23] | | +3.3V | |
| 28 | Ground | | AD[22] | |
| 29 | AD[21] | | AD[20] | |
| 30 | AD[19] | | Ground | |
| 31 | +3.3V | | AD[18] | |
| 32 | AD[17] | | AD[16] | |
| 33 | C/BE[2]# | | +3.3V | |
| 34 | Ground | | FRAME# | |
| 35 | IRDY# | | Ground | |
| 36 | +3.3V | | TRDY# | |
| 37 | DEVSEL# | | Ground | |
| 38 | Ground | | STOP# | |
| 39 | LOCK# | Not Connected | +3.3V | |
| 40 | PERR# | | SDONE | Not Connected |

**Table 8-5. P26, P28, P29—PCI Connectors (continued)**

| Pin Number | Side B | Comments | Side A | Comments |
|---|---|---|---|---|
| 41 | +3.3V | | SBO# | Not Connected |
| 42 | SERR# | | Ground | |
| 43 | +3.3V | | PAR | |
| 44 | C/BE[1]# | | AD[15] | |
| 45 | AD[14] | | +3.3V | |
| 46 | Ground | | AD[13] | |
| 47 | AD[12] | | AD[11] | |
| 48 | AD[10] | | Ground | |
| 49 | M66EN | Coupled to GND, using a 0.01uF capacitor | AD[09] | |
| 50 | Ground | | Ground | |
| 51 | Ground | | Ground | |
| 52 | AD[08] | | C/BE[0]# | |
| 53 | AD[07] | | +3.3V | |
| 54 | +3.3V | | AD[06] | |
| 55 | AD[05] | | AD[04] | |
| 56 | AD[03] | | Ground | |
| 57 | Ground | | AD[02] | |
| 58 | AD[01] | | AD[00] | |
| 59 | +3.3V (I/O) | | +3.3V (I/O) | |
| 60 | ACK64# | Not Connected | REQ64# | Not Connected |
| 61 | +5V | | +5V | |
| 62 | +5V | | +5V | |

### 8.1.7 P31 - ATX Power Supply Connector

This standard ATX form factor power connector is described in Table 8-6.

**Table 8-6. P31—ATX Power Supply Connector**

| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 1 | +3.3VDC | 11 | +3.3VDC-Sense |

**Table 8-6. P31—ATX Power Supply Connector (continued)**

| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 2 | +3.3VDC | 12 | -12VDC |
| 3 | Groung | 13 | Groung |
| 4 | +5VDC | 14 | $\overline{\text{Power_On}}$ |
| 5 | Groung | 15 | Groung |
| 6 | +5VDC | 16 | Groung |
| 7 | Groung | 17 | Groung |
| 8 | Power_OK | 18 | -5VDC |
| 9 | +5VStand_By | 19 | +5VDC |
| 10 | +12VDC | 20 | +5VDC |

## 8.1.8 P3,P4,P25—Mach/Lattice ISP Connector

This is a 10-pin generic 0.100" pitch header connector that provides In System Programming (ISP) capability for Latice-made programmable logic on board. The pinout of P25 is shown in Table 8-7.

**Table 8-7. P3—Lattice ISP Connector**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| 1 | ISPTCK | I | ISP Test port clock. This clock shifts in / out data to / from the programmable logic JTAG chain. |
| 2 | N.C. | - | Not connected. |
| 3 | ISPTMS | I | ISP test mode select. This signal qualified with ISPTCK, changes the state of the prog. logic JTAG machine. |
| 4 | GND | O | Digital GND. Main GND plane. |
| 5 | ISPTDI | I | ISP transmit data In. This is the prog. logic's JTAG serial data input, sampled on the rising edge of TCK. |
| 6 | VCC | O | 5V power supply bus. |
| 7 | ISPTDO | O | ISP transmit data output. This the prog. logic's JTAG serial data output driven by falling edge of TCK. |
| 8 | GND | O | Digital GND. Main GND plane. |
| 9 | N.C. | - | Not connected |
| 10 | N.C. | - | Not connected |

### 8.1.9 P2—System Expansion Connector

P2 is a 128-pin, $90^0$, DIN 41612 connector that provides a minimal system I/F required to interface various types of communication transceivers. This connector contains 16-bit (lower PPC bus) address lines, 16-bit (higher PPC bus) data lines, plus useful GPCM and UPM control lines. The pinout of P2 is shown in Table 8-8.

**Table 8-8. P2—System Expansion Connector**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| A1 | EXPA16 | O | Expansion Address (16 $^1$:31). This is a Latched-Buffered version of the PowerQUICC II's PPC Address lines (16:31), provided for external tool connection. To avoid reflection these lines are series terminated with 43 Ω resistors. |
| A2 | EXPA17 | | |
| A3 | EXPA18 | | |
| A4 | EXPA19 | | |
| A5 | EXPA20 | | |
| A6 | EXPA21 | | |
| A7 | EXPA22 | | |
| A8 | EXPA23 | | |
| A9 | EXPA24 | | |
| A10 | EXPA25 | | |
| A11 | EXPA26 | | |
| A12 | EXPA27 | | |
| A13 | EXPA28 | | |
| A14 | EXPA29 | | |
| A15 | EXPA30 | | |
| A16 | EXPA31 | | |
| A17 | EXP12V | O | These can be connected to the positive 12-V source from the PCI edge connector thru J3. This line is fused by a 0.5A ressetable poly-switch. |
| A18 | | | |
| A19 | N.C. | - | Not Connected. |
| A20 | EXP3.3V | O | 3.3V Power Out. These lines are connected to the main 3.3V plane of the PowerQUICC IIPCIAI-ADS, this, to provide 3.3V power where necessary for external tool connected. |
| A21 | | | |
| A22 | | | |
| A23 | | | |
| A24 | | | |
| A25 | N.C. | - | Not connected. |

## Table 8-8. P2—System Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| A26 | EXPVCC | O | 5-V Supply. Connected to ADS's 5V VCC plane. Provided as power supply for external tool. |
| A27 | | | |
| A28 | | | |
| A29 | | | |
| A30 | | | |
| A31 | | | |
| A32 | | | |
| B1 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| B2 | | | |
| B3 | | | |
| B4 | TSTAT0 | I | Tool Status ($0^1$:7). These lines may be driven by an external tool to be read via BCSR2 of the ADS. These lines are pulled-up on the ADS, by 10 KΩ resistors. See also Table 10-38. "BCSR2 Description" on page 23. |
| B5 | TSTAT1 | | |
| B6 | TSTAT2 | | |
| B7 | TSTAT3 | | |
| B8 | TSTAT4 | | |
| B9 | TSTAT5 | | |
| B10 | TSTAT6 | | |
| B11 | TSTAT7 | | |
| B12 | TOOLREV0 | I | Tool Revision ($0^1$:3). These lines should be driven by an external tool with the Tool Revision Code, to be read via BCSR2 of the ADS. These lines are pulled-up on the ADS, by 10 KΩ resistors. See also Table 10-38. "BCSR2 Description" on page 23. |
| B13 | TOOLREV1 | | |
| B14 | TOOLREV2 | | |
| B15 | TOOLREV3 | | |
| B16 | EXTOLI0 | I | ExternalTool Identification ($0^1$:3). These lines should be driven by an external tool with the Tool Identification Code, to be read via BCSR2 of the ADS. These lines are pulled-up on the ADS, by 10 KΩ resistors. See also Table 10-38. "BCSR2 Description" on page 23 |
| B17 | EXTOLI1 | | |
| B18 | EXTOLI2 | | |
| B19 | EXTOLI3 | | |
| B20 | N.C. | - | Not connected |
| B21 | EXP3.3V | O | 3.3V Power Out. These lines are connected to the main 3.3V plane of the PowerQUICC IIPCIAI-ADS, this, to provide 3.3V power where necessary for external tool connected. |
| B22 | | | |
| B23 | | | |
| B24 | | | |

## Table 8-8. P2—System Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| B25 | N.C. | - | Not Connected |
| B26 | EXPVCC | O | 5V Supply. Connected to ADS's 5V VCC plane. Provided as power supply for external tool. |
| B27 | | | |
| B28 | | | |
| B29 | | | |
| B30 | | | |
| B31 | | | |
| B32 | | | |
| C1 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| C2 | CLK8 | O | Buffered System Clock.. |
| C3 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| C4 | BTOOLCS1# | O | Buffered Tool Chip Select 1 (L). This is a buffered PowerQUICC II's CS6# line, reserved for an external tool. |
| C5 | BTOOLCS2# | O | Buffered Tool Chip Select 2 (L). This is a buffered PowerQUICC II's CS7# line, reserved for an external tool. |
| C6 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| C7 | ATMEN# | O | ATM Port Enable (L). This line enables the ATM port UNI's output lines towards the PowerQUICC II. An external tool, using the same pins as does the ATM port should consult this signal before driving the same lines. **Failure to do so might result in permanent damage to the PM5384 ATM UNI.** |
| C8 | ATMRST# | O | ATM Port Reset (L). This signal resets the ATM UNI (PM5384). An external tool may use this signal to its benefit. |
| C9 | FETHRST# | O | Ethernet Port Reset (L). This signal resets the DM9161 Ethernet transceiver. An external tool may use this signal to its benefit. |
| C10 | HRESET# | I/O, O.D. | PowerQUICC II's Hard Reset (L). When asserted by an external H/W, generates Hard-Reset sequence for the PowerQUICC II. During that sequence, asserted by the PowerQUICC II for 512 system clocks. Pulled Up on the ADS using a 1KΩ resistor. When driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so might result in permanent damage to the PowerQUICC II and / or to ADS logic.** |
| C11 | IRQ6# | I | Interrupt Request 6 (L). Connected to PowerQUICC II's DP6/CSE0/IRQ6# signal. Pulled up on the ADS with a 10 KΩ resistor. This line is shared with the ATM UNI's interrupt line and therefore, when driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so may result in permanent damage to the PowerQUICC II or to ADS logic.** |

## Table 8-8. P2—System Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| C12 | IRQ7# | I | Interrupt Request 7 (L). Connected to PowerQUICC II's DP7/CSE1/IRQ7# signal. Pulled up on the ADS with a 10 KΩ resistor. This line is shared with the Fast Ethernet transceiver's interrupt line and therefore, when driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so might result in permanent damage to the PowerQUICC II and / or to ADS logic.** |
| C13 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| C14 | EXPD0 | I/O, T.S. | Expansion data ($0^1$:15). This double-buffered version of the PPC bus D(0:15) lines is controlled by on-board logic. These lines are driven only if BTOOLCS1# or BTOOLCS2# are asserted. Otherwise they are tristated.<br>The direction of these lines is determined by buffered BCTL0, in function of W/R#. |
| C15 | EXPD1 | | |
| C16 | EXPD2 | | |
| C17 | EXPD3 | | |
| C18 | EXPD4 | | |
| C19 | EXPD5 | | |
| C20 | EXPD6 | | |
| C21 | EXPD7 | | |
| C22 | EXPD8 | | |
| C23 | EXPD9 | | |
| C24 | EXPD10 | | |
| C25 | EXPD11 | | |
| C26 | EXPD12 | | |
| C27 | EXPD13 | | |
| C28 | EXPD14 | | |
| C29 | EXPD15 | | |
| C30 | N.C. | - | Not connected |
| C31 | | | |
| C32 | | | |
| D1 | GND | O | Digital ground. Connected to main GND plane of the ADS. |
| D2 | | | |
| D3 | | | |

## Table 8-8. P2—System Expansion Connector (continued)

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| D4 | EXPWE0# | O | Expansion Write Enable (0:1) (L). These are buffered GPCM Write Enable lines (0:1). They are meant to qualify writes to GPCM controlled 8/16 data bus width memory devices. This to provide eased access to various communication transceivers. EXPWE0# controls EXPD(0:7) while EXPWE1# controls EXPD(8:15). These lines may also function as UPM controlled Byte Select Lines, which allow control over almost any type of memory device. |
| D5 | EXPWE1# | | |
| D6 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| D7 | EXPGL0# | O | Expansion General Purpose Lines (0:5) (L). These are buffered GPL(0:5)# lines which assist UPM control over memory device if necessary. These are output only signals and therefore, do not support H/W controlled UPM waits. |
| D8 | EXPGL1# | | |
| D9 | EXPGL2# | | |
| D10 | EXPGL3# | | |
| D11 | EXPGL4# | | |
| D12 | EXPGL5# | | |
| D13 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| D14 | EXPALE | O | Expansion address latch enable (H). This is the buffered PowerQUICC II‘s ALE, provided for expansion board’s use. |
| D15 | EXPCTL0 | O | Expansion control line 0. This line is a buffered version of PowerQUICC II’s BCTL0 (Bus Control Line 0) which serves as W/R#, provided for expansion board’s use. |

**Table 8-8. P2—System Expansion Connector (continued)**

| Pin No. | Signal Name | Attribute | Description |
|---|---|---|---|
| D16 | GND | O | Digital Ground. Connected to main GND plane of the ADS. |
| D17 | | | |
| D18 | | | |
| D19 | | | |
| D20 | | | |
| D21 | | | |
| D22 | | | |
| D23 | | | |
| D24 | | | |
| D25 | | | |
| D26 | | | |
| D27 | | | |
| D28 | | | |
| D29 | | | |
| D30 | | | |
| D31 | | | |
| D32 | | | |

[1]MS Bit.

