PrPMC880 Processor PMC Module

Installation and Use

PRPMC880A/IH1

January 2004 Edition

Printed in the United States of America.

Safety Summary

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

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

Ground the Instrument.

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

Do Not Operate in an Explosive Atmosphere.

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

Keep Away From Live Circuits Inside the Equipment.

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

Use Caution When Exposing or Handling a CRT.

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

Do Not Substitute Parts or Modify Equipment.

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

Observe Warnings in Manual.

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

Warning

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

CE Notice (European Community)

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

EN55022 “Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment”; this product tested to Equipment Class B

EN55024 “Information technology equipment—Immunity characteristics—Limits and methods of measurement”

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

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

Flammability

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

EMI Caution

Caution

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

Notice

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

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

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

Limited and Restricted Rights Legend

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

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

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

Contents

CHAPTER 4 Connector Pin Assignments

CHAPTER 5 MOTLoad Overview

CHAPTER 6 Modifying the Environment

CHAPTER 7 Memory Map and Programming Details

CHAPTER 8 MOTLoad Vital Product Data

APPENDIX A Specifications

APPENDIX B Thermal Validation

APPENDIX C Related Documentation

List of Figures

List of Tables

About This Manual

This *PrPMC880 Installation and Use* guide provides both general and functional descriptions of the product along with installation instructions, connector pin assignments, specifications, memory maps and programming information for the PrPMC880 Processor PMC Module.

The information contained in this manual applies to the following model configurations:

Model Number	CPU/Speed	Functional Description
PrPMC880-2261	7447, 1 GHz	256MB DDR SDRAM, Front/Rear Gigabit Ethernet and Serial Port
PrPMC880-2271	7447, 1 GHz	512MB DDR SDRAM, Front/Rear Gigabit Ethernet and Serial Port
PrPMC880-3261	7447, 1 GHz	256MB DDR SDRAM, Dual Rear Gigabit Ethernet and Serial Port
PrPMC880-3271	7447, 1 GHz	512MB DDR SDRAM, Dual Rear Gigabit Ethernet and Serial Port
PrPMC-CABLE-005		Cable for Front Ethernet and Serial (console) Port to RJ-45 connector

This manual is intended for anyone who designs OEM systems, supplies additional capability to existing compatible systems, or works in a lab environment for experimental purposes. It is important to note that a basic knowledge of computers and digital logic is assumed.

Overview of Contents

The following paragraphs briefly summarize the contents of each chapter and appendix in this manual.

Chapter 1, *Preparation and Installation*, provides a brief description of the PrPMC880 including a discussion of the difference between Monarch and non-Monarch modes. It also provides basic startup and hardware preparation information. The remainder of the chapter describes the installation procedure for the PrPMC880.

Chapter 2, *Operating Instructions*, provides a description of basic operational characteristics of the PrPMC880 including power-on sequence, sources of reset, status indicators and the DEBUG port.

Chapter 3, *Functional Description*, lists the features of the PrPMC880 provided by the major on-board components. It also provides a brief general description and a block diagram. The remainder of the chapter is an overview of each functional characteristic of the board along with a description of what component or components provide each function.

Chapter 4, *Connector Pin Assignments*, provides a description and pin-by-pin listing of all functional connectors on the board. Some connectors are designated as optional where applicable.

Chapter 5, *MOTLoad Overview*, provides a description of the MOTLoad firmware including a listing of the initialization sequence, basic usage, and a listing of the general commands that are specific to MOTLoad.

Chapter 6, *Modifying the Environment*, discusses the (Global Environment Variables (GEVs) used to control the environment. This chapter provides information on creating, editing, viewing, and deleting GEVs.

Chapter 7, *Memory Map and Programming Details*, provides a description of memory maps and programming information on arbitration, interrupt handling, flash memory, system memory, and system registers.

Chapter 8, *MOTLoad Vital Product Data*, provides information contained in the VPD, specific to the MOTLoad firmware package. This chapter also tells you how to archive, modify, and restore VPD.

Appendix A, *Specifications*, provides general specifications of the PrPMC880 including those relating to mechanical, electrical and temperature characteristics.

Appendix B, *Thermal Validation*, provides information on thermally significant components and an overview of how to measure various junction and case temperatures.

Appendix C, *Related Documentation*, provides a listing of related Motorola product documentation, manufacturer's documents, and industry standard specifications.

Comments and Suggestions

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

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

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

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

Terminology

Throughout this manual, a convention is used which precedes data and address parameters by a character identifying the numeric format as follows: For example, “12” is the decimal number twelve, and “$12” is the decimal number eighteen.

Unless otherwise specified, all address references are in hexadecimal.

A pound sign (#) following the signal name for signals which are level significant denotes that the signal is true or valid when the signal is low.

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

In some places in this document, an underscore (**_L**) following the signal name is used to indicate an active low signal.

In this manual, assertion and negation are used to specify forcing a signal to a particular state. These terms are used independently of the voltage level (high or low) that they represent. In particular:

assertion and assert refer to a signal that is active or true

negation and negate indicate a signal that is inactive or false.

Data and address sizes for MCP74xx chips are defined as follows:

- A *byte* is eight bits, numbered 0 through 7, with bit 0 being the least significant.
- A *half-word* is 16 bits, numbered 0 through 15, with bit 0 being the least significant.
- A *word* or s*ingle word* is 32 bits, numbered 0 through 31, with bit 0 being the least significant.
- A *double word* is 64 bits, numbered 0 through 63, with bit 0 being the least significant.

The terms *control bit* and *status bit* are used extensively in this document.

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

Conventions Used in This Manual

The following typographical conventions are used in this document:

bold

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

italic

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

`courier`

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

<Enter>, <Return> or <CR>

<CR> represents the carriage return or Enter key.

CTRL

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

Preparation and Installation

1

This chapter provides a brief description of the PrPMC880 Processor PMC Module, and instructions for preparing and installing the hardware.

In this manual, the name PrPMC880 refers to all models of the PrPMC880 series boards, unless otherwise specified. These are add-on modules intended for use with any host board that will accept a monarch Processor PMC (PrPMC) module.

PrPMC880 Description

The PrPMC880 is a PMC form factor processor module based on the PowerPC MPC7445/7447 processor and the Discovery II System Controller (SMC), PCI Host Bridge (PHB) and Gigabit Ethernet ASIC. This module is a single wide, standard length PMC and is standard height.

The PrPMC880 features:

- ❏ One bank of onboard ECC-protected SDRAM that provides 1GB of memory using 512Mbit x 9 SDRAM devices
- ❏ Two Gigabit Ethernet ports for high speed network applications
- ❏ PCI/PCI-X bus interface up to 133 MHz
- ❏ 133 MHz system bus interface
- ❏ Up to 64MB of onboard flash using 32MB x 2 devices
- ❏ One UART with RS-232-level interface for debug capabilities

The PrPMC880 interfaces to the host board PCI bus via the PMC P11, P12 and P13 connectors that provide a 64-bit, PCI/PCI-X interface between the host board and the PrPMC880. The PrPMC880 draws +3.3V through the PMC connectors as the power source of the entire board. The onboard processor core power supply derives the core voltage from the +3.3V power. The clock generator derives all of the required onboard clocks from an onboard crystal oscillator.

The PrPMC880 can be configured to operate as a system controller or a slave processor depending on the state of the PMC MONARCH# signal P12:64 from the PMC connector. When configured as the system controller, the PrPMC880 will monitor all four PCI interrupts but can drive PCI interrupt A only.

Monarch and Non-Monarch PrPMCs

The traditional concept of host/master and slave/target processors changes with the inception of the processor PMC because the arbiter or clock source, traditionally located on the host board, does not reside on the PrPMC880. The VITA 32 specification defines the terms monarch and non-monarch to refer to these two modes of operation for PrPMCs. A monarch PrPMC is defined as the main PCI bus PrPMC (or CPU) that performs PCI bus enumeration at power-up or reset and acts as the PCI interrupt handler. The non-monarch is a slave/target processor that does not perform bus enumeration and does not service PCI interrupts but may generate a PCI interrupt to the host processor.

A system may have one monarch PrPMC, in this case a PrPMC880, and one or more non-monarchs creating a loosely coupled multiprocessing system. A PrPMC880 operating as a monarch may be mated to a host board with slave processors, PCI and other I/O devices. For information on configuring the PrPMC880, refer to *MONARCH#* on page 3-12 of this manual for more information.

Host Board Requirements

A host board (also referred to as a carrier board) must provide the standard PCI interface, including +3.3V power, PCI address/control, a PCI clock, and a PCI arbiter REQ/GNT pair. The host board must also ground the MONARCH# pin to enable the monarch operating mode. Leaving the MONARCH# pin open will enable the non-monarch mode.

System Enclosure

The type of system enclosure you use is determined by the configuration and architecture of the host board (either VME, CompactPCI, or custom). In some cases, the host board and PrPMC880 assembly requires only a single slot in a VME or CompactPCI chassis. A customized chassis may accommodate a slightly wider board assembly into each slot. For more information refer to the PrPMC specification, as referenced in Appendix C, *Related Documentation*.

Overview of Start-Up Procedures

The following table lists the things you will need to do before you can use this board, and tells where to find the information you need to perform each step. Be sure to read this entire chapter and read all Caution and Warning notes before beginning.

What you need to do ...	**Refer to ...**
Unpack the hardware	*Guidelines for Unpacking* on page 1-4
Make any settings or adjustments on the PrPMC880 module.	*Preparing the PrPMC880 Hardware* on page 1-6
Prepare any other optional devices or equipment you will be using.	For more information on optional devices and equipment, refer to the documentation provided with that equipment.
Install the PrPMC880 on the host board.	*Installation of PrPMC880 on a Host Board or PPMC Module* on page 1-9
Connect any other optional devices or equipment you will be using.	Chapter 2, *Operating Instructions* Chapter 4, *Connector Pin Assignments*
	For more information on optional devices and equipment, refer to the documentation provided with that equipment.
Power up the system.	*Status Indicators* on page 2-5

What you need to do ...	Refer to ...
Familiarize yourself with the MOTLoad command line interface and MOTLoad image information.	Review Chapter 5, *MOTLoad Overview* Obtain the MOTLoad Firmware Package User's Manual, listed in Appendix C, *Related Documentation*.
Examine the environmental parameters and make any changes needed.	Chapter 6, *Modifying the Environment*
Program the PrPMC880 module and PMCs as needed for your applications.	Chapter 7, *Memory Map and Programming Details*
	You may also want to obtain Manufacturer's product information as listed in Appendix C, *Related Documentation*.

Guidelines for Unpacking

If the shipping carton is damaged upon receipt, request that the carrier's agent be present during the unpacking and inspection of the equipment.

Caution

When unpacking, avoid touching areas of integrated circuitry; static discharge can damage circuits.

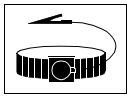
Refer to the packing list and verify that all items are present. Save the packing material for storing and reshipping of equipment.

Installation Preliminaries

This section applies to all hardware installations you may perform that involve the PrPMC880 Processor PMC Module and host board.

If the host board is a hot-swap module, you can install it or remove it without shutting down the operating system or removing system power. Replacing a hot-swap module can be accomplished in under five minutes. For more information about hot swap concepts and the PCI Industrial Computer Manufacturer's Group Hot Swap Specification (PICMG 2.1 R2.0) refer to the sources listed in Appendix C, *Related Documentation*.

Use ESD



Wrist Strap

Motorola strongly recommends that you use an antistatic wrist strap and a conductive foam pad when installing or upgrading a system. Electronic components, such as disk drives, computer boards, and memory modules, can be extremely sensitive to electrostatic discharge (ESD). After removing the component from its protective wrapper or from the system, place the component flat on a grounded, static-free surface (and, in the case of a board, component side up). Do not slide the component over any surface.

If an ESD station is not available, you can avoid damage resulting from ESD by wearing an antistatic wrist strap (available at electronics stores) that is attached to an active electrical ground. Note that a system chassis may not be grounded if it is unplugged.