## 8.1.10 P16, P19—USB Connectors

These are four pins standard USB connectors type-A and Type-B. The pinout is shown in Table 8-9.

**Table 8-9. P16, P19—USB Connectors**

| Pin No. | Signal Name | Description |
|---|---|---|
| 1 | 5V Power | Power line of the USB cable |
| 2 | D- | Twisted-pair transmit data negative |
| 3 | D+ | Twisted-pair receive data positive |
| 4 | GND | Ground connection |

## 8.2 Programmable Logic Equations

The two programmable logic devices on board are the following:

1. U3 - BCSR and PCI interrupt controller
2. U5 - Power switch debounce

### 8.2.1 U3—BCSR Code

```
MODULE MPC8272ADS_BCSR

TITLE 'MPC8272ADS control status register'

"*******************************************************************************
"* This file (Prototype) is based on the PQ2FADS-VR (08/21/03):
"*******************************************************************************
"*******************************************************************************
"* In this file (Prototype) the following changes were made  (03/06/03):
"* - Adjustments for VR board were done - PCI only support in config Word (LBPC).
"*******************************************************************************
"*******************************************************************************
"* In this file (Prototype) the following changes were made  (02/03/03):
"* - All BCSR registers are reset by HRESET and not PORESET.
"*******************************************************************************
"*******************************************************************************
"* In this file (Prototype) the following changes were made  (12/03/02):
"* - Added support for LBPC in Hard Reset Config Word (determined by external
"*   signal nPCI_Mode)
"*******************************************************************************
"*******************************************************************************
"* In this file (Prototype) the following changes were made  (11/01/02):
"* - Added support for USB, Second Fast Ethernet, PARITY option on 60x,
"*   CPM MUX control.
"*******************************************************************************
"*******************************************************************************
"* In this file (Prototype) the following changes were made  (07/15/02):
"* - Added support for a second Fast Ethernet PHY.
"* - Removed support for fast down-load through JTAG.
"*******************************************************************************

"*******************************************************************************
"* Device declaration.                                          *
"*******************************************************************************

"*******************************************************************************
“* EXTERNALS
"*******************************************************************************

"*******************************************************************************
"* Pins declaration.
*******************************************************************************
"* System i/f pins
"*******************************************************************************
SYSCLK              PIN   124;

IntContCs_B          PIN   48; " PCI INterrupt Controller CS
```

```
BrdContRegCs_B          PIN   47; " BCSR CS
DVal_B                PIN   53;
R_B_W                  PIN   46; " BCTL0 signal
BCTL1                  PIN   ;   " Alternate Buffers Enable source
A7                   PIN   69; " for flash support
A8                   PIN   68; " for flash support
A27                   PIN   15;
A28                   PIN   12;
A29                   PIN   11;
D0                   PIN   75istype 'com' ;
D1                   PIN   22istype 'com' ;
D2                   PIN   132istype 'com' ;
D3                   PIN   77istype 'com' ;
D4                   PIN   16istype 'com' ;
D5                   PIN   142istype 'com' ;
D6                   PIN   60istype 'com' ;
D7                   PIN   87istype 'com' ;
D8                   PIN   66istype 'com' ;
D9                   PIN   72istype 'com' ;
D10                   PIN   70istype 'com' ;
D11                   PIN   39istype 'com' ;

"*****************************************************************************
"* Board Control Pins. Read/Write.
"*****************************************************************************

SignaLamp0_B            PIN    44istype 'reg,buffer' ; " status lamp 0 for misc s/w visual
SignaLamp1_B            PIN    38istype 'reg,buffer' ; " status lamp 1 for misc s/w visual
AtmEn_B                PIN    134istype 'reg,buffer' ; " atm uni enable
FCC1Off_B              PIN    104istype 'com' ; " FCC1 (ATM/Ethernet1) off
FCC1On_B               PIN    105istype 'com' ; " FCC1 (ATM/Ethernet1) on
AtmSinglePHY_B          PIN    4istype 'reg,buffer' ; " UTOPIA Single PHY enable
AtmMultiPHY_B           PIN    58istype 'com' ; " UTOPIA Multi PHY enable
AtmRst_B NODE   istype 'reg,buffer' ; " atm uni reset bit
AtmRstOut_B             PIN    76istype 'com' ; " atm uni reset driven by register
                                  " or by HRESET_B
USBEn_B PIN    5istype 'reg,buffer' ; " USB enable
USBDis_B        PIN    86istype 'com' ; " USB disable
USBLowSpd_B PIN    133istype 'com' ; " USB Low Speed Select
"USBVccO        PIN    7istype 'reg,buffer' ;  USB Line Voltage Select
USBVccOut PIN    7istype 'com' ; " USB Line Voltage Select
USBVccO         NODE    istype 'reg,buffer' ; " USB Line Voltage Select

FEthEn1_B              PIN    9istype 'reg,buffer' ; " fast ethernet trans. 1 enable
FEthDis1_B             PIN    23istype 'com' ; " fast ethernet trans. 1 Disable
FEthEn2_B              PIN    40istype 'reg,buffer' ; " fast ethernet trans. 2 enable
FEthDis2_B             PIN    79istype 'com' ; " fast ethernet trans. 2 Disable
FEthRst1_B        NODE   istype 'reg,buffer' ; " fast ethernet trans. 1 reset bit
FEthRstOut1_B           PIN    139istype 'com' ; " fast eth trans 1 reset driven by
                                  " register or by HRESET_B
FEthRst2_B        NODE   istype 'reg,buffer' ; " fast ethernet trans. 2 reset bit
FEthRstOut2_B           PIN    110istype 'com' ; " fast eth trans 2 reset driven by
                                  " register or by HRESET_B
MIIOn_B                PIN    115istype 'com' ; " Ethernet 1/2 On to enable MDIO/MDC
MIIOff_B               PIN    113istype 'com' ; " Ethernet 1/2 Off to disable MDIO/MDC

FEth1PLLOn_B PIN 131 ; " FEth1 TxCLK clock buffer PLL On for 100Base-T
MII1PLLOff_B PIN 129istype'com' ; " FEth1 TxCLK buffer PLL Off - 10Base-T
FEth2PLLOn_B PIN 127 ; " FEth2 TxCLK clock buffer PLL On for 100Base-T
```

```
MII2PLLOff_B PIN 117istype'com' ; " FEth2 TxCLK buffer PLL Off - 10Base-T

RS232En1_B          PIN   32istype 'reg,buffer' ; " RS232 port 1 enable
RS232Dis1_B         PIN   56istype 'com' ; " RS232 port 1 Disable
RS232En2_B          PIN   3istype 'reg,buffer' ; " RS232 port 2 enable
RS232Dis2_B         PIN   8istype 'com' ; " RS232 port 2 Disable

PCI_Mode_B          PIN   19 ;  " Local Bus PCI Select

ModckH0 PIN   67 ; " MODCKH0
ModckH1 PIN   65 ; " MODCKH1
ModckH2 PIN   61 ; " MODCKH2
ModckH3 PIN   59 ; " MODCKH3

PCI_IRQ_B           PIN   100istype 'com,buffer' ; " PCI Interrupt to PQ2 (o.d.)

PCI_INTA_B          PIN   97 ; " PCI Interrupt from PCI card
PCI_INTB_B          PIN   126 ; " PCI Interrupt from PCI card
PCI_INTC_B          PIN   125 ; " PCI Interrupt from PCI card
PCI_INTD_B          PIN   120 ; " PCI Interrupt from PCI card

"*****************************************************************************
"* Board Status Registers Chip-Selects
"*****************************************************************************
Bcsr2Cs_B           PIN   89istype 'com' ;
Bcsr4Cs_B           PIN   81istype 'com' ;

"*****************************************************************************
"* Flash/EEPROM Associated Pins.
"*****************************************************************************

F_PD1               PIN   57 ;
F_PD2               PIN   55 ;
F_PD3               PIN   45 ;
F_PD4               PIN   43 ;

Cs0_B               PIN   54 ; " flash/eeprom chip-select input
Cs4_B               PIN   94 ; " eeprom/flash chip-select input

EEpromCs_B          PIN   137istype 'com' ; " EEPROM chip-select

FlashCs1_B          PIN   144istype 'com' ; " Flash bank1 chip-select
FlashCs2_B          PIN   138istype 'com' ; " Flash bank2 chip-select
FlashCs3_B          PIN   143istype 'com' ; " Flash bank3 chip-select
FlashCs4_B          PIN   140istype 'com' ; " Flash bank4 chip-select

"*****************************************************************************
"* PM5384 ATM UNI Associated Pins.
"*****************************************************************************

AtmUniCsIn_B        PIN   119 ;
AtmUniCsOut_B       PIN   62istype 'com' ; " remove if short of pins

"*****************************************************************************
"* Reset & Interrupt Logic Pins.
"*****************************************************************************

PORIn_B             PIN   41 ;
```

```
"RstConf_B              PIN    istype 'com';  Hard Reset master select.

Rst0                PIN   33 ; " connected to N.C. of Reset P.B.
Rst1                PIN   31 ; " connected to N.O. of Reset P.B.

HardReset_B           PIN   18istype 'com' ; " Actual hard reset output (O.D.)
SoftReset_B           PIN   17istype 'com' ; " Actual soft reset output (O.D.)

Abr0                PIN   30 ; " connected to N.C. of Abort P.B.
Abr1                PIN   29 ; " connected to N.O. of Abort P.B.

NMIEn        NODE   istype 'com' ; " enables T.S. NMI pin
NMI_B                PIN   20istype 'com' ; " Actual NMI pin (O,O.D.)

"*****************************************************************************
"* Data Buffers Enables and Reset configuration support
"*****************************************************************************
TEA_B                PIN   102 ; "  Transfer Error Acknowledge.
DataBufEn_B           PIN   85istype 'com,invert' ; " data buffer enable

ToolCs1_B             PIN   27 ; " comm tool cs line 1.
ToolCs2_B             PIN   21 ; " comm tool cs line 2.

ToolDataBufEn_B         PIN   91istype 'com,invert' ; " tool data buffer enable

"*****************************************************************************
"* Hard Reset Configuration Logic
"*****************************************************************************
boot_device_B PIN   118 ; " selects EEPROM/FLASH_B as boot device
bcsrConfEn PIN   93 ; " selects Hard Reset Configuration Source
 " BCSR or EEPROM/FLASH.
"*****************************************************************************
"* Auxiliary Pins.
"*****************************************************************************
"* System Hard Reset Configuration.
"*****************************************************************************

DataOeNODE istype 'com' ; " data bus output enable on read.
DataPCIOeNODE istype 'com' ; " data bus output enable on PCI read.

"*****************************************************************************
"* Control Register Enable Protection.
"*****************************************************************************

"*****************************************************************************
"* Reset & Interrupt Logic Pins.
"*****************************************************************************
RstDeb1NODE istype 'keep,com' ; " reset push button debouncer
AbrDeb1NODE istype 'keep,com' ; " abort push button debouncer

HardResetEnNODE istype 'com' ; " enables T.S. hard reset pin
SoftResetEnNODE istype 'com' ; " enables T.S. soft reset pin

"*****************************************************************************
"* data buffers enable.
"*****************************************************************************
SyncHardReset_B  NODE istype 'reg,buffer' ; " synchronized hard reset
DSyncHardReset_B  NODE istype 'reg,buffer' ; " double synchronized hard reset
HoldOffCnt2,
```

```
HoldOffCnt1,
HoldOffCnt0     NODE istype 'reg,buffer' ; " data buf en hold-off counter
HoldOffTc       NODE istype 'com' ; " terminal count for that counter

"*****************************************************************************
"* Power On Reset
"*****************************************************************************

S_PORIn_B        NODE  istype 'reg,buffer' ; " synced pon reset.

"*****************************************************************************
"* PCI Interrupt Register.
"*****************************************************************************

Slot0IntANODE istype 'reg,invert' ;  " PCI Slot 0 Interrupt A
Slot0IntBNODE istype 'reg,invert' ;  " PCI Slot 0 Interrupt B
Slot0IntCNODE istype 'reg,invert' ;  " PCI Slot 0 Interrupt C
Slot0IntDNODE istype 'reg,invert' ;  " PCI Slot 0 Interrupt D
Slot1IntANODE istype 'reg,invert' ;  " PCI Slot 1 Interrupt A
Slot1IntBNODE istype 'reg,invert' ;  " PCI Slot 1 Interrupt B
Slot1IntCNODE istype 'reg,invert' ;  " PCI Slot 1 Interrupt C
Slot1IntDNODE istype 'reg,invert' ;  " PCI Slot 1 Interrupt D
Slot2IntANODE istype 'reg,invert' ;  " PCI Slot 2 Interrupt A
Slot2IntBNODE istype 'reg,invert' ;  " PCI Slot 2 Interrupt B
Slot2IntCNODE istype 'reg,invert' ;  " PCI Slot 2 Interrupt C
Slot2IntDNODE istype 'reg,invert' ;  " PCI Slot 2 Interrupt D

"*****************************************************************************
"* PCI Interrupt Mask Register.
"*****************************************************************************

Slot0IntAMaskNODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt A Mask
Slot0IntBMaskNODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt B Mask