Equipment Required

To install the PrPMC880, you need the following equipment.

- ❏ System enclosure with power supply
- ❏ Host board
- ❏ Display console
- ❏ Console/Ethernet Y-cable

Preparing the PrPMC880 Hardware

To produce the desired configuration and ensure proper operation of the PrPMC880, you may need to carry out certain modifications before and after installing the modules.

The following paragraphs discuss the preparation of the PrPMC880 hardware components prior to installing them into a chassis and connecting them.

The PrPMC880 has been factory tested and is shipped with the configurations described in the following sections. It contains a factory-installed start-up firmware, MOTLoad, which operates with those factory settings.

The PrPMC880 provides software control over most options. By setting bits in control registers after installing the PrPMC880 in a system, you can modify its configuration. The PrPMC880 control registers are described in Chapter 7, *Memory Map and Programming Details* in this manual.

Connector and Header Location

The following figure illustrates the placement of connectors and headers on the PrPMC880.

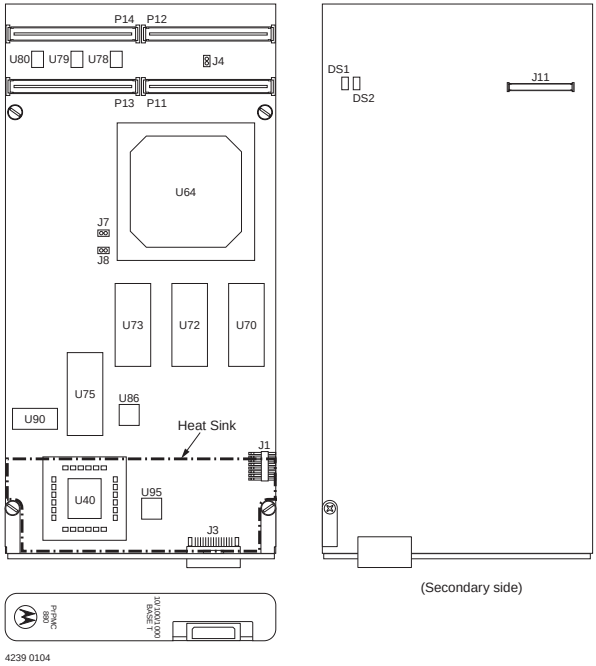


Figure 1-1. PrPMC880 Headers, Connectors and Components

Table 1-1. PrPMC880 Connectors, Headers, and LEDs

Connector	Description
J1	COP header connector
J3 (option)	Front panel Ethernet and COM connector
J4	Flash Write Protect
J7	JTAG Router
J8	Boundary scan configuration header

Table 1-1. PrPMC880 Connectors, Headers, and LEDs

Connector	Description
J11	Debug connector, secondary side
P11, P12, P13	PCI/PCI-X Interface
P14	Serial I/O (Ethernet and COM signals routing to host board for rear Ethernet model)
DS1	User defined LED
DS2	Board fail (BD_FAIL) LED

Table 1-2. J7 and J8 Jumpering

Jumpering	Headers	Description
0 0	J7=0, J8=0	00=All-out float
0 1	J7=0, J8=1	PLD only connected to boundary scan TAP
1 0	J7=1, J8=0	Flash only devices connected to boundary scan TAP
1 1	J7=1, J8=1	All scannable devices connected to daisy chain
Note 0=jumpered 1=unjumpered		

Installation and Removal

The following instructions tell how to install or replace the PrPMC880 on a typical host board or PPMCBase.

Installation of PrPMC880 on a Host Board or PPMC Module

To install a PrPMC880 on a Compact PCI or VME host board or PPMCBase module, refer to the figure in this procedure, read all cautions and warnings, and perform the following steps. This figure is for reference only and may not represent the exact host board you are using.

Warning

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

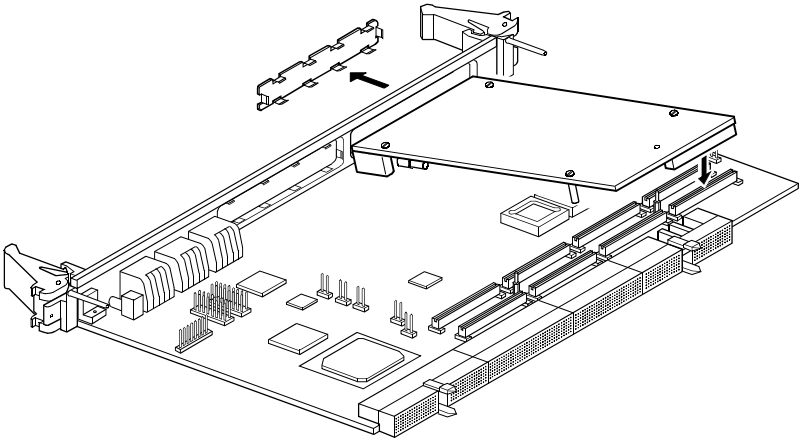
Caution

Inserting or removing modules with power applied may result in damage to module components.

Caution

Avoid touching areas of integrated circuitry; static discharge can damage the circuits.

1. Attach an ESD strap to your wrist. Attach the other end of the ESD strap to the chassis as a ground. The ESD strap must be secured to your wrist and to ground throughout the procedure.



2. Perform an operating system shutdown. Turn the AC or DC power off and remove the AC cord or DC power lines from the system. Remove the chassis or system cover(s) as necessary to gain access to the PPMC module or host board.

3. Carefully remove the host board from its card slot and place it on a clean and adequately protected working surface (preferably an ESD mat) with the backplane connectors facing you.

4. Place the PrPMC880 board on top of the host board and make sure to align the PMC connectors on both boards. Connectors P11, P12, P13, and P14 on the inside edge of the PrPMC880 should connect smoothly with the corresponding connectors on the VME module or CompactPCI host board.

5. On the secondary side of the host board, install the Phillips-head screws through the holes in the host board and the spacers. Tighten the screws.

Caution

It is critical that two prerequisite steps be performed prior to installing your board into the CompactPCI backplane to prevent possible backplane pin damage.

- ❏ Visually inspect the board connectors to ensure they are not damaged by previous insertions or accidental mishandling. If any board connector damage is observed, do not install board into the backplane. This may cause a bent pin on the connector, resulting in an expensive repair.
- ❏ Visually inspect the backplane pins for any bent pins from previous board installations in the slot where the board will be installed.

When it is determined that there are no bent pins in the slot of the backplane or connector damage on the board being inserted, proceed to insert the board into the backplane. Insert the board into the card guide rail and carefully slide the board toward the backplane connectors until the handles engage with the chassis. To seat the board, apply forward pressure while pushing the handles toward each other which seats the board into the backplane.

MCG and our suppliers have taken significant steps to ensure there are no bent pins on the backplane or any connector damage to the boards prior to leaving our factory. Any bent pins caused by improper installation or by inserting boards with damaged connectors could void the MCG warranty for the backplane or boards.

6. Install the PrPMC and host board assembly in its proper card slot. Ensure it is seated properly in the backplane connectors. Do not damage or bend connector pins.
7. Replace the chassis or system cover(s) and connect the system to the AC or DC power source. Turn the equipment power on.

Removal of PrPMC880 from a Host Board or PPMC Module

To remove a PrPMC880 from a Compact PCI or VME host board or PPMC module, refer to the figure in *Installation of PrPMC880 on a Host Board or PPMC Module*, read all cautions and warnings, and perform the following steps.

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

Inserting or removing modules with power applied may result in damage to module components.

Avoid touching areas of integrated circuitry; static discharge can damage the circuits.

1. Attach an ESD strap to your wrist. Attach the other end of the ESD strap to the chassis as a ground. The ESD strap must be secured to your wrist and to ground throughout the procedure.
2. Perform an operating system shutdown. Turn the AC or DC power off and remove the AC cord or DC power lines from the system. Remove the chassis or system cover(s) as necessary to gain access to the PPMC module or host board.
3. Carefully remove the host module from its card slot and place it on a clean and adequately protected working surface (preferably an ESD mat) with the secondary side of the board facing up.
4. Remove the four Phillips-head screws from the holes in the host board.

5. Carefully turn the host board to the primary side and place on your working surface. Gently separate the PrPMC880 from the PMC connectors on the host board. Do not damage or bend connector pins.
6. Tilt the board up slightly and remove it from the front panel slot.

Connecting to a Console

The PrPMC880 lets you monitor system conditions through the serial port connector on the front panel. This connector is configured as DTE.

Configure your console or terminal emulation software using the parameters shown in Table 1-3. The default MOTLoad serial port settings are described in the next table.

Table 1-3. Serial Port Configuration Parameters

Parameter	Setting
Baud rate	9600
Data bits	8
Parity	None
Stop bits	1
Flow control	None

Refer to Figure 1-1 on page 1-4 for a diagram showing the location of the Ethernet connector on the PrPMC880 front panel.

By using the optional Y-cable (PRPMC-CABLE-005) you can connect any available serial port of an external host (for example, a laptop) to the J3 connector on the PrPMC880 front panel. The Y-cable has two RJ-45 connectors, one labeled COM and the other labeled ENET. To connect to a COM port using the Y-cable, an RJ-45 to DB-9 null modem adapter may be required. Refer to *PRPMC-CABLE-005* on page 4-8 for pin assignments.

Operating Instructions 2

This chapter provides information about powering up the PrPMC880, reset sources, functionality of the status indicators, and the I/O port on the PrPMC880 module.

Applying Power

After you have verified that all necessary hardware preparation has been done, that all connections have been made correctly, and that the installation is complete, you can power up the system, refer to *Overview of Start-Up Procedures* for a helpful checklist. The MPU and hardware initialization process is performed by the MOTLoad firmware at power-up or system reset. The firmware initializes the devices on the PrPMC880 module in preparation for booting the operating system.

The firmware is shipped from the factory with an appropriate set of defaults. In most cases there is no need to modify the firmware configuration before you boot the operating system. Refer to Chapter 6, *Modifying the Environment* for further information about modifying defaults.The following list shows the basic initialization process that takes place during PrPMC880 system start-ups.

- ✓ Initialize Processor (7447)
- ✓ North Bridge (MV64360)
- ✓ Initialize memory and scrub
- ✓ Decompress ROM to RAM and execute
- ✓ Build Vector table
- ✓ Initialize kernel
- ✓ Initialize memory allocator

- ✓ Initialize PAL
  - – Invalidate L2 cache, initialize and enable MMU
  - – Initialize PCI address space library
  - – Initialize static hardware, create static device nodes
  - – Enumerate PCI busses (allocate dress space, assign interrupts
- ✓ Create root task
- ✓ Start scheduler (starts root task)
- ✓ Initialize application libraries (log message, WDT, signals
- ✓ Start generic device probe task, create dynamic device nodes
- ✓ Start user task (command shell)

For further information on MOTLoad, refer to Chapter 5, *MOTLoad Overview* or to the MOTLoad documentation listed in Appendix C, *Related Documentation*.

Auto Boot

Auto Boot provides an independent mechanism for booting an operating system where no console is required. MOTLoad does not provide an explicit Auto Boot command, flag, or parameter. Instead, Auto Boot is established by appropriately defining the *mot-script-boot:* Global Environment Variable (GEV). Refer to Chapter 6, *Modifying the Environment* for more information on GEVs and MOTLoad commands.

Upon start-up, MOTLoad checks *mot-script-boot* in the GEV section of NVRAM for boot commands. Boot commands that MOTLoad recognizes are:

- ❑ **diskBoot**
- ❑ **netBoot**

When either command is detected, MOTLoad performs the selected boot command using arguments specified from either the command line command line argument (stored in mot-script-boot with the command) or from a GEV. If neither provides the necessary arguments, use the **help**

command on diskBoot or netBoot to see the defaults. Since not all command arguments can be specified by GEV, defaults values are used where *mot-script-boot:* does not contain the argument's value.

To Auto Boot from a floppy, disk, or CD-ROM use the **diskBoot** command. MOTLoad selects the boot device from a scan list provided as part of the command line arguments (if stored in *mot-script-boot*) or from the diskBoot's corresponding GEV variable, *mot-boot-path*. Refer to the *Reserved GEVs* on page 6-2 for more information on this GEV variable.

To Auto Boot across the Ethernet use the **netBoot** command. The command line parameters that can be specified by GEVs are listed in *Reserved GEVs* on page 6-2.

To create the GEVs to use with either boot command, MOTLoad provides the *gevEdit* command. Existing GEVs can be viewed using *gevShow*.

When using MOTLoad's Auto Boot mechanism, MOTLoad delays execution of the commands by the amount of time (in seconds) defined in *mot-script-delay:*. If *mot-script-delay* is not defined, the default of 7 seconds is used. During this time the boot process can be cancelled by pressing the ESC key.

The following shows an example of setting up an Auto Boot from disk:

```
PPMC880>gevEdit mot-script-boot
   (Blank line terminates input.)
   diskBoot<cr>
   <cr>
   PPMC880>

PPMC880>gevEdit mot-boot-path
   (Blank line terminates input.)
   /dev/scsi0/hdisk0\1\\boot\os.bin<cr>
   <cr>
   PPMC880>
```

In the above example MOTLoad downloads the file to the user download area by default. The execution address offset is 0, also by default. The boot file is located on device **/dev/scsi0/hdisk0**, in partition 1 under the /boot directory, and the file's name is **os.bin**.

This could also have been done by specifying the GEVs as follows:

```
mot-script-boot : diskBoot -f/dev/scsi0/hdisk0\1\\boot\os.bin

mot-boot-path : <leave undefined>
```

Here is an example of Auto Booting across a network using the TFTP protocol:

```
PPMC880>gevEdit mot-script-boot
   (Blank line terminates input.)
   netBoot -d/dev/enet0 -a0x04000000<cr>
   <cr>
   PPMC880>

PPMC880>gevEdit mot-/dev/enet0-cipa
   (Blank line terminates input.)
   144.191.27.143<cr>
   <cr>
   PPMC880>

PPMC880>gevEdit mot-/dev/enet0-sipa
   (Blank line terminates input.)
   144.191.13.33<cr>
   <cr>
   PPMC880>

PPMC880>gevEdit mot-/dev/enet0-file
   (Blank line terminates input.)
   /tftpBoot/bootFile.rom<cr>
   <cr>
   PPMC880>
```

In this example MOTLoad downloads the file from device “enet0” to the location in memory at 0x04000000. The IP address of enet0 (client) is 144.191.27.143; the IP address of the server is 144.191.13.33. The execution address offset is 0 by default. The boot file is located in the **/tftpBoot** directory and the boot file name is **bootFile.rom**.

In both examples above, Auto Boot is initiated on the next reset or power cycle of the board.

To disable Auto Boot, use **gevEdit** or **gevDelete** to modify *mot-script-boot* appropriately.

Status Indicators

There are two LEDs (DS1, DS2) located on the secondary side of the PrPMC880. LED1 is for user-defined purposes and LED2 is for BOARD FAIL (BF). The LED illuminates yellow when the BD_FAIL signal is active (software controlled). Refer to the Chapter 7, *Memory Map and Programming Details* for details of the BD_FAIL function in *Board LED Control Register*.

Debug Port

A debug port, to support module debug, is available at J1, a 16-pin header connector located on the primary side of the PrPMC880. This header provides the interface to the debug serial RS-232 port and an interface to the MPC7447 processor JTAG/COP port. Refer to Table 4-8 on page 4-9 for pin definitions. The port is initialized as COM1, the default configuration is 9600 baud, N81, DTE, no flow control.

After power-up, the baud rate of the debug port can be reconfigured by using the **portSet** command. Refer to the *Commands* chapter in the *MOTLoad Firmware Package User's Manual* for information about setting the communication mode.

Functional Description 3

Introduction

This chapter describes the PrPMC880 Processor PMC on a feature and block diagram level. The *PrPMC880 Description* on page 1-1 provides an overview of the PrPMC880. Figure 3-1 on page 3-3 shows a block diagram of the overall board architecture.

The following sections contain detailed descriptions of several blocks of circuitry. Programmable registers in the processor and peripheral chips can be found in Chapter 7, *Memory Map and Programming Details* or in the data sheet specific to the device as listed in Appendix C, *Related Documentation*.

Features

The following table summarizes the features of the PrPMC880 processor module.

Table 3-1. PrPMC880 Features

Feature	Description
Processor	Single MPC7447 Core frequency up to 1 GHz Bus Clock Frequency up to 133 MHz Address and data bus parity
L2 Cache	512KB located in processor
Flash	Channel 0 (Bank A) 64MB (8-bit) on-board (two 32MB devices)
System Memory	Up to 512MB (with 9 x512 Mbit DDR SDRAM and future 1GB with 9 x1 Gbit) 133 MHz (DDR333) DDR SDRAM onboard memory Double-Bit-Error detect, Single-Bit-Error correct across 72 bits
Memory Controller	Discovery II SMC

Table 3-1. PrPMC880 Features (continued)

Feature	Description
PCI Host Bridge	Discovery II PHB
Interrupt Controller	Discovery II MPIC
Ethernet	Two 10/100/1000 BaseT channels in Discovery II PHB
DMA	Discovery II integrated four independent IDMA channels
Timers	Discovery II integrated four independent timers, plus watchdog timer
I^2C	Discovery II integrated I^2C port with full master support
I2O	Discovery II integrated I2O compliant messaging interface
PCI/PCI-X Interface	One PCI-X 133 bus Up to 66 MHz bus, Rev 2.2 compliant 3.3V universal signaling interface P11, P12, P13, P14 PMC connectors Address/data parity per PCI specification
Serial EEPROM	One 256-byte EEPROM for Serial Presence Detect (SPD) and 512 x 8 I^2C EEPROM for Vital Product Data (VPD) One 256KB x 8 I^2C EEPROM for User VPD
Serial I/O	One asynchronous serial port with RS-232 interface routed to P14 connector or front panel connector
Debug Support	One processor JTAG/COP interface, signals routed to on-board header (J1)
Form Factor	Single width, standard length PMC (74mm x 149 mm) with 14mm board-to-board stacking height Standard (3.5mm) without mezzanine, or tall (20mm max.)
Input Power Requirements	3.17V to 3.43V ± 4%

Block Diagram

The next figure shows a block diagram of the PrPMC880's overall architecture.

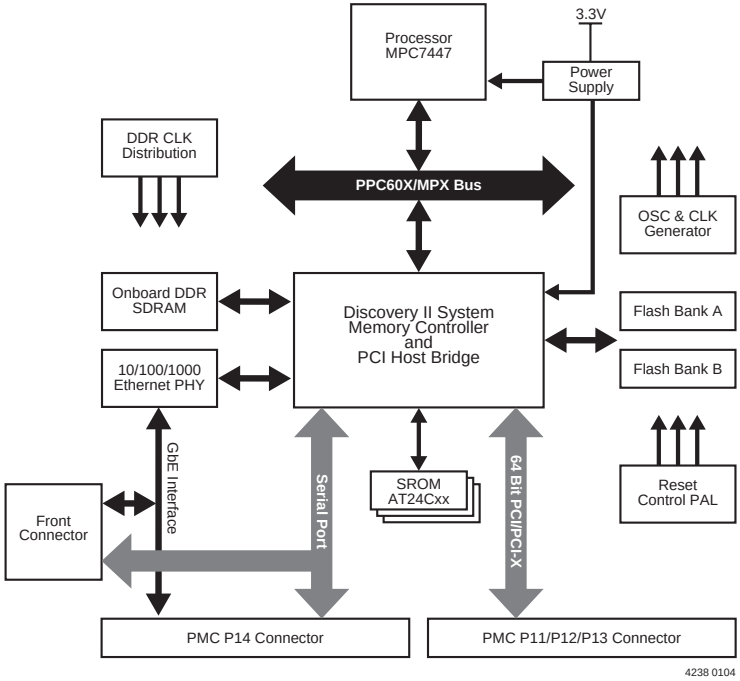


Figure 3-1. PrPMC880 Block Diagram

MPC7447 Processor

The PrPMC880 uses the MPC7447 processor, which is a high-performance, low-power 32-bit implementation of the PowerPC RISC architecture. The processor has full SMP capability, 36-bit physical address space, high bandwidth 133 MHz 64-bit MPX bus/60x bus, speeds up to 1 GHz with a core voltage of 1.3V, with integrated L1 and L2 cache.

Processor configurations are available in 1 GHz and other possible frequencies with memory bus speed capabilities of up to 133 MHz.

Cache

The MPC7447 processor has on-chip L1 cache of 32KB instruction cache and 16KB of data cache. The on-chip L2 cache is 512KB. There is parity checking on L1 and L2 cache arrays.

Discovery II System Controller

The Discovery II System Controller, MV64360, contains an integrated CPU bus interface, DDR SDRAM interface, 32-bit interface to devices, dual 64-bit PCI/PCI-X interfaces, integrated Gigabit Ethernet MACs, integrated 2Mb SRAM, interrupt controller, four general purpose timer/counters and a watchdog timer. These interfaces are connected via a high-performance crossbar switch architecture with any-to-any connectivity.

The following sections further describe the functions of the Discovery II System Controller.

PCI Host Bridge

The Discovery II System Controller contains an integrated PCI Host Bridge and Memory Controller which provides the bridge function between the internal 60x bus and the external PCI Local Bus. The MPC7447 supports a 32/64-bit PCI/PCI-X interface that is compliant with the PCI Local Bus Specification, Revision 2.2.

Onboard System Memory

The Discovery II System Controller supports one bank of DDR SDRAM memory using 256 Mbit, 512 Mbit, and up to 1 Gbit DDR SDRAM devices that support ECC. The memory devices can operate at a frequency of up to 266 MHz DDR with a clock frequency of 133 MHz. The DRAM has four transaction queues: two for write buffers and two for read buffers. There is full PowerPC cache coherency between the CPU L1/L2 caches and DRAM.

During system initialization, the firmware determines the presence and configuration of the onboard memory bank installed by reading the contents of the first 256 bytes of the 512 x 8 serial EEPROM device used for SPD memory configuration information. The system firmware then initializes the memory controller for proper operation based on the SPD information. Refer to Chapter 7, *Memory Map and Programming Details* for additional information and programming details.

Device Interfaces

The Discovery II System Controller supports up to five banks of devices, and the PrPMC880 uses three of the banks for interfacing to various devices. One is used for flash Bank A on the PrPMC880, and one for flash Bank B (not implemented). Each of these banks supports up to 512MB of address space that results in a total device space of 1.5GB. Bank A (onboard) consists of two, 32MB flash to make 32/64-bit system flash.

Table 3-2. Device Bank Assignment

Device Bank	Data Width	Function	Notes
0	32 bit	Bank A soldered flash	
1	32 bit	Bank B socketed flash	Not implemented
2	8 bit	I/O devices	
3	N/A	N/A	Not used
Boot	32 bit	Bank A soldered flash (32 bit)	Bank B if implemented, otherwise Bank A

PCI/PCI-X Interfaces

The Discovery II System Controller provides support for two 64-bit PCI/PCI-X busses that operate at a maximum frequency of 133 MHz when configured to PCI-X mode. When configured to conventional PCI mode, busses operate at 66 MHz. Only one of the two available buses is used on the PrPMC880. The Discovery II PCI interfaces are fully compliant to the PCI Local Bus Specification, Rev 2.2, and PCI-X addendum and contain

the required PCI configuration registers, which are accessible from either bus. Refer to Chapter 7, *Memory Map and Programming Details* for additional information and programming details.

The PrPMC880 module has four EIA-E700 AAAB connectors that provide a 64-bit PCI-X interface to an IEEE P1386.1 PMC-compliant host board.

- ❏ P11, P12, and P13 connectors provide the PCI-X interface (the PCI_0 signal is connected via the PMC connectors P11, P12, and P13; the PCI_1 signal is unused)
- ❏ P14 provides an I/O path from the PrPMC module to the host board for the I/O Gigabit Ethernet and serial signals

For further information on signaling, refer to *Special Function PMC Pins* on page 3-11.