Slot0IntCMaskNODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt C Mask
Slot0IntDMaskNODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt D Mask
Slot1IntAMaskNODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt A Mask
Slot1IntBMaskNODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt B Mask
Slot1IntCMaskNODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt C Mask
Slot1IntDMaskNODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt D Mask
Slot2IntAMaskNODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt A Mask
Slot2IntBMaskNODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt B Mask
Slot2IntCMaskNODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt C Mask
Slot2IntDMaskNODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt D Mask

"*****************************************************************************
"* PCI Interrupt Request to PQ2.
"*****************************************************************************

PCI_InterruptNODE istype 'com' ;  " generated Interrupt to PQ2

"*****************************************************************************
"* USB Speed Select.
"*****************************************************************************

USBHiSpd_BNODE istype 'reg,buffer' ; " USB Hi Speed

"*****************************************************************************
"* Misceleneous.
"*****************************************************************************
```

```
KeepPinsConnected  NODE istype 'com' ;
"*******************************************************************************
"*  EQUATIONS *
"*******************************************************************************
H, L, X, Z = 1, 0, .X., .Z. ;
C, D, U    = .C., .D., .U. ;
"*******************************************************************************
"* SIMULATION = 1 ;
"*******************************************************************************
"* Signal groups
"*******************************************************************************

Add = [A27..A29] ;

Data = [D0..D7] ;

DataPCI = [D0..D11] ;

ContReg = [SignaLamp0_B,
           SignaLamp1_B,
           AtmEn_B,
           AtmRst_B,
  AtmSinglePHY_B,
           FEthEn1_B,
  FEthRst1_B,
           FEthEn2_B,
  FEthRst2_B,
           RS232En1_B,
           RS232En2_B,
           USBEn_B,
  USBHiSpd_B,
  USBVccO] ;

ReadBcsr0 = [0,
             0,
             0,
             0,
             0,
             SignaLamp0_B,
             SignaLamp1_B] ;

ReadBcsr1 = [bcsrConfEn,
             boot_device_B,
             AtmEn_B,
             AtmRst_B.fb,
             FEthEn1_B,
             FEthRst1_B.fb,
             RS232En1_B,
             RS232En2_B] ;

ReadBcsr3 = [USBEn_B,
             USBHiSpd_B,
             USBVccO,
             FEthEn2_B,
             FEthRst2_B.fb,
             0,
    AtmSinglePHY_B,
```

```
            PCI_Mode_B];

DrivenContReg = [SignaLamp0_B,
                 SignaLamp1_B,
 AtmEn_B,
         AtmSinglePHY_B,
                 FEthEn1_B,
             FEthEn2_B,
                 RS232En1_B,
                 RS232En2_B,
                 USBEn_B] ;
"           USBVccO] ;

ClockedContReg = [SignaLamp0_B,
                  SignaLamp1_B,
  AtmEn_B,
                  AtmRst_B,
          AtmSinglePHY_B,
                  FEthEn1_B,
                  FEthEn2_B,
                  FEthRst1_B,
                  FEthRst2_B,
                  RS232En1_B,
                  RS232En2_B,
                  USBEn_B,
          USBHiSpd_B,
          USBVccO] ;

IntReg     = [Slot0IntA,
      Slot0IntB,
      Slot0IntC,
      Slot0IntD,
       Slot1IntA,
       Slot1IntB,
      Slot1IntC,
       Slot1IntD,
      Slot2IntA,
      Slot2IntB,
       Slot2IntC,
       Slot2IntD] ;

IntMaskReg = [Slot0IntAMask,
      Slot0IntBMask,
      Slot0IntCMask,
      Slot0IntDMask,
       Slot1IntAMask,
       Slot1IntBMask,
      Slot1IntCMask,
       Slot1IntDMask,
      Slot2IntAMask,
      Slot2IntBMask,
       Slot2IntCMask,
       Slot2IntDMask] ;


ToolCs =         [ToolCs1_B,ToolCs2_B] ;
FlashCsOut =     [FlashCs4_B,FlashCs3_B,FlashCs2_B,FlashCs1_B] ;
Reset =  [HardReset_B,SoftReset_B] ;
ResetEn =  [HardResetEn,SoftResetEn] ;
```

```
TransRst =  [AtmRstOut_B,FEthRstOut1_B,FEthRstOut2_B] ;
Rst =       [Rst1,Rst0] ;
Abr =       [Abr1,Abr0] ;
Debounce =  [RstDeb1,AbrDeb1] ;
SyncReset =    [SyncHardReset_B,DSyncHardReset_B] ;
RstCause =  [PORIn_B,Rst1,Rst0,Abr1,Abr0] ;
HoldOffCnt =   [HoldOffCnt2,HoldOffCnt1,HoldOffCnt0] ;
F_PD =      [F_PD4, F_PD3, F_PD2, F_PD1] ;
Cs =  [Cs0_B,Cs4_B,BrdContRegCs_B,IntContCs_B,AtmUniCsIn_B,ToolCs1_B,ToolCs2_B] ;
BufEn =     [DataBufEn_B,ToolDataBufEn_B] ;
ConfAdd =     [A27,A28];


CfgByte0 = [0,0,0,0,1,1,0,0];
CfgByte1 = [0,1,1,1,0,0,1,0];
CfgByte2 = [0,0,1,1,0,1,1,0];
CfgByte3 = [0,0,0,0,ModckH0,ModckH1,ModckH2,ModckH3];

"*****************************************************************************
"* Power On Reset definitions
"*****************************************************************************
PON_RESET_ACTIVE = 0 ;

PON_RESET = (S_PORIn_B.fb == PON_RESET_ACTIVE) ;

"*****************************************************************************
"* Register Access definitions
"*****************************************************************************
BCSR0_ADD = 0 ;
BCSR1_ADD = 1 ;
BCSR2_ADD = 2 ;
BCSR3_ADD = 3 ;
BCSR4_ADD = 4 ;

VGR_WRITE_BCSR_0 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & !A28 &!A29) ;
VGR_WRITE_BCSR_1 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & !A28 & A29);
VGR_WRITE_BCSR_2 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & A28 & !A29) ;
VGR_WRITE_BCSR_3 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & A28 & A29) ;
VGR_WRITE_BCSR_4 = (!BrdContRegCs_B & !DVal_B & R_B_W & A27 & !A28 & !A29) ;

VGR_READ_BCSR_0 = (!BrdContRegCs_B & !R_B_W & !A27 & !A28 & !A29) ;
VGR_READ_BCSR_1 = (!BrdContRegCs_B & !R_B_W & !A27 & !A28 & A29) ;
VGR_READ_BCSR_2 = (!BrdContRegCs_B & !R_B_W & !A27 & A28 & !A29) ;
VGR_READ_BCSR_3 = (!BrdContRegCs_B & !R_B_W & !A27 & A28 & A29) ;
VGR_READ_BCSR_4 = (!BrdContRegCs_B & !R_B_W & A27 & !A28 & !A29) ;

"*****************************************************************************
"*****************************************************************************
"* BCSR 0 definitions.
"*****************************************************************************
"*****************************************************************************
SIGNAL_LAMP_ON = 0 ;

"***********************************************
"******* Power On Defaults Assignments
"***********************************************
SIGNAL_LAMP0_PON_DEFAULT = !SIGNAL_LAMP_ON ;
SIGNAL_LAMP1_PON_DEFAULT = !SIGNAL_LAMP_ON ;
```

```
"*****************************************
"******* Data Bits Assignments
"*****************************************

SIGNAL_LAMP0_DATA_BIT = [D6] ;
SIGNAL_LAMP1_DATA_BIT = [D7] ;

"*****************************************************************************
"*****************************************************************************
"* BCSR 1 definitions.
"*****************************************************************************
"*****************************************************************************
BCSR_BOOT = 0 ;" bcsrConfEn = 0 Hard Reset Conf Word from BCSR
MEMORY_BOOT = 1 ;" bcsrConfEn = 1 Hard Reset Conf from EEPROM/FLASH
FLASH_BOOT = 0 ;" boot_device_B = 0
EEPROM_BOOT = 1 ;" boot_device_B = 1
ATM_ENABLED = 0 ;
ATM_RESET_ACTIVE = 0 ;
FETH1_ENABLED = 0 ;
FETH1_RESET_ACTIVE = 0 ;
RS232_1_ENABLE = 0 ;
RS232_2_ENABLE = 0 ;

"*********************************************
"******* Power On Defaults Assignments
"*********************************************
ATM_ENABLE_PON_DEFAULT = !ATM_ENABLED ;
ATM_RESET_PON_DEFAULT = !ATM_RESET_ACTIVE ;
FETH1_ENABLE_PON_DEFAULT = !FETH1_ENABLED ;
FETH1_RESET_PON_DEFAULT = !FETH1_RESET_ACTIVE ;
RS232_1_ENABLE_PON_DEFAULT = !RS232_1_ENABLE ;
RS232_2_ENABLE_PON_DEFAULT = !RS232_2_ENABLE ;

"*****************************************
"******* Data Bits Assignments
"*****************************************
CONF_WORD_DATA_BIT =      [D0] ;
BOOT_DEVICE_DATA_BIT =    [D1] ;
ATM_ENABLE_DATA_BIT =     [D2] ;
ATM_RESET_DATA_BIT =      [D3] ;
FETH1_ENABLE_DATA_BIT =   [D4] ;
FETH1_RESET_DATA_BIT =    [D5] ;
RS232_1_ENABLE_DATA_BIT = [D6] ;
RS232_2_ENABLE_DATA_BIT = [D7] ;

"*****************************************************************************
"*****************************************************************************
"* BCSR 3 definitions.
"*****************************************************************************
"*****************************************************************************
USB_ENABLED = 0 ;
USB_SPEED_HIGH = 0 ;
USB_VCCO_ON = 1 ;
FETH2_ENABLED = 0 ;
FETH2_RESET_ACTIVE = 0 ;

ATM_SINGLE_PHY_ENABLED = 0 ;

"*********************************************
```

```
"******* Power On Defaults Assignments
"*********************************************
USB_ENABLE_PON_DEFAULT = !USB_ENABLED ;
USB_SPEED_PON_DEFAULT = USB_SPEED_HIGH ;
USB_VCCO_PON_DEFAULT = !USB_VCCO_ON ;
FETH2_ENABLE_PON_DEFAULT = !FETH2_ENABLED ;
FETH2_RESET_PON_DEFAULT = !FETH2_RESET_ACTIVE ;

ATM_SINGLE_PHY_ENABLE_PON_DEFAULT = ATM_SINGLE_PHY_ENABLED ;

"*******************************************
"******* Data Bits Assignments
"*******************************************
USB_ENABLE_DATA_BIT =          [D0] ;
USB_SPEED_DATA_BIT =           [D1] ;
USB_VCCO_DATA_BIT =            [D2] ;
FETH2_ENABLE_DATA_BIT =        [D3] ;
FETH2_RESET_DATA_BIT =         [D4] ;

ATM_SINGLE_PHY_ENABLE_DATA_BIT = [D6] ;
LOCAL_BUS_DATA_BIT =           [D7] ;

"*****************************************************************************
"* PCI Interrupt Register Access definitions
"*****************************************************************************
IntReg_ADD     = 0 ;
IntMaskReg_ADD = 1 ;

"VGR_WRITE_IntReg     = (!IntContCs_B & !DVal_B & R_B_W & !A27 & !A28 & !A29) ;
VGR_WRITE_IntMaskReg = (!IntContCs_B & !DVal_B & R_B_W & !A27 & !A28 & A29) ;

VGR_READ_IntReg     = (!IntContCs_B & !R_B_W & !A27 & !A28 & !A29) ;
VGR_READ_IntMaskReg = (!IntContCs_B & !R_B_W & !A27 & !A28 & A29) ;

"*****************************************************************************
"* Interrupt Request Definitions.
"*****************************************************************************

"IrqOe = (Slot0IntA #
" Slot0IntB #
" Slot0IntC #
" Slot0IntD #
" Slot1IntA #
" Slot1IntB #
" Slot1IntC #
" Slot1IntD #
" Slot2IntA #
" Slot2IntB #
" Slot2IntC #
" Slot2IntD) ;

"*****************************************************************************
"*****************************************************************************
"* PCI Interrupt Register definitions.
"*****************************************************************************
"*****************************************************************************
Slot0IntA_Active = 1 ; " PCI Slot 0 Interrupt A asserted
Slot0IntB_Active = 1 ; " PCI Slot 0 Interrupt B asserted
Slot0IntC_Active = 1 ; " PCI Slot 0 Interrupt C asserted
```

```
Slot0IntD_Active = 1 ; " PCI Slot 0 Interrupt D asserted
Slot1IntA_Active = 1 ; " PCI Slot 1 Interrupt A asserted
Slot1IntB_Active = 1 ; " PCI Slot 1 Interrupt B asserted
Slot1IntC_Active = 1 ; " PCI Slot 1 Interrupt C asserted
Slot1IntD_Active = 1 ; " PCI Slot 1 Interrupt D asserted
Slot2IntA_Active = 1 ; " PCI Slot 2 Interrupt A asserted
Slot2IntB_Active = 1 ; " PCI Slot 2 Interrupt B asserted
Slot2IntC_Active = 1 ; " PCI Slot 2 Interrupt C asserted
Slot2IntD_Active = 1 ; " PCI Slot 2 Interrupt D asserted

"***********************************************
"******* Power On Defaults Assignments
"***********************************************
Slot0IntA_PON_DEFAULT = !Slot0IntA_Active ;
Slot0IntB_PON_DEFAULT = !Slot0IntB_Active ;
Slot0IntC_PON_DEFAULT = !Slot0IntC_Active ;
Slot0IntD_PON_DEFAULT = !Slot0IntD_Active ;
Slot1IntA_PON_DEFAULT = !Slot1IntA_Active ;
Slot1IntB_PON_DEFAULT = !Slot1IntB_Active ;
Slot1IntC_PON_DEFAULT = !Slot1IntC_Active ;
Slot1IntD_PON_DEFAULT = !Slot1IntD_Active ;
Slot2IntA_PON_DEFAULT = !Slot2IntA_Active ;
Slot2IntB_PON_DEFAULT = !Slot2IntB_Active ;
Slot2IntC_PON_DEFAULT = !Slot2IntC_Active ;
Slot2IntD_PON_DEFAULT = !Slot2IntD_Active ;

"*****************************************
"******* Data Bits Assignments
"*****************************************
Slot0IntA_DATA_BIT = [D0] ;
Slot0IntB_DATA_BIT = [D1] ;
Slot0IntC_DATA_BIT = [D2] ;
Slot0IntD_DATA_BIT = [D3] ;
Slot1IntA_DATA_BIT = [D4] ;
Slot1IntB_DATA_BIT = [D5] ;
Slot1IntC_DATA_BIT = [D6] ;
Slot1IntD_DATA_BIT = [D7] ;
Slot2IntA_DATA_BIT = [D8] ;
Slot2IntB_DATA_BIT = [D9] ;
Slot2IntC_DATA_BIT = [D10] ;
Slot2IntD_DATA_BIT = [D11] ;

"*************************************************************************
"*************************************************************************
"* PCI Interrupt Mask Register definitions.
"*************************************************************************
"*************************************************************************
Slot0IntAMask_Active = 1 ; " PCI Slot 0 Interrupt A Masked
Slot0IntBMask_Active = 1 ; " PCI Slot 0 Interrupt B Masked
Slot0IntCMask_Active = 1 ; " PCI Slot 0 Interrupt C Masked
Slot0IntDMask_Active = 1 ; " PCI Slot 0 Interrupt D Masked
Slot1IntAMask_Active = 1 ; " PCI Slot 1 Interrupt A Masked
Slot1IntBMask_Active = 1 ; " PCI Slot 1 Interrupt B Masked
Slot1IntCMask_Active = 1 ; " PCI Slot 1 Interrupt C Masked
Slot1IntDMask_Active = 1 ; " PCI Slot 1 Interrupt D Masked
Slot2IntAMask_Active = 1 ; " PCI Slot 2 Interrupt A Masked
Slot2IntBMask_Active = 1 ; " PCI Slot 2 Interrupt B Masked
Slot2IntCMask_Active = 1 ; " PCI Slot 2 Interrupt C Masked
Slot2IntDMask_Active = 1 ; " PCI Slot 2 Interrupt D Masked