Interrupt Controller

The PrPMC880 uses the interrupt controller which is integrated into the Discovery II to manage internal and external interrupts, which are routed to the Discovery II MPP pins from onboard resources. Currently, defined external interrupting devices include:

- ❏ Ethernet PHY interrupts
- ❏ Watchdog and timer interrupts
- ❏ Four PCI interrupts from host board (when PrPMC880 is the monarch)

MPP (Multi Purpose Pins) pins can be configured as general purpose I/O pins, as external interrupt inputs, or specific control/status pins for one of the system controller's internal devices. For more MPP pin information, refer Chapter 7, *Memory Map and Programming Details* in this manual.

Gigabit Ethernet

The Discovery II Ethernet interfaces provide support for two 10/100/1000 Mbps full duplex Ethernet ports, one of which (0) may be accessed via the front panel J3 connector or Channel 1 to the P14 connector, depending on

your board. The PrPMC880 supports ports configured to 10/100 Mbps MII interface and GMII 1 Gbps interface. Receive and transmit buffer management is based on a buffer-descriptor linked list. Descriptors and data transfers are performed by the port dedicated SDMA. Every board is assigned two Ethernet Station Addresses, with the higher address port assigned to port 0.

The Discovery II Ethernet interfaces connect to the BroadCom BCM5421S 10/100/1000BASE-T gigabit transceiver with GMII interfaces.

SRAM

The Discovery II System Controller integrates 2 Mbit (144-bit wide and 2KB deep) of general purpose SRAM accessible from the CPU or any of the other interfaces. The SRAM can be used as fast CPU access memory and for off-loading DRAM traffic. A typical use of the SRAM can be a descriptor RAM for the gigabit port.

General Purpose Timers/Counters

There are four 32-bit wide timers/counters on the Discovery II. Each timer/counter can be selected to operate as either a timer or a counter. The timing reference is based on the Discovery II Tclk input which is set at 133 MHz. Each counter/timer is capable of generating an interrupt.

Watchdog Timer

The Discovery II internal watchdog timer is a 32-bit countdown counter that can be used to generate a nonmaskable interrupt or to reset the system in the event of unpredictable software behavior. After the watchdog timer is enabled it becomes a free running counter that must be serviced periodically to keep it from expiring. To prevent the board from being reset, the WDT can be configured. It has two output pins: WDNMI# and WDE#.

- ❏ The WDNMI# is asserted after the timer is enabled and the 24-bit NMI_VAL count is reached. The WDNMI# pin is connected to

one of the Discovery II interrupt pins so an interrupt will be generated when the NMI_VAL count is reached.

- ❏ The WDE# pin is asserted after the watchdog timer is enabled and the 32-bit watchdog count expires. The Discovery II holds WDE# asserted for the duration of 16 system cycles after reset assertion. The pin is connected to the board reset logic so a board reset will be generated when WDE# is asserted.

DMA Controller

The PrPMC880 provides DMA capability through a four-channel DMA controller integrated into the Discovery II System Controller. Each DMA channel is capable of performing local memory-to-local memory, PCI memory-to-local memory, local memory-to-PCI memory, and PCI memory-to-PCI memory data transfers. The DMA channels can be accessed by the local CPU, as well as external PCI bus masters and support unaligned transfers, data chaining and scatter gather.

I^2C Interface

The Discovery II System Controller provides a two-wire serial interface via an integrated master/slave capable I^2C serial controller. The PrPMC880 contains these I^2C serial devices:

- ❏ One 256-byte EEPROM for SPD
- ❏ 8KB EEPROM for VPD
- ❏ 8KB EEPROM for user VPD

The 8KB EEPROMs use 2-byte addressing to address the 8KB of the device. Refer to the *Two-Wire Serial Interface* on page 7-10 in this manual for further information and I^2C addressing.

I2O Message Unit

I2O-compliant messaging for the PrPMC880 is provided by an I2O messaging unit integrated into the Discovery II System Controller. The messaging unit includes hardware hooks for message transfers between

PCI devices and the CPU, including all of the registers required for implementing the I2O.

PCI Bus Arbitration

The Discovery II System Controller integrates two PCI arbiters, one for each PCI interface. Each arbiter handles up to six external agents plus one internal agent. In accordance with VITA32 specifications the Discovery II PCI arbiters are disabled on the PrPMC880. Refer to Chapter 7, *Memory Map and Programming Details*

Sources of Reset

There are several potential sources of reset on the PrPMC880.

- ❏ Power-on reset
- ❏ PCIRST# (P12:13)
- ❏ COP_TRST# (J1)
- ❏ Board reset register bit 0
- ❏ Discovery II watchdog timer

The onboard reset control logic is located on the PLD. Each source of reset will result in a reset of the processor and all other onboard logic. Reset is provided to the host board via the RSTOUT# signal (P12:60) to reset the system. The RSTOUT# signal is independent of the PMC PCI reset.

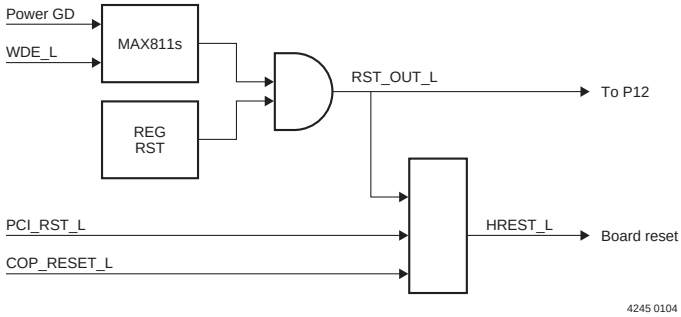


Figure 3-2. Reset Diagram

EEPROMs

The PrPMC880 module contains three +3.3V, serial EEPROM devices. The first is a 256-byte device for Serial Presence Detect (SPD) memory configuration information, the second is a 512 x 8 device that contains the Vital Product Data (VPD) storage of the module hardware configuration, and the third is a 256 x 8 device that provides storage of user VPD. The EEPROMs are accessed through the Discovery II's I^2C port. See *I2C Interface* for more information.

Flash

The PrPMC880 has one bank of onboard flash memory accessed via the integrated memory controller contained within the Discovery II System Controller. Bank A flash memory has two Intel Strata flash devices that are configured to operate in 16-bit mode, forming a 32-bit flash port providing 64MB capacity.

The supported flash type and size for Bank A is AMD part number 28F256K3, flash bank size of 64MB, device size of 256 Mbit, with a 28F256K3-8803 device ID.

Serial Port

The PrPMC880 has one asynchronous serial port with RS-232 interface routed to P14 connector or front panel connector (J3). The board is designed to use only one of these ports, they cannot be used together. The front serial port (J3) may be used for connecting a terminal to the PrPMC880 to serve as the firmware console. Use the PRPMC-CABLE-005 adapter cable. Contact your local Motorola Sales Office or Distributor for more information on the cable or to order. Refer to *Connecting to a Console* on page 1-13 for information on how this port is configured.

PCI-X Signaling Voltage Level

The PrPMC880 is a universal PMC module which operates with +3.3V or +5V signaling levels depending on the VIO voltage from the host board.

Special Function PMC Pins

PRESENT# Signal

The PRESENT# signal on the PrPMC880 module is grounded to indicate to the carrier board that the module is installed.

INTA#-INTD# Signals

The four PCI interrupt signals are routed to the MPIC external interrupt inputs so they can be monitored by the MPC7447 when the PrPMC880 is operating in the monarch mode. In non-monarch mode, the PrPMC880 can generate an interrupt to the host board processor by activating the INTA# interrupt output. Refer to the interrupt section of Chapter 7, *Memory Map and Programming Details* for interrupt assignments.

IDSELB#, REQB# and GNTB# Signals

These signals are not used.

M66EN#/PCIXCAP Signal

The PrPMC880 module is designed to operate on 66/133 MHz PCI-X mode, or 33/66 MHz traditional PCI bus, depending on the combination of the M66EN and PCIXCAP pin routed to the host board. The host board will sense the PCIXCAP signal to determine whether the add on module is capable of working on PCI-X mode, then drive the correct command pattern during PCI reset phase. When in the PCI-X mode, the PrPMC880 recognizes the mode via sensing the command pattern during

PCI_RESET#. The PrPMC880 utilizes M66EN# to signal and determine the bus frequency when operating in traditional PCI mode.

EREADY# Signal

The Processor PMC PCI bus Enumeration Ready (EREADY#) signal is connected to the on-board control logic. This pin is asserted low at power up and must be deasserted by software control. Refer to *PCI ENUM Control Register* on page 7-17.

MONARCH#

The MONARCH# input signal allows the host board to enable the monarch features on the PrPMC880. The PrPMC880 will pullup the MONARCH# signal. If the host board grounds this pin, the PrPMC880 operates as a monarch and provides PCI interrupt handling. If the host leaves MONARCH# floating, the PrPMC880 operates as a non-monarch module with no PCI interrupt handling or bus enumeration.

NVRAM and Real Time Clock

NVRAM and the Real Time Clock is contained in the MAX6900 chip. It has a fixed I^2C of $A0 which conflicts with the SPD memory, a special logic on CPLU (U96) is designed for I^2C interface to the chip. Refer to *Second I2C Control Register* on page 7-18 for information on generating the I^2C bus timing.

Real Time Clock

The MAX6900 device contains a real-time clock/calendar and 31-byte, 8-bit wide of Static Random Access Memory (SRAM). The real-time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year up to the year 2100. The clock operates in either the 24hr or 12hr format with an

AM/PM indicator. A super capacitor provides up to 12 hours of backup power the RTC.

SRAM

The Static RAM is 31 bytes addressed consecutively in the RAM address space. Even Address/Commands (C0h–FCh) are used for Writes, and odd Address/ Commands (C1h–FDh) are used for Reads. The contents of the RAM are static and remain valid for VCC down to 2V.

System Clock Generator

The clock generator uses an MPC9352 PLL clock driver to provide the clocks for the processor, the Discovery II Host Bridge, and other devices. All clocks are referenced to the 66 MHz onboard oscillator.

Table 3-3. Clock Signal Designations

Clock Name	Voltage Level (V)	Frequency (MHz)	Signal Destination
CPU_SYSCLK	2.5	133	MPC7447/7447
MV64360_SYSCLK	3.3	133	MV64360
PAL_CLK	3.3	133	PAL
CLK66M	3.3	66.66	MPC9352
25M_REFCLK	3.3	25	PHYs
DDR_CLK[8:0]	2.5	133 to 200	DDR memory devices
DDR_CLK_L[8:0]	2.3	133 to 200	DDR memory devices

DDR SDRAM Clock Generator

The PrPMC880 uses DDR SDRAM as system memory which requires a differential clock. There are nine DDR clocks and each one is routed to their respective DDR chip. All clocks are referenced to the output of the differential DDR clock from the Discovery II System Controller.

DDR SDRAM Power Supply

The PrPMC880 module uses the Linear Technology LT1702 Switching Regulator to generate the DDR SDRAM supply voltage from the +3.3V input. The LT1702, along with some outside MOS FETs and inductors, generate a +2.5V supply to the DDR SDRAMs.

Processor Core Power Supply

The PrPMC880 module uses the Linear Technology LT1702 Switching Regulator to generate the processor core voltage from the +3.3V input. The LT1702, along with outside MOS FETs and inductors, generate a +1.3V supply to the processor. The core power supply circuit can supply up to 16 Amps of output current.

Processor I/O Power Supply

The PrPMC880 module uses the Linear Technology LTC1702 Switching Regulator to generate the processor IO voltage from the +3.3V input. The LTC1702, along with some outside MOS inductors and filter capacitors, generate a +2.5V supply to the processor IO module and Discovery II host processor bus module.

Connector Pin Assignments

4

This chapter provides connector pin assignments for all connectors on the PrPMC880 board.