```

```
"*********************************************
"******* Power On Defaults Assignments
"*********************************************
Slot0IntAMask_PON_DEFAULT = Slot0IntAMask_Active ;
Slot0IntBMask_PON_DEFAULT = Slot0IntBMask_Active ;
Slot0IntCMask_PON_DEFAULT = Slot0IntCMask_Active ;
Slot0IntDMask_PON_DEFAULT = Slot0IntDMask_Active ;
Slot1IntAMask_PON_DEFAULT = Slot1IntAMask_Active ;
Slot1IntBMask_PON_DEFAULT = Slot1IntBMask_Active ;
Slot1IntCMask_PON_DEFAULT = Slot1IntCMask_Active ;
Slot1IntDMask_PON_DEFAULT = Slot1IntDMask_Active ;
Slot2IntAMask_PON_DEFAULT = Slot2IntAMask_Active ;
Slot2IntBMask_PON_DEFAULT = Slot2IntBMask_Active ;
Slot2IntCMask_PON_DEFAULT = Slot2IntCMask_Active ;
Slot2IntDMask_PON_DEFAULT = Slot2IntDMask_Active ;

"*******************************************
"******* Data Bits Assignments
"*******************************************
Slot0IntAMask_DATA_BIT = [D0];
Slot0IntBMask_DATA_BIT = [D1];
Slot0IntCMask_DATA_BIT = [D2];
Slot0IntDMask_DATA_BIT = [D3];
Slot1IntAMask_DATA_BIT = [D4];
Slot1IntBMask_DATA_BIT = [D5];
Slot1IntCMask_DATA_BIT = [D6];
Slot1IntDMask_DATA_BIT = [D7];
Slot2IntAMask_DATA_BIT = [D8];
Slot2IntBMask_DATA_BIT = [D9];
Slot2IntCMask_DATA_BIT = [D10];
Slot2IntDMask_DATA_BIT = [D11];

"*****************************************************************************
"* Flash Declarations.
"*****************************************************************************
FLASH_ENABLE_ACTIVE = 0 ;

" the presence detect encoding for the below is fictional
" needs to be updated with real data.

CP29020    = (F_PD == 8) ;   " 1 X 2 MByte bank
SM73228XU1 = (F_PD == 2) ;   " 1 X 8 MByte bank
SM73248XU2 = (F_PD == 1) ;   " 2 X 8 MByte banks
SM73288XU4 = (F_PD == 0) ;   " 4 X 8 MByte banks

FLASH_BANK1 = ( CP29020 #
              SM73228XU1 #
       (SM73248XU2 & !A8) #
             (SM73288XU4 & !A7 & !A8) ) ;

FLASH_BANK2 = ( (SM73248XU2 & A8) #
              (SM73288XU4 & !A7 & A8) ) ;

FLASH_BANK3 = ( A7 & !A8 & SM73288XU4 ) ;

FLASH_BANK4 = ( A7 & A8 & SM73288XU4 ) ;

"*****************************************************************************
```

```
"* ATM UNI Declarations.
"*****************************************************************************

"*****************************************************************************
"* Reset Declarations.
"*****************************************************************************

HARD_RESET_ACTIVE = 0 ;
SOFT_RESET_ACTIVE = 0 ;

HARD_RESET_ASSERTED = (SyncHardReset_B.fb == HARD_RESET_ACTIVE) ;

"*****************************************************************************
"* data buffers enable.
"*****************************************************************************
BUFFER_DISABLED = 1 ;
BUFFER_ENABLED = !BUFFER_DISABLED ;

BUFFER_HOLD_OFF = (HoldOffCnt.fb != 0) ; " the delay is required for read as well
                                " since a fast device (eg bcsr) may
                                " content with the flash/eeprom

END_OF_FLASH_EEPROM_READ = !DVal_B & (!Cs0_B # !Cs4_B) & !R_B_W & DSyncHardReset_B.fb ;
 " end of flash/eeprom read cycle.
                                " not during hard reset config

END_OF_PCI_INT_CONT_READ =  !DVal_B & !IntContCs_B & !R_B_W ;
 " end of PCI Interrupt Controller read cycle.

END_OF_ATM_READ = !DVal_B & !AtmUniCsIn_B & !R_B_W ; " end of atm uni m/p i/f read cycle

END_OF_OTHER_CYCLE = (!DVal_B & Cs0_B & Cs4_B & AtmUniCsIn_B & IntContCs_B #
                !DVal_B & !AtmUniCsIn_B & R_B_W #
                !DVal_B & !ToolCs1_B & R_B_W #
                !DVal_B & !ToolCs2_B & R_B_W #
     !DVal_B & (!Cs0_B # !Cs4_B) & R_B_W #
     !DVal_B & !IntContCs_B & R_B_W) ;
" another access or atm uni write or tool 1 write or tool 2 write or
" flash/eeprom write PCI int cont write
"*****************************************************************************
"* Hard Reset Configuration Logic
"*****************************************************************************
HRESET_CFG_IN_BCSR = (bcsrConfEn == 1); " HRESET Conf Word in BCSR
HRESET_BOOT_IN_FLASH = ((bcsrConfEn == 0) & (boot_device_B == 0));
" HRESET Conf Word and Boot Code in FLASH
BOOT_IN_FLASH = ((bcsrConfEn == 1) & (boot_device_B == 0));
" HRESET Conf Word in BCSR and Boot Code in FLASH
HRESET_BOOT_IN_EEPROM = ((bcsrConfEn == 0) & (boot_device_B == 1));
" HRESET Conf Word and Boot Code in EEPROM
BOOT_IN_EEPROM = ((bcsrConfEn == 1) & (boot_device_B == 1));
" HRESET Conf Word in BCSR and Boot Code in EEPROM
HARD_RESET_ASSERTION = ( (HardReset_B == 0) & (SyncHardReset_B.fb == 0) &
                (DSyncHardReset_B.fb == 1) );

CS0_ASSERTED = (Cs0_B == 0);
CS4_ASSERTED = (Cs4_B == 0);

FIRST_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 0) &
HRESET_CFG_IN_BCSR & !R_B_W);
```

```
SCND_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 1) &
HRESET_CFG_IN_BCSR & !R_B_W);
THIRD_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 2) &
HRESET_CFG_IN_BCSR & !R_B_W);
FORTH_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 3) &
HRESET_CFG_IN_BCSR & !R_B_W);

"*****************************************************************************
"* Equations, state diagrams.                                                *
"*****************************************************************************
"*****************************************************************************
"*****************************************************************************

equations

ClockedContReg.clk = SYSCLK ;
ClockedContReg.ar = 0 ;
ClockedContReg.ap = 0;
DrivenContReg.oe = ^hffff ;

"*****************************************************************************
"*****************************************************************************
"* BCSR 0
"*****************************************************************************
"*****************************************************************************
equations
"*****************************************************************************
state_diagram SignaLamp0_B
state SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP0_DATA_BIT.pin == !SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP0_PON_DEFAULT != SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP0_PON_DEFAULT == !SIGNAL_LAMP_ON)) ) then
!SIGNAL_LAMP_ON
else
SIGNAL_LAMP_ON ;
state !SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP0_DATA_BIT.pin == SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP0_PON_DEFAULT != !SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP0_PON_DEFAULT == SIGNAL_LAMP_ON)) ) then
SIGNAL_LAMP_ON
else
!SIGNAL_LAMP_ON ;
"*****************************************************************************
state_diagram SignaLamp1_B
state SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP1_DATA_BIT.pin == !SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP1_PON_DEFAULT != SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP1_PON_DEFAULT == !SIGNAL_LAMP_ON)) ) then
!SIGNAL_LAMP_ON
else
SIGNAL_LAMP_ON ;
state !SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP1_DATA_BIT.pin == SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP1_PON_DEFAULT != !SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP1_PON_DEFAULT == SIGNAL_LAMP_ON)) ) then
```

```
SIGNAL_LAMP_ON
else
!SIGNAL_LAMP_ON ;
"*****************************************************************************
"*****************************************************************************
"* BCSR1 State Machines
"*****************************************************************************
"*****************************************************************************
state_diagram AtmEn_B
state ATM_ENABLED:
if (VGR_WRITE_BCSR_1 &
(ATM_ENABLE_DATA_BIT.pin == !ATM_ENABLED) &
 (!PON_RESET # (ATM_ENABLE_PON_DEFAULT != ATM_ENABLED)) #
 (PON_RESET & (ATM_ENABLE_PON_DEFAULT == !ATM_ENABLED)) ) then
!ATM_ENABLED
else
ATM_ENABLED ;
state !ATM_ENABLED:
if (VGR_WRITE_BCSR_1 & !FETH1_ENABLED &
(ATM_ENABLE_DATA_BIT.pin == ATM_ENABLED) &
 (!PON_RESET # (ATM_ENABLE_PON_DEFAULT != !ATM_ENABLED)) #
 (PON_RESET & (ATM_ENABLE_PON_DEFAULT == ATM_ENABLED)) ) then
ATM_ENABLED
else
!ATM_ENABLED ;
"*****************************************************************************
state_diagram AtmRst_B
state ATM_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(ATM_RESET_DATA_BIT.pin == !ATM_RESET_ACTIVE) &
 (!PON_RESET # (ATM_RESET_PON_DEFAULT != ATM_RESET_ACTIVE)) #
 (PON_RESET & (ATM_RESET_PON_DEFAULT == !ATM_RESET_ACTIVE)) ) then
!ATM_RESET_ACTIVE
else
ATM_RESET_ACTIVE ;
state !ATM_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(ATM_RESET_DATA_BIT.pin == ATM_RESET_ACTIVE) &
 (!PON_RESET # (ATM_RESET_PON_DEFAULT != !ATM_RESET_ACTIVE)) #
 (PON_RESET & (ATM_RESET_PON_DEFAULT == ATM_RESET_ACTIVE)) ) then
ATM_RESET_ACTIVE
else
!ATM_RESET_ACTIVE ;
"*****************************************************************************
state_diagram FEthEn1_B
state FETH1_ENABLED:
if (VGR_WRITE_BCSR_1 &
(FETH1_ENABLE_DATA_BIT.pin == !FETH1_ENABLED) &
 (!PON_RESET # (FETH1_ENABLE_PON_DEFAULT != FETH1_ENABLED)) #
 (PON_RESET & (FETH1_ENABLE_PON_DEFAULT == !FETH1_ENABLED)) ) then
!FETH1_ENABLED
else
FETH1_ENABLED ;
state !FETH1_ENABLED:
if (VGR_WRITE_BCSR_1 & !ATM_ENABLED &
(FETH1_ENABLE_DATA_BIT.pin == FETH1_ENABLED) &
 (!PON_RESET # (FETH1_ENABLE_PON_DEFAULT != !FETH1_ENABLED)) #
 (PON_RESET & (FETH1_ENABLE_PON_DEFAULT == FETH1_ENABLED)) ) then
FETH1_ENABLED
```

```
else
!FETH1_ENABLED ;
"*******************************************************************************
state_diagram FEthRst1_B
state FETH1_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(FETH1_RESET_DATA_BIT.pin == !FETH1_RESET_ACTIVE) &
 (!PON_RESET # (FETH1_RESET_PON_DEFAULT != FETH1_RESET_ACTIVE)) #
 (PON_RESET & (FETH1_RESET_PON_DEFAULT == !FETH1_RESET_ACTIVE)) ) then
!FETH1_RESET_ACTIVE
else
FETH1_RESET_ACTIVE ;
state !FETH1_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(FETH1_RESET_DATA_BIT.pin == FETH1_RESET_ACTIVE) &
 (!PON_RESET # (FETH1_RESET_PON_DEFAULT != !FETH1_RESET_ACTIVE)) #
 (PON_RESET & (FETH1_RESET_PON_DEFAULT == FETH1_RESET_ACTIVE)) ) then
FETH1_RESET_ACTIVE
else
!FETH1_RESET_ACTIVE ;
"*******************************************************************************
state_diagram RS232En1_B
state RS232_1_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_1_ENABLE_DATA_BIT.pin == !RS232_1_ENABLE) &
 (!PON_RESET # (RS232_1_ENABLE_PON_DEFAULT != RS232_1_ENABLE)) #
 (PON_RESET & (RS232_1_ENABLE_PON_DEFAULT == !RS232_1_ENABLE)) ) then
!RS232_1_ENABLE
else
RS232_1_ENABLE ;
state !RS232_1_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_1_ENABLE_DATA_BIT.pin == RS232_1_ENABLE) &
 (!PON_RESET # (RS232_1_ENABLE_PON_DEFAULT != !RS232_1_ENABLE)) #
 (PON_RESET & (RS232_1_ENABLE_PON_DEFAULT == RS232_1_ENABLE)) ) then
RS232_1_ENABLE
else
!RS232_1_ENABLE ;
"*******************************************************************************
state_diagram RS232En2_B
state RS232_2_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_2_ENABLE_DATA_BIT.pin == !RS232_2_ENABLE) &
 (!PON_RESET # (RS232_2_ENABLE_PON_DEFAULT != RS232_2_ENABLE)) #
 (PON_RESET & (RS232_2_ENABLE_PON_DEFAULT == !RS232_2_ENABLE)) ) then
!RS232_2_ENABLE
else
RS232_2_ENABLE ;
state !RS232_2_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_2_ENABLE_DATA_BIT.pin == RS232_2_ENABLE) &
 (!PON_RESET # (RS232_2_ENABLE_PON_DEFAULT != !RS232_2_ENABLE)) #
 (PON_RESET & (RS232_2_ENABLE_PON_DEFAULT == RS232_2_ENABLE)) ) then
RS232_2_ENABLE
else
!RS232_2_ENABLE ;
"*******************************************************************************
"*******************************************************************************
"* BCSR3 State Machines
```

```
"*****************************************************************************
"*****************************************************************************
state_diagram USBEn_B
state USB_ENABLED:
if (VGR_WRITE_BCSR_3 &
(USB_ENABLE_DATA_BIT.pin == !USB_ENABLED) &
 (!PON_RESET # (USB_ENABLE_PON_DEFAULT != USB_ENABLED)) #
 (PON_RESET & (USB_ENABLE_PON_DEFAULT == !USB_ENABLED)) ) then
!USB_ENABLED
else
USB_ENABLED ;
state !USB_ENABLED:
if (VGR_WRITE_BCSR_3 &
(USB_ENABLE_DATA_BIT.pin == USB_ENABLED) &
 (!PON_RESET # (USB_ENABLE_PON_DEFAULT != !USB_ENABLED)) #
 (PON_RESET & (USB_ENABLE_PON_DEFAULT == USB_ENABLED)) ) then
USB_ENABLED
else
!USB_ENABLED ;
"*****************************************************************************
state_diagram USBHiSpd_B
state USB_SPEED_HIGH:
if (VGR_WRITE_BCSR_3 &
(USB_SPEED_DATA_BIT.pin == !USB_SPEED_HIGH) &
 (!PON_RESET # (USB_SPEED_PON_DEFAULT != USB_SPEED_HIGH)) #
 (PON_RESET & (USB_SPEED_PON_DEFAULT == !USB_SPEED_HIGH)) ) then
!USB_SPEED_HIGH
else
USB_SPEED_HIGH ;
state !USB_SPEED_HIGH:
if (VGR_WRITE_BCSR_3 &
(USB_SPEED_DATA_BIT.pin == USB_SPEED_HIGH) &
 (!PON_RESET # (USB_SPEED_PON_DEFAULT != !USB_SPEED_HIGH)) #
 (PON_RESET & (USB_SPEED_PON_DEFAULT == USB_SPEED_HIGH)) ) then
USB_SPEED_HIGH
else
!USB_SPEED_HIGH ;
"*****************************************************************************
state_diagram USBVccO
state USB_VCCO_ON:
if (VGR_WRITE_BCSR_3 &
(USB_VCCO_DATA_BIT.pin == !USB_VCCO_ON) &
 (!PON_RESET # (USB_VCCO_PON_DEFAULT != USB_VCCO_ON)) #
 (PON_RESET & (USB_VCCO_PON_DEFAULT == !USB_VCCO_ON)) ) then
!USB_VCCO_ON
else
USB_VCCO_ON ;
state !USB_VCCO_ON:
if (VGR_WRITE_BCSR_3 &
(USB_VCCO_DATA_BIT.pin == USB_VCCO_ON) &
 (!PON_RESET # (USB_VCCO_PON_DEFAULT != !USB_VCCO_ON)) #
 (PON_RESET & (USB_VCCO_PON_DEFAULT == USB_VCCO_ON)) ) then
USB_VCCO_ON
else
!USB_VCCO_ON ;
"*****************************************************************************
state_diagram FEthEn2_B
state FETH2_ENABLED:
if (VGR_WRITE_BCSR_3 &
```

```
(FETH2_ENABLE_DATA_BIT.pin == !FETH2_ENABLED) &