PCI Mezzanine Card (PMC) Connectors

There are four 64-pin EIA E700 AAAB SMT connectors (P11, P12, P13, P14) on the PrPMC880 to provide the 64-bit PCI/PCI-X interface to the host board. The P14 connector provides an interface to the serial I/O and Ethernet ports. The pin assignments are as follows.

Table 4-1. P11 - PMC Connector Pin Assignments

Pin	Signal	Signal	Pin
1	PMC_TCK	Not Used (-12V)	2
3	GND	PMCINTA_#	4
5	PMCINTB_#	PMCINTC_#	6
7	PRESENT#	+5Vdc	8
9	PMCINTD_#	Not Used (PCI RSVD)	10
11	GND	+3.3V aux	12
13	PMC_CLK	GND	14
15	GND	PMC_GNT_#	16
17	PMC_REQ_#	+5Vdc	18
19	VIO	AD31	20
21	AD28	AD27	22
23	AD25	GND	24
25	GND	C/BE3_#	26
27	AD22	AD21	28

Table 4-1. P11 - PMC Connector Pin Assignments (continued)

29	AD19	+5Vdc	30
31	VIO	AD17	32
33	FRAME_#	GND	34
35	GND	IRDY_#	36
37	DEVSEL_#	+5Vdc	38
39	PCIXCAP	LOCK_#	40
41	Not Used (RSVD)	RSVD	42
43	PAR	GND	44
45	VIO	AD15	46
47	AD12	AD11	48
49	AD09	+5Vdc	50
51	GND	C/BE0_#	52
53	AD06	AD05	54
55	AD04	GND	56
57	VIO	AD03	58
59	AD02	AD01	60
61	AD00	+5Vdc	62
63	GND	REQ64_#	64

Table 4-2. P12 - PMC Connector Pin Assignments

Pin	Signal	Signal	Pin
1	+12Vdc	TRST_#	2
3	TMS	TDO	4
5	TDI	GND	6
7	GND	RSVD (not used)	8
9	RSVD (not used)	RSVD (not used)	10

Table 4-2. P12 - PMC Connector Pin Assignments (continued)

11	RSVD	+3.3Vdc	12
13	PCIRST_#	Pulldown	14
15	+3.3Vdc	Pulldown	16
17	PME# (not used)	GND	18
19	AD30	AD29	20
21	GND	AD26	22
23	AD24	+3.3Vdc	24
25	IDSEL	AD23	26
27	+3.3Vdc	AD20	28
29	AD18	GND	30
31	AD16	C/BE2_#	32
33	GND	RSVD (not used)	34
35	TRDY_#	+3.3Vdc	36
37	GND	STOP_#	38
39	PERR_#	GND	40
41	+3.3Vdc	SERR_#	42
43	C/BE1_#	GND	44
45	AD14	AD13	46
47	M66EN	AD10	48
49	AD08	+3.3Vdc	50
51	AD07	RSVD (not used)	52
53	+3.3Vdc	RSVD (not used)	54
55	RSVD (not used)	GND	56

Table 4-2. P12 - PMC Connector Pin Assignments (continued)

57	RSVD (not used)	RSVD (not used)	58
59	GND	RSTOUT_#	60
61	ACK64_#	+3.3Vdc	62
63	GND	MONARCH#	64

Table 4-3. P13 - PMC Connector Pin Assignments

Pin	Signal	Signal	Pin
1	RSVD (not used)	GND	2
3	GND	CBE7_#	4
5	CBE6_#	CBE5_#	6
7	CBE4_#	GND	8
9	VIO	PAR64	10
11	AD63	AD62	12
13	AD61	GND	14
15	GND	AD60	16
17	AD59	AD58	18
19	AD57	GND	20
21	VIO	AD56	22
23	AD55	AD54	24
25	AD53	GND	26
27	GND	AD52	28
29	AD51	AD50	30
31	AD49	GND	32
33	GND	AD48	34
35	AD47	AD46	36
37	AD45	GND	38

Table 4-3. P13 - PMC Connector Pin Assignments (continued)

39	VIO	AD44	40
41	AD43	AD42	42
43	AD41	GND	44
45	GND	AD40	46
47	AD39	AD38	48
49	AD37	GND	50
51	GND	AD36	52
53	AD35	AD34	54
55	AD33	GND	56
57	VIO	AD32	58
59	RSVD (not used)	RSVD	60
61	RSVD	GND	62
63	GND	RSVD (not used)	64

Table 4-4. P14 - PMC Connector Pin Assignments

Pin	Signal	Signal	Pin
1	LPa_DA+	LPa_DC+	2
3	LPa_DA-	LPa_DC-	4
5	GND	GND	6
7	LPa_DB+	LPa_DD+	8
9	LPa_DB-	LPa_DD-	10
11	GND	GND	12
13	LPb_DA+	LPb_DC+	14
15	LPb_DA-	LPb_DC-	16
17	GND	GND	18
19	LPb_DB+	LPb_DD+	20
21	LPb_DB-	Lb_DD-	22
23	GND	GND	24

Table 4-4. P14 - PMC Connector Pin Assignments (continued)

25	NC	NC	26
27	NC	NC	28
29	NC	NC	30
31	NC	NC	32
33	NC	NC	34
35	NC	NC	36
37	NC	NC	38
39	NC	NC	40
41	NC	NC	42
43	NC	NC	44
45	NC	NC	46
47	NC	NC	48
49	NC	NC	50
51	NC	NC	52
53	NC	NC	54
55	NC	NC	56
57	GND	NC	58
59	RXD	RTS	60
61	TXD	GND	62

Gigabit Ethernet Connector

On some PrPMC880 models, an AMP 15-pin INFOPORT Series II PC card (low profile) connector provides optional front panel access. An external RJ-45 adapter cable, *PRPMC-CABLE-005* on page 4-8, is required for a standard RJ-45 interface.

Table 4-5. J3 - Gigabit Ethernet Connector Pin Assignments

Pin #	Signal
1	TXD(COM)
2	RXD(COM)
3	GND
4	No Connect
5	E0_BI_DA+
6	E0_BI_DA-
7	E0_BI_DB+
8	E0_BI_DB-
9	E0_BI_DC+
10	E0_BI_DC-
11	E0_BI_DD+
12	E0_BI_DD-
13	No Connect
14	No Connect
15	No Connect

PRPMC-CABLE-005

This cable is required if the front panel Ethernet connector is ordered as a build option. The following table describes the pin assignments for each 8-pin RJ-45 connector (one labeled ENET and one labeled COM). The second table provides the pin assignments for the 15-pin connector that plugs into the front panel of the PrPMC880.

Table 4-6. RJ-45 Y-Cable Pin Assignments

Pin	GigE	COM	COM	GigE	Pin
1	DA+	NC	NC	DA-	2
3	DB+	NC	TXD	DB-	4
5	DC+	RXD	GND	DC-	6
7	DD+	NC	NC	DD-	8

Table 4-7. Front Panel Connector Pin Assignments

Pin #	Signal	Signal	Pin #
1	TXD(COM)	RXD(COM)	2
3	GND	No Connect	4
5	DA+	DA-	6
7	DB+	DB-	8
9	DC+	DC-	10
11	DD+	DD-	12
13	No Connect	No Connect	14
15	No Connect		16

Debug Header

The debug and JTAG router are located on primary side of the PrPMC880. These headers provide the interface to the async serial port and the processor JTAG/COP port. The pin assignments for these headers are as follows.

Table 4-8. J1- Debug Header Pin Assignments

Pin	Signal	Signal	Pin
1	COP_TDO	COP_QACK_#3	2
3	COP_TDI	COP_TRST_#	4
5	RSVE (Pullup)	COP_VDD	6
7	COP_TCK	COP_CHKSTPI_#	8
9	COP_TMS	No connect	10
11	COP_SRST_#	GND	12
13	COP_HRST_#	No connect	14
15	COP_CHKSTPO_#	GND	16

MOTLoad Overview

5

MOTLoad is a PowerPC firmware package developed for Motorola's single board computers. MOTLoad is controlled through an easy to use, UNIX-like, command line interface. Its format was designed with the application-oriented needs of the end user. Consequently, the MOTLoad software package is similar to that of many end user applications designed for the embedded market. Functionally, this design allows MOTLoad to detect typical system level product devices.

The main purpose of the MOTLoad firmware package is to serve as a board power-up and initialization package, and to serve as a way to boot user applications. Although MOTLoad was not specifically designed as a diagnostics application, the test suites and the individual tests (with their various options) provide a significant amount of information that can be used for debug and diagnostic purposes. To use the MOTLoad firmware package successfully, you should have some familiarity with MCG products and firmware methodology.

For more detail on using MOTLoad and a listing of all commands, refer to the *MOTLoad Firmware Package User's Manual*, listed in Appendix C, *Related Documentation*.

Command Line Interface

The MOTLoad command line interface is similar to a UNIX command line shell interface. Commands are initiated by entering a valid MOTLoad command (a text string) at the MOTLoad command line prompt and pressing the Return key to signify the end of input. MOTLoad then performs the specified action. The MOTLoad command line prompt that appears in this chapter is generic, the exact command prompt designation is determined by the product being purchased, for example, MOTLoad or PrPMC880).

Example: `MOTLoad>`

If an invalid MOTLoad command is entered at the MOTLoad command line prompt, MOTLoad displays a message that the command was not found.

Example:

```
MOTLoad> mytest
“mytest” not found
MOTLoad>
```


If the you enter a partial MOTLoad command string that can be resolved to a unique valid MOTLoad command and press the Return key, the command will be executed as if the entire command string had been entered. This feature is an input shortcut that minimizes the required amount of command line input.

Example:

```
MOTLoad> version
Copyright: Motorola Inc. 1999-2003, All Rights Reserved
MOTLoad RTOS Version 2.0
PAL Version 1.1 RM01
Mon Mar 10 12:01:28 MST 2003
```

Example:

```
MOTLoad> ver
Copyright: Motorola Inc. 1999-2003, All Rights Reserved
MOTLoad RTOS Version 2.0
PAL Version 1.1 RM01
Mon Mar 10 12:01:28 MST 2003
```

If the partial command string cannot be resolved to a single unique command, MOTLoad informs the user that the command was ambiguous.

Example:

```
MOTLoad> te
“te" ambiguous
MOTLoad>
```

Command Line Help

Each MOTLoad firmware package has an extensive, product-specific, help facility that can be accessed through the **help** command. To display a complete listing of all available tests and utilities, enter **help** at the command line.

Example:

```
MOTLoad> help
```

For help with a specific test or utility, enter: **help** <**command_name**> at the prompt.

Example:

```
MOTLoad> help testRam
Usage: testRam [-aPh] [-bPh] [-iPd] [-nPh] [-tPd] [-v]
Description: RAM Test Directory
Argument/Option Description
-a Ph: Address to Start (Default = Dynamic Allocation)
-b Ph: Block Size (Default = 16MB)
-i Pd: Iterations (Default = 1)
-n Ph: Number of Bytes (Default = 1MB)
-t Ph: Time Delay Between Blocks in OS Ticks (Default = 1)
-v 0 : Verbose Output
MOTLoad>
```

Command Line Rules

There are a few things to remember when entering a MOTLoad command:

- ❏ Multiple commands are permitted on a single command line, provided they are separated by a single semicolon(“;”)
- ❏ Spaces separate the various fields on the command line (command/arguments/options)
- ❏ The argument/option identifier character is always preceded by a hyphen (“-”) character
- ❏ Options are identified by a single character

- ❏ Option arguments immediately follow (no spaces) the option
- ❏ All commands, command options, device tree strings, etc., are case sensitive

Example:

```
MOTLoad> flashProgram -d/dev/flash0 -n00100000
```


Command History Buffer

MOTLoad saves command line inputs into a command history buffer. Up to 128 previously entered commands can be recalled, edited, and reentered at the command line. Once the desired command appears on the command line it can be reexecuted by pressing the Return key.