 (!PON_RESET # (FETH2_ENABLE_PON_DEFAULT != FETH2_ENABLED)) #
 (PON_RESET & (FETH2_ENABLE_PON_DEFAULT == !FETH2_ENABLED)) ) then
!FETH2_ENABLED
else
FETH2_ENABLED ;
state !FETH2_ENABLED:
if (VGR_WRITE_BCSR_3 &
(FETH2_ENABLE_DATA_BIT.pin == FETH2_ENABLED) &
 (!PON_RESET # (FETH2_ENABLE_PON_DEFAULT != !FETH2_ENABLED)) #
 (PON_RESET & (FETH2_ENABLE_PON_DEFAULT == FETH2_ENABLED)) ) then
FETH2_ENABLED
else
!FETH2_ENABLED ;
"*****************************************************************************
state_diagram FEthRst2_B
state FETH2_RESET_ACTIVE:
if (VGR_WRITE_BCSR_3 &
(FETH2_RESET_DATA_BIT.pin == !FETH2_RESET_ACTIVE) &
 (!PON_RESET # (FETH2_RESET_PON_DEFAULT != FETH2_RESET_ACTIVE)) #
 (PON_RESET & (FETH2_RESET_PON_DEFAULT == !FETH2_RESET_ACTIVE)) ) then
!FETH2_RESET_ACTIVE
else
FETH2_RESET_ACTIVE ;
state !FETH2_RESET_ACTIVE:
if (VGR_WRITE_BCSR_3 &
(FETH2_RESET_DATA_BIT.pin == FETH2_RESET_ACTIVE) &
 (!PON_RESET # (FETH2_RESET_PON_DEFAULT != !FETH2_RESET_ACTIVE)) #
 (PON_RESET & (FETH2_RESET_PON_DEFAULT == FETH2_RESET_ACTIVE)) ) then
FETH2_RESET_ACTIVE
else
!FETH2_RESET_ACTIVE ;
"*****************************************************************************
state_diagram AtmSinglePHY_B
state ATM_SINGLE_PHY_ENABLED:
if (VGR_WRITE_BCSR_3 &
(ATM_SINGLE_PHY_ENABLE_DATA_BIT.pin == !ATM_SINGLE_PHY_ENABLED) &
 (!PON_RESET
   # (ATM_SINGLE_PHY_ENABLE_PON_DEFAULT != ATM_SINGLE_PHY_ENABLED)) #
 (PON_RESET
   & (ATM_SINGLE_PHY_ENABLE_PON_DEFAULT == !ATM_SINGLE_PHY_ENABLED)) ) then
!ATM_SINGLE_PHY_ENABLED
else
ATM_SINGLE_PHY_ENABLED ;
state !ATM_SINGLE_PHY_ENABLED:
if (VGR_WRITE_BCSR_3 &
(ATM_SINGLE_PHY_ENABLE_DATA_BIT.pin == ATM_SINGLE_PHY_ENABLED) &
 (!PON_RESET
   # (ATM_SINGLE_PHY_ENABLE_PON_DEFAULT != !ATM_SINGLE_PHY_ENABLED)) #
 (PON_RESET
   & (ATM_SINGLE_PHY_ENABLE_PON_DEFAULT == ATM_SINGLE_PHY_ENABLED)) ) then
ATM_SINGLE_PHY_ENABLED
else
!ATM_SINGLE_PHY_ENABLED ;

"*****************************************************************************
equations

AtmMultiPHY_B = !AtmSinglePHY_B ;
```

```
USBDis_B = !USBEn_B ;
USBLowSpd_B = !USBHiSpd_B ;

USBVccOut = !USBVccO ;

FEthDis1_B = !FEthEn1_B ;
FEthDis2_B = !FEthEn2_B ;

RS232Dis1_B = !RS232En1_B ;
RS232Dis2_B = !RS232En2_B ;

FCC1On_B = FEthEn1_B & AtmEn_B ;
FCC1Off_B = !FCC1On_B ;

MIIOn_B = FEthEn1_B & FEthEn2_B ;
MIIOff_B = !MIIOn_B ;


MII1PLLOff_B = !FEth1PLLOn_B ;
MII2PLLOff_B = !FEth2PLLOn_B ;

"*******************************************************************************

"*******************************************************************************
"*******************************************************************************
"* PCI Interrupt Register
"*******************************************************************************
"*******************************************************************************
equations

IntReg.clk = SYSCLK ;
IntReg.ar = 0 ;
IntReg.ap = 0 ;

"*******************************************************************************
state_diagram Slot0IntA
state Slot0IntA_Active:
if ( ((HardReset_B == 0) & (Slot0IntA_PON_DEFAULT == !Slot0IntA_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTA_B & !Slot0IntAMask.fb) #
    Slot0IntAMask.fb)) )
then
!Slot0IntA_Active
else
Slot0IntA_Active ;
state !Slot0IntA_Active:
if ( ((HardReset_B == 0) & (Slot0IntA_PON_DEFAULT == Slot0IntA_Active)) #
    (!(HardReset_B == 0) & !PCI_INTA_B & !Slot0IntAMask.fb) )
then
Slot0IntA_Active
else
!Slot0IntA_Active ;
"*******************************************************************************
state_diagram Slot0IntB
state Slot0IntB_Active:
if ( ((HardReset_B == 0) & (Slot0IntB_PON_DEFAULT == !Slot0IntB_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTB_B & !Slot0IntBMask.fb) #
    Slot0IntBMask.fb)) )
then
```

```
!Slot0IntB_Active
else
Slot0IntB_Active ;
state !Slot0IntB_Active:
if ( ((HardReset_B == 0) & (Slot0IntB_PON_DEFAULT == Slot0IntB_Active)) #
    (!(HardReset_B == 0) & !PCI_INTB_B & !Slot0IntBMask.fb) )
then
Slot0IntB_Active
else
!Slot0IntB_Active ;
"*****************************************************************************
state_diagram Slot0IntC
state Slot0IntC_Active:
if ( ((HardReset_B == 0) & (Slot0IntC_PON_DEFAULT == !Slot0IntC_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTC_B & !Slot0IntCMask.fb) #
    Slot0IntCMask.fb)) )
then
!Slot0IntC_Active
else
Slot0IntC_Active ;
state !Slot0IntC_Active:
if ( ((HardReset_B == 0) & (Slot0IntC_PON_DEFAULT == Slot0IntC_Active)) #
    (!(HardReset_B == 0) & !PCI_INTC_B & !Slot0IntCMask.fb) )
then
Slot0IntC_Active
else
!Slot0IntC_Active ;
"*****************************************************************************
state_diagram Slot0IntD
state Slot0IntD_Active:
if ( ((HardReset_B == 0) & (Slot0IntD_PON_DEFAULT == !Slot0IntD_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTD_B & !Slot0IntDMask.fb) #
    Slot0IntDMask.fb)) )
then
!Slot0IntD_Active
else
Slot0IntD_Active ;
state !Slot0IntD_Active:
if ( ((HardReset_B == 0) & (Slot0IntD_PON_DEFAULT == Slot0IntD_Active)) #
    (!(HardReset_B == 0) & !PCI_INTD_B & !Slot0IntDMask.fb) )
then
Slot0IntD_Active
else
!Slot0IntD_Active ;
"*****************************************************************************
state_diagram Slot1IntA
state Slot1IntA_Active:
if ( ((HardReset_B == 0) & (Slot1IntA_PON_DEFAULT == !Slot1IntA_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTD_B & !Slot1IntAMask.fb) #
    Slot1IntAMask.fb)) )
then
!Slot1IntA_Active
else
Slot1IntA_Active ;
state !Slot1IntA_Active:
if ( ((HardReset_B == 0) & (Slot1IntA_PON_DEFAULT == Slot1IntA_Active)) #
    (!(HardReset_B == 0) & !PCI_INTD_B & !Slot1IntAMask.fb) )
then
Slot1IntA_Active
```

```
else
!Slot1IntA_Active ;
"*******************************************************************************
state_diagram Slot1IntB
state Slot1IntB_Active:
if ( ((HardReset_B == 0) & (Slot1IntB_PON_DEFAULT == !Slot1IntB_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTA_B & !Slot1IntBMask.fb) #
    Slot1IntBMask.fb)) )
then
!Slot1IntB_Active
else
Slot1IntB_Active ;
state !Slot1IntB_Active:
if ( ((HardReset_B == 0) & (Slot1IntB_PON_DEFAULT == Slot1IntB_Active)) #
    (!(HardReset_B == 0) & !PCI_INTA_B & !Slot1IntBMask.fb) )
then
Slot1IntB_Active
else
!Slot1IntB_Active ;
"*******************************************************************************
state_diagram Slot1IntC
state Slot1IntC_Active:
if ( ((HardReset_B == 0) & (Slot1IntC_PON_DEFAULT == !Slot1IntC_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTB_B & !Slot1IntCMask.fb) #
    Slot1IntCMask.fb)) )
then
!Slot1IntC_Active
else
Slot1IntC_Active ;
state !Slot1IntC_Active:
if ( ((HardReset_B == 0) & (Slot1IntC_PON_DEFAULT == Slot1IntC_Active)) #
    (!(HardReset_B == 0) & !PCI_INTB_B & !Slot1IntCMask.fb) )
then
Slot1IntC_Active
else
!Slot1IntC_Active ;
"*******************************************************************************
state_diagram Slot1IntD
state Slot1IntD_Active:
if ( ((HardReset_B == 0) & (Slot1IntD_PON_DEFAULT == !Slot1IntD_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTC_B & !Slot1IntDMask.fb) #
    Slot1IntDMask.fb)) )
then
!Slot1IntD_Active
else
Slot1IntD_Active ;
state !Slot1IntD_Active:
if ( ((HardReset_B == 0) & (Slot1IntD_PON_DEFAULT == Slot1IntD_Active)) #
    (!(HardReset_B == 0) & !PCI_INTC_B & !Slot1IntDMask.fb) )
then
Slot1IntD_Active
else
!Slot1IntD_Active ;
"*******************************************************************************
state_diagram Slot2IntA
state Slot2IntA_Active:
if ( ((HardReset_B == 0) & (Slot2IntA_PON_DEFAULT == !Slot2IntA_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTC_B & !Slot2IntAMask.fb) #
    Slot2IntAMask.fb)) )
```

```
then
!Slot2IntA_Active
else
Slot2IntA_Active ;
state !Slot2IntA_Active:
if ( ((HardReset_B == 0) & (Slot2IntA_PON_DEFAULT == Slot2IntA_Active)) #
    (!(HardReset_B == 0) & !PCI_INTC_B & !Slot2IntAMask.fb) )
then
Slot2IntA_Active
else
!Slot2IntA_Active ;
"*****************************************************************************
state_diagram Slot2IntB
state Slot2IntB_Active:
if ( ((HardReset_B == 0) & (Slot2IntB_PON_DEFAULT == !Slot2IntB_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTD_B & !Slot2IntBMask.fb) #
    Slot2IntBMask.fb)) )
then
!Slot2IntB_Active
else
Slot2IntB_Active ;
state !Slot2IntB_Active:
if ( ((HardReset_B == 0) & (Slot2IntB_PON_DEFAULT == Slot2IntB_Active)) #
    (!(HardReset_B == 0) & !PCI_INTD_B & !Slot2IntBMask.fb) )
then
Slot2IntB_Active
else
!Slot2IntB_Active ;
"*****************************************************************************
state_diagram Slot2IntC
state Slot2IntC_Active:
if ( ((HardReset_B == 0) & (Slot2IntC_PON_DEFAULT == !Slot2IntC_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTA_B & !Slot2IntCMask.fb) #
    Slot2IntCMask.fb)) )
then
!Slot2IntC_Active
else
Slot2IntC_Active ;
state !Slot2IntC_Active:
if ( ((HardReset_B == 0) & (Slot2IntC_PON_DEFAULT == Slot2IntC_Active)) #
    (!(HardReset_B == 0) & !PCI_INTA_B & !Slot2IntCMask.fb) )
then
Slot2IntC_Active
else
!Slot2IntC_Active ;
"*****************************************************************************
state_diagram Slot2IntD
state Slot2IntD_Active:
if ( ((HardReset_B == 0) & (Slot2IntD_PON_DEFAULT == !Slot2IntD_Active)) #
    (!(HardReset_B == 0) & ((PCI_INTB_B & !Slot2IntDMask.fb) #
    Slot2IntDMask.fb)) )
then
!Slot2IntD_Active
else
Slot2IntD_Active ;
state !Slot2IntD_Active:
if ( ((HardReset_B == 0) & (Slot2IntD_PON_DEFAULT == Slot2IntD_Active)) #
    (!(HardReset_B == 0) & !PCI_INTB_B & !Slot2IntDMask.fb) )
then
```

```
Slot2IntD_Active
else
!Slot2IntD_Active ;
"*****************************************************************************
"*****************************************************************************
"* PCI Interrupt Mask Register
"*****************************************************************************
"*****************************************************************************
equations

IntMaskReg.clk = SYSCLK ;
IntMaskReg.ar = 0 ;
IntMaskReg.ap = 0 ;

"*****************************************************************************
state_diagram Slot0IntAMask
state Slot0IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntAMask_DATA_BIT.pin == !Slot0IntAMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntAMask_PON_DEFAULT == !Slot0IntAMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntAMask_PON_DEFAULT == !Slot0IntAMask_Active)) )
then
!Slot0IntAMask_Active
else
Slot0IntAMask_Active ;
state !Slot0IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntAMask_DATA_BIT.pin == Slot0IntAMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntAMask_PON_DEFAULT == Slot0IntAMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntAMask_PON_DEFAULT == Slot0IntAMask_Active)) )
then
Slot0IntAMask_Active
else
!Slot0IntAMask_Active ;
"*****************************************************************************
state_diagram Slot0IntBMask
state Slot0IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntBMask_DATA_BIT.pin == !Slot0IntBMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntBMask_PON_DEFAULT == !Slot0IntBMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntBMask_PON_DEFAULT == !Slot0IntBMask_Active)) )
then
!Slot0IntBMask_Active
else
Slot0IntBMask_Active ;
state !Slot0IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntBMask_DATA_BIT.pin == Slot0IntBMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntBMask_PON_DEFAULT == Slot0IntBMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntBMask_PON_DEFAULT == Slot0IntBMask_Active)) )
then
Slot0IntBMask_Active
else
!Slot0IntBMask_Active ;
"*****************************************************************************
state_diagram Slot0IntCMask
state Slot0IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntCMask_DATA_BIT.pin == !Slot0IntCMask_Active) &
```

```
     (!(HardReset_B == 0) # (Slot0IntCMask_PON_DEFAULT == !Slot0IntCMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntCMask_PON_DEFAULT == !Slot0IntCMask_Active)) )
then
!Slot0IntCMask_Active
else
Slot0IntCMask_Active ;
state !Slot0IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntCMask_DATA_BIT.pin == Slot0IntCMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntCMask_PON_DEFAULT == Slot0IntCMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntCMask_PON_DEFAULT == Slot0IntCMask_Active)) )
then
Slot0IntCMask_Active
else
!Slot0IntCMask_Active ;
"*****************************************************************************
state_diagram Slot0IntDMask
state Slot0IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntDMask_DATA_BIT.pin == !Slot0IntDMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntDMask_PON_DEFAULT == !Slot0IntDMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntDMask_PON_DEFAULT == !Slot0IntDMask_Active)) )
then
!Slot0IntDMask_Active
else
Slot0IntDMask_Active ;
state !Slot0IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntDMask_DATA_BIT.pin == Slot0IntDMask_Active) &
     (!(HardReset_B == 0) # (Slot0IntDMask_PON_DEFAULT == Slot0IntDMask_Active)) #
   ((HardReset_B == 0) & (Slot0IntDMask_PON_DEFAULT == Slot0IntDMask_Active)) )
then
Slot0IntDMask_Active
else
!Slot0IntDMask_Active ;
"*****************************************************************************
state_diagram Slot1IntAMask
state Slot1IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot1IntAMask_DATA_BIT.pin == !Slot1IntAMask_Active) &
     (!(HardReset_B == 0) # (Slot1IntAMask_PON_DEFAULT == !Slot1IntAMask_Active)) #
   ((HardReset_B == 0) & (Slot1IntAMask_PON_DEFAULT == !Slot1IntAMask_Active)) )
then
!Slot1IntAMask_Active
else
Slot1IntAMask_Active ;
state !Slot1IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot1IntAMask_DATA_BIT.pin == Slot1IntAMask_Active) &
     (!(HardReset_B == 0) # (Slot1IntAMask_PON_DEFAULT == Slot1IntAMask_Active)) #
   ((HardReset_B == 0) & (Slot1IntAMask_PON_DEFAULT == Slot1IntAMask_Active)) )
then
Slot1IntAMask_Active
else
!Slot1IntAMask_Active ;
"*****************************************************************************
state_diagram Slot1IntBMask
state Slot1IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
```

```
      (Slot1IntBMask_DATA_BIT.pin == !Slot1IntBMask_Active) &
      (!(HardReset_B == 0) # (Slot1IntBMask_PON_DEFAULT == !Slot1IntBMask_Active)) #
    ((HardReset_B == 0) & (Slot1IntBMask_PON_DEFAULT == !Slot1IntBMask_Active)) )
then
!Slot1IntBMask_Active
else
Slot1IntBMask_Active ;
state !Slot1IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
      (Slot1IntBMask_DATA_BIT.pin == Slot1IntBMask_Active) &
      (!(HardReset_B == 0) # (Slot1IntBMask_PON_DEFAULT == Slot1IntBMask_Active)) #
    ((HardReset_B == 0) & (Slot1IntBMask_PON_DEFAULT == Slot1IntBMask_Active)) )
then
Slot1IntBMask_Active
else
!Slot1IntBMask_Active ;
"*****************************************************************************
state_diagram Slot1IntCMask
state Slot1IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
      (Slot1IntCMask_DATA_BIT.pin == !Slot1IntCMask_Active) &
      (!(HardReset_B == 0) # (Slot1IntCMask_PON_DEFAULT == !Slot1IntCMask_Active)) #
    ((HardReset_B == 0) & (Slot1IntCMask_PON_DEFAULT == !Slot1IntCMask_Active)) )
then
!Slot1IntCMask_Active
else
Slot1IntCMask_Active ;
state !Slot1IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
      (Slot1IntCMask_DATA_BIT.pin == Slot1IntCMask_Active) &
      (!(HardReset_B == 0) # (Slot1IntCMask_PON_DEFAULT == Slot1IntCMask_Active)) #
    ((HardReset_B == 0) & (Slot1IntCMask_PON_DEFAULT == Slot1IntCMask_Active)) )
then
Slot1IntCMask_Active
else
!Slot1IntCMask_Active ;
"*****************************************************************************
state_diagram Slot1IntDMask
state Slot1IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
      (Slot1IntDMask_DATA_BIT.pin == !Slot1IntDMask_Active) &
      (!(HardReset_B == 0) # (Slot1IntDMask_PON_DEFAULT == !Slot1IntDMask_Active)) #
    ((HardReset_B == 0) & (Slot1IntDMask_PON_DEFAULT == !Slot1IntDMask_Active)) )
then
!Slot1IntDMask_Active
else
Slot1IntDMask_Active ;
state !Slot1IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
      (Slot1IntDMask_DATA_BIT.pin == Slot1IntDMask_Active) &
      (!(HardReset_B == 0) # (Slot1IntDMask_PON_DEFAULT == Slot1IntDMask_Active)) #
    ((HardReset_B == 0) & (Slot1IntDMask_PON_DEFAULT == Slot1IntDMask_Active)) )
then
Slot1IntDMask_Active
else
!Slot1IntDMask_Active ;
"*****************************************************************************
state_diagram Slot2IntAMask
state Slot2IntAMask_Active:
```

```
if (VGR_WRITE_IntMaskReg &
     (Slot2IntAMask_DATA_BIT.pin == !Slot2IntAMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntAMask_PON_DEFAULT == !Slot2IntAMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntAMask_PON_DEFAULT == !Slot2IntAMask_Active)) )
then
!Slot2IntAMask_Active
else
Slot2IntAMask_Active ;
state !Slot2IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntAMask_DATA_BIT.pin == Slot2IntAMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntAMask_PON_DEFAULT == Slot2IntAMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntAMask_PON_DEFAULT == Slot2IntAMask_Active)) )
then
Slot2IntAMask_Active
else
!Slot2IntAMask_Active ;
"*****************************************************************************
state_diagram Slot2IntBMask
state Slot2IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntBMask_DATA_BIT.pin == !Slot2IntBMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntBMask_PON_DEFAULT == !Slot2IntBMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntBMask_PON_DEFAULT == !Slot2IntBMask_Active)) )
then
!Slot2IntBMask_Active
else
Slot2IntBMask_Active ;
state !Slot2IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntBMask_DATA_BIT.pin == Slot2IntBMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntBMask_PON_DEFAULT == Slot2IntBMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntBMask_PON_DEFAULT == Slot2IntBMask_Active)) )
then
Slot2IntBMask_Active
else
!Slot2IntBMask_Active ;
"*****************************************************************************
state_diagram Slot2IntCMask
state Slot2IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntCMask_DATA_BIT.pin == !Slot2IntCMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntCMask_PON_DEFAULT == !Slot2IntCMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntCMask_PON_DEFAULT == !Slot2IntCMask_Active)) )
then
!Slot2IntCMask_Active
else
Slot2IntCMask_Active ;
state !Slot2IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntCMask_DATA_BIT.pin == Slot2IntCMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntCMask_PON_DEFAULT == Slot2IntCMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntCMask_PON_DEFAULT == Slot2IntCMask_Active)) )
then
Slot2IntCMask_Active
else
!Slot2IntCMask_Active ;
"*****************************************************************************
state_diagram Slot2IntDMask
```

```
state Slot2IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntDMask_DATA_BIT.pin == !Slot2IntDMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntDMask_PON_DEFAULT == !Slot2IntDMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntDMask_PON_DEFAULT == !Slot2IntDMask_Active)) )
then
!Slot2IntDMask_Active
else
Slot2IntDMask_Active ;
state !Slot2IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntDMask_DATA_BIT.pin == Slot2IntDMask_Active) &
     (!(HardReset_B == 0) # (Slot2IntDMask_PON_DEFAULT == Slot2IntDMask_Active)) #
   ((HardReset_B == 0) & (Slot2IntDMask_PON_DEFAULT == Slot2IntDMask_Active)) )
then
Slot2IntDMask_Active
else
!Slot2IntDMask_Active ;

"*******************************************************************************
"*******************************************************************************
" External Read Registers' Chip-Selects
"*******************************************************************************
"*******************************************************************************
equations

Bcsr2Cs_B.oe = H ;
Bcsr4Cs_B.oe = H ;

!Bcsr2Cs_B = VGR_READ_BCSR_2 ;
!Bcsr4Cs_B = VGR_READ_BCSR_4 ;

"*******************************************************************************
"*******************************************************************************
"* Read Registers.
"* All registers have read capabilty. (BCSR2 and BCSR4 are read externally)
"*******************************************************************************
"*******************************************************************************
equations
DataOe = VGR_READ_BCSR_0 #
        VGR_READ_BCSR_1 #
        VGR_READ_BCSR_3 #
        (HRESET_CFG_IN_BCSR & CS0_ASSERTED & !DSyncHardReset_B.fb) ;

Data.oe = DataOe ;

when (VGR_READ_BCSR_0) then
Data = ReadBcsr0 ;
else when (VGR_READ_BCSR_1) then
Data = ReadBcsr1 ;
else when (VGR_READ_BCSR_3) then
Data = ReadBcsr3 ;
       else when (FIRST_CFG_BYTE_READ) then
              Data = CfgByte0;
       else when (SCND_CFG_BYTE_READ) then
              Data = CfgByte1;
       else when (THIRD_CFG_BYTE_READ) then
              Data = CfgByte2;
       else when (FORTH_CFG_BYTE_READ) then
```

```
            Data = CfgByte3;

DataPCIOe = VGR_READ_IntReg #
   VGR_READ_IntMaskReg ;

DataPCI.oe = DataPCIOe ;

when (VGR_READ_IntReg) then
DataPCI = IntReg.fb ;
else when (VGR_READ_IntMaskReg) then
DataPCI = IntMaskReg.fb ;

"*********************   brd_ctl   ******************************************
"*******************************************************************************
"*******************************************************************************
"* Reset Logic
"*******************************************************************************
"*******************************************************************************
"*******************************************************************************
equations

Reset.oe = ResetEn ;

Reset = 0 ;" open drain

RstDeb1 = !( Rst1 & (!( RstDeb1.com & Rst0) ) ) ;    " Reset push-button debouncer

AbrDeb1 = !( Abr1 & (!( AbrDeb1.com & Abr0) ) ) ;    " Abort push-button debouncer


HardResetEn = RstDeb1.com & AbrDeb1.com  ;" both buttons are depressed;

SoftResetEn = RstDeb1.com & !AbrDeb1.com ;" only reset button depressed


TransRst.oe = 7 ;" transceivers' reset, always enabled.

!AtmRstOut_B = !AtmRst_B.fb # !HardReset_B ;

!FEthRstOut1_B = !FEthRst1_B.fb # !HardReset_B ;
!FEthRstOut2_B = !FEthRst2_B.fb # !HardReset_B ;

"*******************************************************************************
"* Hard reset configuration
"*******************************************************************************
"equations

"RstConf_B.oe = H;

"RstConf_B = L;

"*******************************************************************************
"* NMI generation
"*******************************************************************************
equations

NMI_B.oe = NMIEn ;

NMI_B = 0 ;" O.D.
```

```
NMIEn = !RstDeb1.com & AbrDeb1.com ;" only abort button depressed

"*****************************************************************************
"* local data buffers enable
"*****************************************************************************
equations

SyncHardReset_B.clk = SYSCLK ;
SyncHardReset_B.ar = 0;
SyncHardReset_B.ap = 0;

DSyncHardReset_B.clk = SYSCLK ;
DSyncHardReset_B.ar = 0;
DSyncHardReset_B.ap = 0;

SyncHardReset_B := HardReset_B ;
DSyncHardReset_B := SyncHardReset_B.fb ;

DataBufEn_B.oe = H ;

!DataBufEn_B = ( !Cs0_B #     " covers also hard reset config
  !Cs4_B #
 !BrdContRegCs_B #
 !IntContCs_B #
  !AtmUniCsOut_B #   " provides data-hold for write
  !ToolCs1_B #
  !ToolCs2_B ) &
       ( !BUFFER_HOLD_OFF ) ;

ToolDataBufEn_B.oe = H ;

!ToolDataBufEn_B = ( !ToolCs1_B #
                     !ToolCs2_B ) &
    HardReset_B.pin &
   ( !BUFFER_HOLD_OFF ) ;

"*****************************************************************************
"* local data buffers disable (data contention protection)
"*****************************************************************************
"* Since with Voyager, hard-reset conf is read from flash/eeprom during HRESET
"* asserted and since these are all consequitive read cycles and since
"* the cycles following hard reset are also reads (boot) the hold-off
"* state machine may be left in NO_HOLD_OFF for HRESET_B asserted duration
"* without warrying about contention between flash and data buffers.

equations

HoldOffCnt.clk = SYSCLK ;
HoldOffCnt.ar = 0;
HoldOffCnt.ap = 0;

HoldOffTc = (HoldOffCnt.fb == 3) ;

when ( (((END_OF_FLASH_EEPROM_READ # END_OF_ATM_READ )
       & (HoldOffCnt.fb == 0)) #
         (HoldOffCnt.fb != 0)) & !(HoldOffCnt.fb == 4) & DSyncHardReset_B.fb  ) then
      HoldOffCnt := HoldOffCnt.fb + 1 ;
else
```

```
    HoldOffCnt := 0 ;

"*****************************************************************************
"* Flash/EEPROM Chip Selects
"*****************************************************************************
equations

FlashCsOut.oe = ^hf ;

!FlashCs1_B = CS0_ASSERTED & FLASH_BANK1 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK1 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
    CS4_ASSERTED & FLASH_BANK1 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

!FlashCs2_B = CS0_ASSERTED & FLASH_BANK2 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK2 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
          CS4_ASSERTED & FLASH_BANK2 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

!FlashCs3_B = CS0_ASSERTED & FLASH_BANK3 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK3 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
          CS4_ASSERTED & FLASH_BANK3 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

!FlashCs4_B = CS0_ASSERTED & FLASH_BANK4 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK4 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
          CS4_ASSERTED & FLASH_BANK4 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

EEpromCs_B.oe = H ;

!EEpromCs_B = CS0_ASSERTED & HRESET_BOOT_IN_EEPROM #
    CS0_ASSERTED & BOOT_IN_EEPROM & DSyncHardReset_B.fb #
    CS4_ASSERTED & (HRESET_BOOT_IN_FLASH # BOOT_IN_FLASH) ;

"*****************************************************************************
"* ATM UNI Chip Select
"*****************************************************************************
equations

AtmUniCsOut_B.oe = H ;

!AtmUniCsOut_B = !AtmUniCsIn_B;

"*****************************************************************************
"* Power On Reset
"*****************************************************************************
equations

S_PORIn_B.clk = SYSCLK ;
S_PORIn_B.ar = 0;
S_PORIn_B.ap = 0;

S_PORIn_B := PORIn_B ;

"*****************************************************************************
"*****************************************************************************
"* Generating Interrupt Request to the PQ2.
"*****************************************************************************
"*****************************************************************************
equations

PCI_Interrupt = (Slot0IntA #
```

```
 Slot0IntB #
 Slot0IntC #
 Slot0IntD #
 Slot1IntA #
 Slot1IntB #
 Slot1IntC #
 Slot1IntD #
 Slot2IntA #
 Slot2IntB #
 Slot2IntC #
 Slot2IntD) ;

PCI_IRQ_B.oe = PCI_Interrupt ;  " Open-Drain output
!PCI_IRQ_B = PCI_Interrupt ; " Interrupt Request shows after OE

"*****************************************************************************
"* Auxiliary functions
"*****************************************************************************
equations

KeepPinsConnected = TEA_B & BCTL1 # KeepPinsConnected.com;

"*****************************************************************************

END
```