Command Recall Mode

MOTLoad supports a command recall through the ESC and the **j** and **k** keys. Typing ESC and then **k** moves backwards through the history command buffer and displays the preceding commands. Typing ESC and then **j** moves forward through the history command buffer and displays the more recent commands. After the ESC key is pressed the **j** and/or **k** key may be pressed as often as needed to bring up the desired command from the command history buffer. For a complete listing of commands, refer to the *MOTLoad Firmware Packager User's Manual* listed in Appendix C, *Related Documentation*.

Command Line Execution Modes

MOTLoad utilities such as help always executes in the foreground. MOTLoad tests can be executed in the foreground (sequentially) or in the background (concurrently) as background tasks.

Note Not all tests can execute in background mode. As an example, cache tests must run in the foreground.

When a sequential test starts executing in the foreground, no new MOTLoad tests can execute until the current test running in the foreground is complete. This does not apply to background tests.

Example:

```
MOTLoad> testRam
```

In concurrent test mode, each test gets a time sliced share of the CPU execution time. The amount of user control over the background task time slicing operations is determined by the underlying OS. The operator specifies concurrent test execution by ending the test command line with the ampersand "&" character (prior to the Return). The MOTLoad command prompt reappears after a concurrent test is started.

Example:

```
MOTLoad> testRam &
```

After the MOTLoad prompt reappears another test or utility may be started (in the foreground or background execution mode) as long as it does not interfere (use the same computer resources) with the operations of other test(s) running in background mode. The test execution status of a test running in background mode can be monitored through the use of the **taskActive** and **testStatus** commands. Refer to the *MOTLoad Firmware Package User's Manual* listed in Appendix C, *Related Documentation*.

Copying/Transferring MOTLoad Images

As an example, to copy flash, a MOTLoad image from soldered flash to socketed flash, or visa versa, you can perform one of the following steps depending upon the transfer direction.

To copy MOTLoad from socketed flash to soldered flash, at the MOTLoad prompt, enter:

MOTLoad> **MOTLoad (flash 1 -> 0)**

```
MOTLoad>
```

To copy MOTLoad from soldered flash to socketed flash:

MOTLoad> **MOTLoad (flash 0 -> 1)**

```
MOTLoad>
```

To move one part of a vxWorks image from one part of flash to another, perform the following:

MOTLoad> **flashProgram -d/dev/flash0 -o00000000 -sf3f00000 -v**

Modifying the Environment 6

Global Environment Variables

Global Environment Variables (GEVs) are used to store nearly any value for later retrieval, even after loss of power or hardware reset. Each value saved needs a unique label, the label being defined at the same time as the value. GEVs are typically stored in NVRAM. MOTLoad requires 8Kb at the top end of flash bank 0 for use as NVRAM. The amount of space set aside in the NVRAM for storage of GEVs is 3592 bytes. The flash offset of 0x100000 is used as the GEV address.

GEV Command List

These configuration and environment variables are persistent through a power loss. The choice for storing these variable is in NVRAM or flash memory. The following commands permit the manipulation of GEVs.

Table 6-1. GEV Commands

Command	Description
gevDelete	Global Environment Variable Delete
gevDump	Global Environment Variable(s) Dump (NVRAM Header + Data)
gevEdit	Global Environment Variable Edit
gevInit	Global Environment Variable Area Initialize (NVRAM Header)
gevShow	Global Environment Variable Show

Reserved GEVs

The MOTLoad firmware reserves several GEV names for the invoking of special features. MOTLoad's approach to configuration and environment variables storage is simple. There are no predefined locations (within the storage area) for each of the possible variables. Each variable is defined by an identifier string. All variables are basically ASCII strings terminated by a null character. This format of ASCII null terminated strings was utilized by PReP (PowerPC Reference Platform) based computer systems. The name of these parameters are Global Environment Variables (GEV). Please refer to the *MOTLoad Firmware Package User's Manual* Appendix C, *Related Documentation* for more information. Below is a list of reserved GEVs.

Table 6-2. Startup GEVs

Command	Description
Startup (Boot) Script Contents	
mot-script-boot:	Startup (Boot) Script Contents (Commands/Tests). This GEV is basically a script which is executed upon start-up. The contents of this script is any combination of commands and/or tests which can be executed from the command line. This script allows you to automate the process of testing and booting.
Startup (Boot) Script Delay Value	
mot-script-delay:	Startup (Boot) Script Delay Value (in seconds). The value associated with this GEV is the time in seconds the boot process will wait for the you to have the opportunity to cancel the Startup Script. If this GEV is not defined the default wait time is 7 seconds. This GEV is only used if the mot-script-boot has been defined.

Table 6-3. Network GEVs, TFTP and BOOTP

Variable	Description
mot-boot-cipa	Client IP Address (Decimal Dot Notation)
mot-boot-sipa	Server IP Address (Decimal Dot Notation)
mot-boot-gipa	Gateway IP Address (Decimal Dot Notation)
mot-boot-snma	Subnet IP Address Mask (Decimal Dot Notation)
mot-boot-file	Name of File that was Loaded
mot-boot-device	Name of Device (Interface Node)

The following GEV variables can be used in substitution of the command line options on network commands. The */dev/enet0* portion of the variable may be any network interface that is present in the system. The presence of the device node shows MOTLoad support, for example, the associated driver is packaged with the executable binary and has been initialized/instantiated. If **netBoot** was used to boot the board and the update flag (-*u*) has been specified on the command line, the following GEVs will be updated

Table 6-4. Alternate GEV Variables

Variable	Description
mot-/dev/enet0-cipa	Client IP Address (Decimal Dot Notation)
mot-/dev/enet0-sipa	Server IP Address (Decimal Dot Notation)
mot-/dev/enet0-gipa	Gateway IP Address (Decimal Dot Notation)
mot-/dev/enet0-bipa	Broadcast IP Address (Decimal Dot Notation)
mot-/dev/enet0-snma	Subnet IP Address Mask (Decimal Dot Notation)
mot-/dev/enet0-file	Name of File to Load (Get)

Example: `MOTLoad>`**tftpGet -d/dev/*enet0***,

This command will use all GEV variables for command line options and if the GEV is not defined, the standard defaults are used.

Upon a successful network boot (load), the following GEVs are updated to reflect the boot parameters of this boot instance:

Table 6-5. Disk Boot GEVs

Variable	Description
mot-boot-path	This GEV may specify multiple boot paths. A path consists of a device name, a partition number, and a file name. For some disk boot media formats, the partition number and file name are not required.This would be the case for PReP formatted boot media. When specifying multiple boot paths, a colon (:) character must be used to separate the individual boot paths. For example: **/dev/fd0\1\\boot.bin:/dev/ide0/hdisk0\1\\boot\os.bin**
Boot Results GEV	
mot-boot-device	This is updated with the name of the boot path in which a successful boot (load) was accomplished. For example: **dev/ide0/hdisk0\1\\boot\os.bin**

Table 6-6. SCSI GEVs

Variable	Description
mot-/dev/scsi0-id	SCSI Identifier of Host (0 to 15) Default is 7 for 8-bit SCSI interfaces and 15 for 16-bit capable SCSI interfaces. The */dev/scsi0* portion of the variable may be any SCSI interface present in the system.

Table 6-7. GEV Test Suites

Test Suite	Description
xxxxxxxxx	Users may save a test suite directly to NVRAM. Invoking the saved test suite is a two-step process. The first step is to retrieve it from NVRAM and the second step is the invocation. For example: MOTLoad> **testSuiteMake -ngevTestSuite -r** MOTLoad> **testSuite -ngevTestSuite** The first step only needs to be done once (for example, for each instance of MOTLoad - reset, power up)

How to Create a Configurable POST

Each time startup occurs the POST (Power On Self Test) commands are displayed. All commands are run in the background. To create a POST, follow these steps.

1. Define the POST using a GEV.

   `MOTLoad>` **gevEdit -nPOST**

   Test1 for POST
   Test2 for POST
   Test3 for POST

2. Define the mot-script-boot GEV.

   `MOTLoad>` **gevEdit -n*mot-script-boot***

   testSuiteMake -nPOST -r (*this creates a test suite from what is stored in NVRAM*)
   testSuite -nPOST -r (this runs the test suite in the background).

Enter the **testStatus** command to find out if POST passed.

Viewing GEV Values

All GEVs currently stored in NVRAM may be viewed with the **gevShow** command. The order of the GEVs will be the order in which they were created. Each GEV will be shown as *label=value*. If the value is comprised of more than one line of data, the label will be shown on a separate line, above the value line(s).

```
gevShow
example1=Hi 12345 Hi
example2=Come Back Soon
jazz=
a
b
c
e
d
g
e
t
lkjkj
jsjs
ieie
vnvnv
s’s’s’s
c

apple=apple GEV
jazz3=short jazz3
example3=August 7, 2002
Total Number of GE Variables =6, Bytes Utilized =160, Bytes
Free =3432
```

Viewing GEV Labels

The labels of all currently defined GEVs can be listed with the **gevList** command. The order of the GEVs are in the order in which they were created:

```
gevList
example1
example2
jazz
apple
jazz3
example3
Total Number of GE Variables =6, Bytes Utilized =160, Bytes
Free = 3432
```


Creating GEVs

The **gevEdit** command is used to create a new GEV. Execute **gevEdit**, and provide a label name which is currently not used, as in this example of a GEV labeled “example3” with a value of “August 7, 2002":

```
gevEdit example3
(Blank line terminates input.)
August 7, 2002

Update Global Environment Area of NVRAM (Y/N)? y
```

GEV labels can be up to 255 bytes long. The label itself is stored in NVRAM along with the GEV value. Therefore, as GEV space is limited, users are encouraged to select labels of appropriate length. GEV values are stored as ASCII strings, which may be up to 511 bytes long. GEV labels and values are both case sensitive.

If there is not enough space remaining for storage of the new GEV, a message similar to the following is displayed:

```
Not all variables were copied, 1 remaining
```

The newly added variable is not added, even if the `Update Global Environment Area of NVRAM (Y/N)?` question is answered with a **Y**.

Editing GEVs

The **gevEdit** command is used to modify the value of an existing GEV. Simply execute **gevEdit**, and provide the label of the GEV to be modified, as:


```
gevEdit example2
example2=goodbye 54321 goodbye
(Blank line terminates input.)
Come Back Soon.

Update Global Environment Area of NVRAM (Y/N)? y
```

Entering a **y** or **Y** replaces the original GEV value with the new. Any other answer will preserve the original GEV.

Deleting GEVs

To remove a GEV from NVRAM, use the **gevDelete** command and provide the GEV label, as:

```
gevDelete jazz2
jazz2=
jsjsjs
sjjsjs
eieieie
82828282
xxxxx

Update Global Environment Area of NVRAM (Y/N)?
```

Entering a **y** or **Y** deletes the GEV label and value. Any other answer will preserve the GEV.

When a GEV is deleted its label can be reused. Also, the NVRAM space which was used to store both the deleted label and value is made available by the deletion.

Initializing the GEV Storage Area

The **gevInit** command is used to initialize the GEV area of the NVRAM device. Execution of this command deletes all currently defined GEVs and prepares the GEV area for its first variable. This command should be used with caution, as reentry of all removed GEVs (as with gevEdit) can be time consuming.


```
HXEB100> gevInit

Initialize Global Environment Area of NVRAM
Warning: This will DELETE any existing Global Environment
Variables!
Continue? (Y/N)?
```

Entering a **y** or **Y** deletes all GEV labels and values. Any other answer will preserve the GEV area.

Memory Map and Programming Details

7

The PrPMC880 programming model is based on the PowerPlusIII. Where possible, the PowerPlusIII addressing assignments have been kept. A new programming model based on the Discovery II System Controller is utilized by extending the addressing range to include the MV64360 System Controller, which has 36-bit addressing, 64GB. The 36-bit addressing is implemented on the planar.

These memory maps assume a 4GB addressable range (32-bit). This is accomplished using the System Controller's Real Addressing Mode, where translation is disabled by clearing bits in the Machine State Register (MSR), MSR[IR] for instruction fetching, or MSR[DR] for data accesses. When address translation is disabled the physical address is identical to the effective address. Refer to the *PowerPC Programming Environment User's Manual* listed in Appendix C, *Related Documentation* for more information.

Memory Maps

The following section describes the memory maps for the PrPMC880 series modules.

Processor Memory Map

The Discovery II System Controller presents a default CPU memory map following a RESET negation. The next table shows the default memory map from the point of view of the processor. Address bits [35:32] are only

relevant for the MPC7445's extended address mode and are not shown in the next table.

Table 7-1. Default Processor Memory Map

Processor Address		Size	Definition
Start	End		
0x00000000	0x20000000	512MB	DRAM
0x80000000	0x9FFFFFFF	512MB	PCI Memory Space
0xF0000000	0xF07FFFFF	8MB	PCI I/O Space
0xF4000000	0xF7FFFFFF	64MB	Flash Bank A
0xFF800000	0xFF9FFFFF	2MB	Reserved

Example Processor Memory Map

The following table describes a suggested memory map from the point of view of the processor. The beginning of PCI memory space is determined by the end of DRAM, and is rounded up to the nearest 256MB boundary. For example, if memory was 1GB on the baseboard and 192MB on a mezzanine, the beginning of PCI memory would be rounded up to address 0x50000000 (1GB + 256MB). This is the memory map used by MOTLoad.

Table 7-2. Suggested Memory Map

Processor Address		Size	Definition	Note
Start	**End**			
00000000	top_dram-1	DRAM size	System memory (onboard DRAM)	
80000000	DFFFFFFF	1.5GB	PCI bus0 memory space	
E0000000	EFFFFFFF	0.25GB	Not assigned	
F0000000	F0700000	8MB	PCI bus0 I/O space	

Table 7-2. Suggested Memory Map (continued)

Processor Address		Size	Definition	Note
F0800000	FEFFFFFF	8MB	Not assigned	PCI bus1 not used
F1000000	F10FFFFF	1MB	MV64360 internal registers	
F1100000	F11FFFFF	1MB	DEV CS2*	Mainly for system register
F1200000	F3FFFFFF	46MB	Not assigned	
F4000000	F7FFFFFF	64MB	Device CS0* flash	
F8000000	FF7FFFFF	120MB	Not assigned	
FF800000	FFFFFFFF	8MB	Device boot bank flash	

MPC7447 System Bus Function

The next sections discuss the system bus functions implemented in the PrPMC880 module.

Processor

The maximum external processor bus speed is 133 MHz, parity checking is supported for the system address and data busses. The MPC7447 processor is configured to operate in the MPX bus mode.

Processor Type Identification

Software is able to distinguish between the MPC7447 and future MPC7447 processor by reading the Processor Version Register. The most significant 16 bits (0:15) of the MPC7445 Processor Version Register reads as 0x8002. Refer to the *MPC7450 RISC Processor Family User Guide* for more information.

Processor PLL Configuration

The processor internal clock frequency (core frequency) is a multiple of the system bus frequency. There are five configuration pins, PLL_CFG[0:4) for hardware strapping of the processor core frequency used to select between 2x and 16x of the system bus frequency.

Table 7-3. Processor PLL Description

CFG [0:4]	Example Bus-to-Core Frequency in MHz (VCO Frequency in MHz)									
	Bus-to-Core Multiplier	Core-to-VCO Multiplier	Bus (SYSCLK) Frequency							
			33.3 MHz	50 MHz	66.6 MHz	75 MHz	83 MHz	100 MHz	133 MHz	167 MHz
01000	2x	2x								
10000	3x	2x								500 (1000)
10100	4x	2x							532 (1064)	667 (1333)
10110	5x	2x						500 (1000)	667 (1333)	835 (1670)
10010	5.5x	2x						550 (1100)	733 (1466)	919 (1837)
11010	6x	2x						600 (1200)	800 (1600)	1002 (2004)
01010	6.5x	2x					540 (1080)	650 (1300)	866 (1730)	1086 (2171)
00100	7x	2x				525 (1050)	580 (1160)	700 (1400)	931 (1862)	1169 (2338)
00010	7.5x	2x			500	563	623	750	1000	1253

Table 7-3. Processor PLL Description (continued)

CFG [0:4]	Example Bus-to-Core Frequency in MHz (VCO Frequency in MHz)									
	Bus-to-Core Multiplier	Core-to-VCO Multiplier	Bus (SYSCLK) Frequency							
					(1000)	(1125)	(1245)	(1500)	(2000)	(2505)
11000	8x	2x			533	600	664	800	1064	
					(1066)	(1200)	(1328)	(1600)	(2128)	
01100	8.5x	2x			566	638	706	850	1131	
					(1132)	(1276)	(1412)	(1700)	(2261)	
01111	9x	2x			600	675	747	900	1197	
					(1200)	(1350)	(1494)	(1800)	(2394)	
01110	9.5x	2x			633	712	789	950	1264	
					(1266)	(1524)	(1578)	(1900)	(2528)	

CPU Bus Mode

The CPU bus operating mode (MPX) can be determined by reading the BMODE bites [16:17] in the processor's Memory Subsystem Control Register (MSCCR0). the power-up state of the BMODE (0:1) pins is captured in these register bits.

Table 7-4. MSSCR0 Field Descriptions

Bits	Name	Function
0–2	—	Reserved
3–5	DTQ	DTQ size. Determines the maximum number of outstanding data bus transactions that the MPC74xx can support. Refer to the *MPC7450 RISC Processor Family User Guide* for more information.

Table 7-4. MSSCR0 Field Descriptions (continued)

Bits	Name	Function
3-5		(continued) The DTQ bit values are as follows: 000 8 Entries 001 16 Entries 010 2 Entries 011 3 Entries 100 4 Entries 101 5 Entries 110 6 Entries 111 7 Entries
6	—	Reserved
7	EIDIS	Disable external intervention in MPX bus mode 0 External interventions occur. 1 The MPC74xx performs external pushes instead of external interventions. External interventions are disabled.
8–9	—	Reserved
10	L3TCEXT	L3 turn around clock count extension (MPC7457-Specific) 0 Used with MSSCR0[L3TC] to determine the L3 turnaround clock count. See L3CR[L3TC] field description. 1 Used with MSSCR0[L3TC] to determine the L3 turnaround clock count. See MSSCR0[L3TC] field description. **Note** The MSSCR0[10] bit is reserved on the MPC74xx and is used as an L3 turnaround clock count only on the MPC7457.
11	ABD	Address bus driven mode 0 Address bus driven mode disabled 1 Address bus driven mode enabled The read-only bit reflects the state of the BMODE0 signal after HRSET negation and indicates whether the processor is address bus driven mode. Refer to the *MPC7450 RISC Processor Family User Guide* for more information.

Table 7-4. MSSCR0 Field Descriptions (continued)

Bits	Name	Function
12	L3TCEN	L3 turnaround clock enable 0 L3 turnaround clock disabled 1 L3 turnaround clock is enabled. Refer to the *MPC7450 RISC Processor Family User Guide* for more information.
13–14	L3TC	L3 turnaround clock count. The following bit values determine the number of cycles the L3 waits between read and write transactions if L3TCEN is set.The following values are correct for the MPC74xx. Note that only for the MPC7457, the following values are correct when MSSCR0[L3TCEXT] = 0: 00 2 L3CKn cycles 01 3 L3CKn cycles 10 4 L3CKn cycles 11 5 L3CKn cycles Also note that only for the MPC7457, the following values are correct when MSSCR0[L3TCEXT] = 1. These values are not used on the MPC74xx. 00 6 L3CKn cycles 01 7 L3CKn cycles 10 8 L3CKn cycles 11 9 L3CKn cycles
15	—	Reserved.
16–17	BMODE	Bus mode (read-only). Reflects the inverse of the voltage levels on BMODE[0:1] while HRESET is asserted. Indicates whether the system interface uses the 60x or MPX bus protocol. 00 60x bus mode 01 Reserved 10 MPX bus mode 11 Reserved Note that the value on BMODE[0:1] after reset negates determines other values of MSSCR0 as follows: BMODE0 (post reset) MSSCR0[ABD] BMODE1 (post reset) MSSCR0[ID]

Table 7-4. MSSCR0 Field Descriptions (continued)

Bits	Name	Function
18–25	—	Reserved. Normally cleared, used in debug, writing nonzero values may cause boundedly undefined results.
26	ID	Processor identification. Sets the processor ID to either processor 0 or 1. Determined by the inverse of the voltage levels on BMODE1 while HRESET is negated. 0 BMODE1 negated after HRESET negated 1 BMODE1 asserted after HRESET negated In a multiprocessor system, one processor can be assigned by the BMODE1 as processor 0 and all other processor can be assigned as processor 1. Then software can find processor 0 and use it to reidentify the other processors by writing unique values to the PIR of the other CPUs.
27-29	—	Reserved. Read as zeroes.
30-31	L2PFE	L2 prefetching enabled. The following values determine the number of L2 prefetch engines enabled as follows: 00 L2 prefetching disabled, no prefetch engines 01One prefetch engine enabled 10 Two prefetch engines enabled 11 Three prefetch engines enabled These bits enable alternate sector prefetching in the 2-sectored L2 cache; up to 3 outstanding prefetch engines may be active.

Flash Memory

The PrPMC880 contains one bank of flash memory, which is accessed via the integrated memory controller contained within the Discovery II System Controller. Bank A is onboard and consists of two 3.3V devices that are configured to operate in 16-bit mode, forming a 32-bit flash port, and provides 64MB capacity.

Discovery II MPP Configuration

The Discovery II System Controller contains a 32-bit multipurpose port. The MPP pins can be configured as general purpose I/O pins, external interrupt inputs, or as a specific control/status pin for one of the system controller's internal devices. After a reset, all MPP pins default to GPP (General Purpose Pins) inputs. Software is used to configure each of the pins for the desired function. The next table defines the function assigned to each MPP on the PrPMC880.

Table 7-5. MPP Pin Functions

MPP Pin Number	Input/ Output	Function
GPP[0]	O	Sout for UART
GPP[1]	I	Sin for UART
GPP[2]	O	RTS for UART
GPP[3]	I	CTS for UART
GPP[4]	I	Unused
GPP[5]	I	Unused
GPP[6]	I	Unused
GPP[7]	I	BCM5421S PHY interrupts (ORed)
GPP[8:15]	I	Unused
GPP[16]	I	PCI_INTA#
GPP[17]	I	PCI_INTB#

Table 7-5. MPP Pin Functions (continued)

GPP[18]	I	PCI_INTC#
GPP[19]	I	PCI_INTD#
GPP[20:23]	I	Unused
GPP[24]	I	Unused
GPP[25]	O	Watchdog Timer Expired output (WDE#)
GPP[26]	O	Watchdog Timer NMI output (WDNMI#)
GPP[27:31]	I	Unused

Two-Wire Serial Interface

A two-wire serial interface for the PrPMC880 is provided by an I^2C compatible serial controller integrated into the Discovery II System Controller. The I^2C serial controller provides two basic functions:

- The MV64360 can be configured to automatically read data out of a serial EEPROM following a reset and initialize any number of internal registers.
- The second function is where the controller is used by the system software to read the contents of the VPD and SPD EEPROMs contained on the PrPMC880 to initialize the memory controller and other interfaces.

For additional details regarding the MV64360 two-wire serial controller operation, refer to the *MV64360 System Controller Data Sheet*, listed in

Manufacturers' Documents in Appendix C, *Related Documentation*. The following table shows the I^2C devices on the PrPMC880 and their assigned device IDs.

Table 7-6. I^2C Device IDs

Device Function	Size	Device Address	I2C Bus Address	Notes
Memory SPD (Bank 0 and 1)	256KBx8	000b	$A0	1, 3
Not Used	NA	001b	$A2	1
Not Used	NA	010b	$A4	2
Configuration VPD	8KBx8	100b	$A8	2
User VPD	8KBx8	101b	$AA	2

Notes

1. The SPD defines the physical attributes of each bank or group of banks. For instance, if both banks of a group are populated they will be the same speed and memory size.
2. This is a dual address serial EEPROM
3. This address also used by NVRAM/RT, refer to *NVRAM and Real Time Clock*.