## 8.2.2 U41—Power switch debounce

```
MODULE Power_Debouncer

TITLE 'MPC8272 Power Debouncer'

"*****************************************************************************
"* Device declaration.                                                       *
"*****************************************************************************

"*****************************************************************************
"*****************************************************************************

"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************
SYSCLK                  PIN  5 ;

ChasisPowerIn_B PIN  15 ;   "Chassis Power Switch
PowerOn_B PIN  16  istype 'reg' ;  " Power Supply Power-On

"*****************************************************************************
"*****************************************************************************

"*****************************************************************************
"* Chassis Power Switch Buffer.
"*****************************************************************************
```

```
Power_Buffer NODE istype 'reg,buffer' ;

"*****************************************************************************
"* Creating internal clock generator.
"*****************************************************************************

inv1 NODE istype 'com,keep' ;
inv2 NODE istype 'com,keep' ;
inv3 NODE istype 'com,keep' ;
inv4 NODE istype 'com,keep' ;
inv5 NODE istype 'com,keep' ;

counter0,
counter1,
counter2,
counter3,
counter4,
counter5,
counter6,
counter7,
countera0,
countera1,
countera2,
countera3,
countera4,
countera5,
countera6,
countera7,
counterb0,
counterb1,
counterb2,
counterb3,
counterb4,
counterb5,
counterb6,
counterb7       NODE istype 'reg,buffer' ;

"*****************************************************************************
"*****************************************************************************
H, L, X, Z = 1, 0, .X., .Z. ;
C, D, U    = .C., .D., .U. ;
"*****************************************************************************
"* SIMULATION = 1 ;
"*****************************************************************************
"* Signal groups
"*****************************************************************************


counter    = [counter7,counter6,counter5,counter4,
              counter3,counter2,counter1,counter0] ;
countera   = [countera7,countera6,countera5,countera4,
              countera3,countera2,countera1,countera0] ;
counterb   = [counterb7,counterb6,counterb5,counterb4,
              counterb3,counterb2,counterb1,counterb0] ;

"*****************************************************************************
"* ATX Power Declarations.
"*****************************************************************************
```

```
PowerOn  = 0 ;
PowerOff = 1 ;

"*****************************************************************************
"* Equations, state diagrams.                                                *
"*****************************************************************************
"*****************************************************************************
"*****************************************************************************

equations

"*****************************************************************************
"*****************************************************************************
"* Generating PowerOn signal to the ATX Power Supply.
"*****************************************************************************
"*****************************************************************************
equations

inv1 = !inv5.com ;" generating internal clock oscilator
inv2 = !inv1.com ;
inv3 = !inv2.com ;
inv4 = !inv3.com ;
inv5 = !inv4.com ;

counter.ar = 0 ;
counter.ap = 0 ;
counter.clk = !inv5.com ;

when ( counter.fb == 255 ) then counter := 0 else counter := counter + 1 ;

countera.ar = 0 ;
countera.ap = 0 ;
countera.clk = ( counter.fb == 0 ) ;

when ( countera.fb == 255 ) then countera := 0 else countera := countera + 1 ;

counterb.ar = 0 ;
counterb.ap = 0 ;
counterb.clk = ( countera.fb == 0 ) ;

when ( counterb.fb == 255 ) then counterb := 0 else counterb := counterb + 1 ;

Power_Buffer.ar = 0 ;
Power_Buffer.ap = 0 ;
Power_Buffer.clk = ( counterb.fb == 0 ) ;
Power_Buffer := ChasisPowerIn_B ;

PowerOn_B.oe  = H ;
PowerOn_B.ar  = 0 ;
PowerOn_B.ap  = 0 ;
PowerOn_B.clk = ( counterb.fb == 0 ) ;
PowerOn_B := !Power_Buffer.fb ;


"*****************************************************************************
"*****************************************************************************
@ifdef SIMULATION {
}
END
```

# Appendix A
# Revision History

This appendix provides a list of the major differences between revisions of the *MPC8272ADS User Guide*.

## A.1 Revision Changes From Revision 0 to Revision 0.1

Changes to the *MPC8272ADS User Guide* from Revision 0 to Revision 0.1 are as follows:

| Section, Page | Changes |
|---|---|
| Throughout manual | Non-technical reformatting. In addition, an index was added. |

# Index


## Numerics

-12V indicator, 4-4
12V indicator, 4-4
3.3-V indicator, 4-4
3IDDL Measurement, 4-2
5-V indicator, 4-4
60X bus mode, 1-2

## A

abbreviations, xvii
ABORT interrupt, 5-9
ABORT Switch, 4-1
acronyms, xvii
add-in cards, 7-2, 7-3, 7-5
ATM ON indicator, 4-5
ATM UNI interrupt, 5-9
ATMEN, 5-22
ATX power supply
  connector, 8-15
  on/off switch, 2-2
  power OK indicator light, 4-4
auto-refresh mode, 5-17

## B

BCSR, 1-3, 2-1, 2-4, 2-8
BCSR/Flash power-on reset configuration, 6-6
BCSR0, 4-6
BCSR1, 4-5
block diagram, 1-4
boot
  sequence, 2-4
  source, 2-5
buffering, 5-14
bus configuration, 5-14

## C

CBR refresh commands, 5-16
chip-select
  generator, 5-15, 6-5
  line, 5-10
clock generator, 5-13
clock-in
  source selection, 2-7
  source types, 2-2
communication ports, 5-22
configuration
  host computer, 3-1
  stand alone, 3-2
connectors
  ATX 155 port, 7-5
  ATX power, 7-4, 7-6
  COP/JTAG, 7-5
  CPM expansion, 7-5
  Fast Ethernet port, 7-4
  functions, 7-4
  ISP, 7-5
  logic analyzer, 7-5
  parallel port, 7-6
  PCI, 7-5
  RS232, 7-5
  system expansion, 7-6
  USB, 7-6
controls and indicators, 4-1
conventions, xvi
COP controller, 3-1
COP/JTAG
  connector, 3-3, 4-7, 8-2
  connector on-board, 1-2
  establishing connection, 2-9
  port, 5-2
CPM Expansion connector, 8-4
CPM PLLs, 2-1
current consumption, maximum for expansion connectors, 7-2

## D

debounce, 8-23
debugger, 1-1, 3-1
dimensions, 1-2
dip-switch settings, 2-1
DM9161 control, 5-23
dual Fast Ethernet controller, 6-5

## E

E2PROM
  as boot device, 6-8
  configuration words, 5-4
  general features, 1-2
  memory, 5-19
  power-on reset configuration, 6-6
EEPROM, 2-5
equations, 8-23
Ethernet
  mode selection, 2-7, 2-8
  PHY on FCC1, 2-7
  PHY on FCC2, 2-8
  port 2 LINK indicator, 4-7
  transceivers, 5-9
external debugger connection indicator, 4-7

## F

Fast Ethernet ports, 4-5
Flash memory
  access time provided with MPC8272ADS, 5-17
  demonstration tool, 1-1
  installation module on MPC8272ADS, 3-4
  programming voltage, 5-19
  SIMM, 2-5, 6-1
  SIMM Installation, 3-4
  stand alone operation, 3-2
Flash SIMM, 1-2, 2-5, 4-3, 6-7
forced parallel port connection, 2-9
full duplex indicator, 4-5

## G

general purpose Led 2 indicator, 4-6
GND bridges, 4-4

## H

hard reset
  configuration source, 2-4
  sources, 5-2
  switches, 4-1
hardware preparation, 2-1
Hip7, 1-1, 1-2
host parallel port, 3-2
HRESET, 2-4, 5-2, 5-22
humidity, 1-2

## I

IDDH measurement, 4-3
installation
  host-controlled operation, 3-1
  stand-alone environment, 3-1, 3-2
  working environments, 3-1
insulated GND clips, 4-4
interconnect signals, 8-1
internal logic of MPC8272ADS, 7-3
internal reset sources
  configuration, 5-1, 5-3
interrupt controller, 5-9
Interrupt Mask Register, 5-21
IRQ0 input, 5-9

## J

jumper JP12, 2-2
jumper selectable, 1-2

## L

LINK indicator, 4-7
local interrupter, 5-9
logic analyzer MICTOR connectors, 8-13
logic equations, 8-23

## M

Mach/Lattice ISP connector, 8-16
manual reset, 5-1
memory
  controller register programming, 6-7
  E2PROM, 6-1
  Flash, 6-1
  mapping in the MPC8272ADS, 6-1
  SDRAM, 6-1
  slaves, 6-1
microprocessor, 1-1
MICTOR connectors, 8-13
MII or RMII interface, 2-7, 2-8
MODCK(1
  3), 2-3, 2-4
mode-set command, 5-16
MPC8272 clock, 5-13
muxing devices, 5-22

## N

NMI, 5-9

## O

off-board application development, 7-2
on-board clock oscillator, 2-7
on-board logic, 1-2
on-chip arbiter, 5-20
open-collector gate, 5-9
open-drain gate, 5-9

## P

parallel port connection, 4-7
PCI
  bridge, 5-21
  Bus, 5-20
  bus, 5-8, 7-2
  bus layout, 5-21
  bus reset, 5-1
  connectors, 8-13
  expansion board, 5-9
  interrupt, 5-9
  interrupt controller, 8-23
  interrupt register description, 5-10
  IRQ routing, 5-10
  mode-enabled, 2-6
  slots, 5-9, 5-10
PCI Standard, 7-2
PCI Standard 2.2-compliant, 1-2
PCI_ARBITER, 2-1, 2-6
PCI_DLL for PCI mode-enabled, 2-6
peripherals, 7-3
PHY interrupt, 5-9
physical properties of MPC8272ADS, 7-1
PLLs multiplication factor, 2-3
potentiometer RP1, 2-1
power
  O.K. indicator, 4-4
  on/off switch, 2-9
  rails on the board, 7-1
  scheme of MPC8272ADS, 7-2
power supply
  ATX, 7-1
  sources, 7-1
power-on reset
  description, 5-1
  switch, 4-1
PowerPC, 1-1
PowerQUICC II
  boot code source, 2-1
  COP port, 2-9
  COP/JTAG connection, 2-2
  hard reset configuration word source, 2-1
  internal logic supply level, 2-1
  PCI arbiter, 2-1
  PCI_DLL, 2-2
  PowerPC core, 1-1
  register programming, 6-5
  single mode, 5-4
PPC bus SDRAM controller, 6-5
precharge-all command, 5-16
programmable logic equations, 8-23

## R

register programming, 6-5
Reset Configuration Switch, 4-2
reset sources on the MPC8272ADS, 5-1
revision history of this document, A-1
RP1 rotation direction, 2-3
RS232
  connection, 3-3
  port, 4-5
  serial port connector, 3-3
RUN indicator, 4-5

## S

SDRAM programming, 5-16
single PowerQUICC II mode, 5-14
SIU Register Programming, 6-7
Smart Modular Technology, 5-17
Soft reset
  sources, 5-7
SOFT RESET Switch, 4-1
Software Options switch, 1-3, 4-2
standby rail, 7-3
static discharge, 2-1
SW2 description, 2-6
SW5 description, 2-4
switches
  ABORT, 4-1
  hard reset, 4-1
  power-on reset, 4-1
  reset configuration, 4-2
  soft reset, 4-1
  software options, 4-2
synchronous DRAM, 5-15
synchronous Dynamic RAM DIMM, 1-1
system expansion connector, 8-17
system initialization, 6-6

## T

temperature
  operating, 1-2
  storage, 1-2
TEPBGA package, 1-1
thermal sense connector, 4-2
top side part location diagram, 2-2
trimming potentiometer, 7-3
Tx/Rx indicator, 4-7

## U

UART for terminal or host computer connection, 6-5
unpacking and inspection, 2-1
USB
  connectors, 8-22

controller, 6-5
enabled indicator, 4-6
host/slave mode, 2-2
mode selection, 2-8
port, 1-3, 2-8
power indicator, 4-4
speed selection, 2-8
speed software, 2-2
UTOPIA multi PHY indicator, 4-4

## V

VDDH rail, 7-3
VDDL
bus, 7-3
indicator, 4-7
trimmer, 2-3
voltage supply level, 2-3
voltage
changing core logic of the MPC8272, 7-3
internal logic operation, 1-3
VPP source selector, 4-3