Discovery II Reset Configuration

The Discovery II System Controller supports two methods of device initialization following reset:

- ❏ Pins sampled on the deassertion of reset
- ❏ Partial pin sample on deassertion of reset plus serial ROM initialization via the I^2C bus for user defined initialization

The PrPMC880 supports only pin sample options. States of the various pins on the device AD bus are sampled when reset is deasserted to

determine the desired operating modes. Combinations of pull-ups and pull-downs are used to set the options. There are several ways to set options:

- ❏ Some options are fixed
- ❏ Options selectable at build time using the proper pull-up/pull-down resistor
- ❏ Options driven by onboard PAL

While the PrPMC880 is designed to work in both monarch and non-monarch mode, the PCI clock and PCI-X/PCI work pattern are selected by the carrier board whether in monarch or non-monarch mode, so some parameters such as PCI PLL control are determined by a PCI reset pattern each time during PCI reset.

See the following table for descriptions of the configuration options. XXX or XX in the table means those value have no fixed power up values and must be determined by system configuration.

Table 7-7. Configuration Options for Discovery II Reset

Device AD Bus Signal	Select Option	Default Power-Up Setting	Description	State of Bits vs. Function	
AD[0]	Fixed	0	SROM Initiation	0	No SROM Initiation
				1	SROM Initiation Enabled
AD[1]	Resistor	1	DRAM Pads Calibration	0	Calibration Disabled
				1	Calibration Enabled
AD[3:2]	Resistor	10	SROM Device Address	00	1010000 ($A0)
				01	1010001 ($A2)
				10	1010010 ($A4)
				11	1010011 ($A6)
AD[4]	Fixed	1	Internal 60X bus Arbiter	0	Internal Arbiter Disabled

Table 7-7. Configuration Options for Discovery II Reset (continued)

Device AD Bus Signal	Select Option	Default Power-Up Setting	Description	State of Bits vs. Function	
				1	Internal Arbiter Enabled
AD[5]	Resistor	1	Internal Space Default Address	0	0x14000000
				1	0xf1000000
AD[7:6]	Jumpers	01	CPU Bus Configuration	00	60X bus
				01	MPX bus
				10	Reserved
				11	Reserved
AD[8]	Resistor	1	CPU Pads Calibration	0	Calibration Disabled
				1	Calibration Enabled
AD[9]	Fixed	0	Multi MV64360 Support	0	Not Supported
				1	Supported
AD[12]	Resistor	1	PCI0 pads Calibration	0	Calibration Disabled
				1	Calibration Enabled
AD[13]	Resistor	1	PCI1 pads Calibration	0	Calibration Disabled
				1	Calibration Enabled
AD[15:14]	Resistor	XX	Boot device Width	00	8 Bits
				01	16 Bits
				10	32 Bits
				11	Reserved

Table 7-7. Configuration Options for Discovery II Reset (continued)

Device AD Bus Signal	Select Option	Default Power-Up Setting	Description	State of Bits vs. Function	
AD[16]	Resistor	1	PCI Retry	0	Disabled
				1	Enabled
AD[17]	Fixed	1			Must pull High
AD[18]	Resistor	1	DRAM Clock Select	0	DRAM Runs At a Higher Frequency Than Core Clock
				1	DRAM Runs At a Same Clock Frequency as Core Clock
AD[19]	Resistor	0	DRAM Address/Control Delay	0	DRAM Address and Control Signal Toggled on Falling Edge of DRAM Clock
				1	DRAM Address and Control Signal Toggled on rising Edge of DRAM Clock
AD[20:21]	Resistor	11	DRAM Control Pipeline select	00	Reserved
				01	Two pipe stages
				10	Reserved
				11	Three pipe stages
AD[24:22]	Resistor	000	DRAM read path control	000100	DRAM run in sync mode
				001111	DRAM run in async mode
AD[25]	Fixed	0	Gigabit Port3 Enable	0	Disabled
				1	Enabled

Table 7-7. Configuration Options for Discovery II Reset (continued)

Device AD Bus Signal	Select Option	Default Power-Up Setting	Description	State of Bits vs. Function	
AD[28:26]	Resistor	XXX	PCI_1 DLL control	000	DLL disable
				001	Convention PCI mode at 66 MHz
				101	PCI-X mode at 133 MHz
				110	PCI-X mode at 66MHz
AD[31:29]	Resistor	XXX	PCI_0 DLL control	000	DLL disable
				001	Convention PCI mode at 66 MHz
				101	PCI-X mode at 133 MHz
				110	PCI-X mode at 66 MHz
TXD0[0]	Resistor	0	Gbit port 0 GMII/PCS select	0	GMII/MII
				1	PCS
TXD1[0]	Resistor	0	Gbit port 1 GMII/PCS select	0	GMII/MII
				1	PCS
WE[3:0]	Resistor	TBD	DRAM PLL N divider N[7:4]	M=2	TBD by MV64360 datasheet
DP[3:0]	Resistor	TBD	DRAM PLL N divider N[3:0]	N=3	TBD by MV64360 datasheet
BADR[0]	Resistor	1	DRAM PLL NP	1	Pull up NP
BADR[1]	Resistor	1	DRAM PLL HKVCO	1	Pull down HKVCO
BADR[2]	Resistor	1	DRAM PLL NP	0	PLL power down

Table 7-7. Configuration Options for Discovery II Reset (continued)

Device AD Bus Signal	Select Option	Default Power-Up Setting	Description	State of Bits vs. Function	
				1	PLL power up (normal operation)
TXD0(6:1)	TBD	TBD	DRAM PLL M divider		MV64360 datasheet
TXD0[7]	Resistor	0	JATAG Pad calibration bypass	0	Normal Operation
				1	bypass pad calibration
TXD1[1]	Resistor	0	Core PLL bypass	0	Normal operation
				1	Bypass the core's PLL
TXD1[4:2]	Resistors	000	Core PLL control	000	Tuning of the PLL clock tree

Discovery II Registers

Registers for the system controller are extensive and described in the MV64360 Data Sheet. Please refer to the data sheet and related documentation as listed in Appendix C, *Related Documentation*.

System Register

The PrPMC880 system registers include the Status, Reset, Enum, I^2C channel 2, and LED registers. These registers provide module control and status information.

Board Status Register

Address:F1100008h **Bit Definition**								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Definition	MONARCH_L	FAULT_ PWR	PHY1_ LINK2_L	PHY1_ LINK1_L	PHY1_ QUALITY_L	PHY0_ LINK2_L	PHY0_ LINK1_L	PHY0_ QUALITY_L)
R/W	R	R	R	R	R	R	R	R

PCI ENUM Control Register

Address:F1100007h **Bit Definition**								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Definition	0	0	0	0	0	0	EREADY1	EREADY0
R/W	R	R	R	R	R	R	R/W	R

Second I²C Control Register

This register provides simulation for the RTC I²C controller due to the SPD occupying theMV64360 I²C 0xA0 address.

Dev Bank 2			
I2CSDAin	I2C Data Input	R	
I2CDIR	I2C Direction	R/W	0: input 1: output
I2CSCL	I2C Clock	R/W	
I2CSDAO	I2C Data Output	R/W	

Address:F1100006h **Bit Definition**								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Definition	0	0	0	0	I2CSDAin	I2CDIR	I2CSCL	I2CSDAO
R/W	R	R	R	R	R	R/W	R/W	R/W

Board LED Control Register

Address:F1100005h LED0/1 **Bit Definition**								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Definition	0	0	0	0	0	0	LED2 BD_FAIL	LED1 User defined
R/W	R	R	R	R	R	R	R/W	R/W
Note							1=on	1=on

Interrupt Handling

The following sections further describe PrPMC880 interrupt related issues.

The PrPMC880 uses the MV64360 interrupt controller to route internal and external interrupt requests to the CPU and the PCI bus. The MV64360 interrupt controller registers are implemented as part of the CPU interface unit in order to have minimum read latency from CPU interrupt handler. This is not backward compatible with the Discovery implementation since the registers are placed at different offsets. The external interrupt sources will use the GPP interface to register external interrupts.

Table 7-8. MV64360 Interrupt Controller Sources

GPP Group	MV64360 MPP Pin #	Edge/Level	Polarity	Interrupt Source	Notes
	GPP[0]	Level	High		
	GPP[1]	Level	High		
	GPP[2]	Level	Low		
	GPP[3]	Level	Low		
	GPP[4]	Level	Low		
	GPP[5]	Level	Low		
	GPP[6]	Level	Low		
	GPP[7]	Level	Low	PHY_INT	
	GPP[16]	Level	Low	PCI_INTA#	
	GPP[17]	Level	Low	PCI_INTB#	
	GPP[18]	Level	Low	PCI_INTC#	
	GPP[19]	Level	Low	PCI_INTD#	
	GPP[20]	Level	Low		
	GPP[21]	Level	Low		
	GPP[22]	Level	Low		

Table 7-8. MV64360 Interrupt Controller Sources (continued)

GPP Group	MV64360 MPP Pin #	Edge/Level	Polarity	Interrupt Source	Notes
	GPP[23]	Level	Low		
	GPP[24]	Level	Low	Reserved for SROM initialization active InitActoutput	
	GPP[25]	Level	Low	Reserved for Watchdog Timer WDE# output	
	GPP[26]	Level	Low	Reserved for Watchdog Timer WDNMI# output	
	GPP[27]	Level	Low		Reserved for future device interrupt

Endian Issues

The Discovery II System Controller supports only a big endian CPU bus. The designation of the local memory (DDR and SRAM) is also big endian. Data transferred to/from the local memory is never swapped. The internal registers of the System Controller are always programmed in little endian. On a CPU access to the internal registers, data is byte-swapped.

Data swapping on a CPU access to the PCI is controlled via PCISwap bits of each PCI Low Address register. This configurable setting allows a CPU access to PCI agents with a different endian convention.

For software compatibility with the GT-64120/130 devices, the system controller maintains ByteSwap and WordSwap bits in the PCI Command register. If the PCI Command register's SwapEn bit is set to 1, the System Controller's PCI master performs data swapping according to PCISwap bits setting. If set to 0 (default), it works according to ByteSwap and WordSwap bits setting. Refer to the *Discovery II System Controller Data Sheet* for additional information and programming.

MOTLoad Vital Product Data 8

Introduction

Nonvolatile data is stored data that remains in memory after power-down. Some of the data is permanent and fixed while other portions are temporary and can be modified. Most of the permanent data is entered by the factory at the time the product is built, while the temporary or variable data is entered by the user after the product start-up. There are three types of nonvolatile data in MOTLoad:

1. **Vital Product Data (VPD)**: describes the unique characteristics of a specific board, such as marketing product number, serial number, assembly number, processor family, hardware clock frequencies, and component configuration information. Because most of the information is unique to that board, it is considered permanent and is not usually changed by the user. Since the firmware uses certain VPD information during the boot process, changing this information can prevent the firmware from coming on-line (for example., no firmware prompt) making the board inoperable or unstable.
2. **Serial Presence Detect (SPD)**: device-specific parameters, such as information for memory devices. This data is determined by the device itself and is stored in a private nonvolatile storage device.
3. **Global Environment Variables (GEVs)**: any stored information that the user may want to define on a board-by-board basis for use from one power-up to another. Boards can operate without any GEV, but errors may occur. However, even if errors occur, or the GEV is missing, the firmware should still come on-line and display a prompt.

Vital Product Data (VPD) Use

This section briefly explains the purpose of VPD, and describes how to read, archive, and edit that information.

Purpose

The purpose of the Vital Product Data (VPD) portion of nonvolatile data is to store board-specific information that is not easily retrievable from other software sources. It is considered permanent and should not be changed. The information is useful during board initialization, configuration and verification. The firmware (in this case MOTLoad) uses some of this information during the boot process. This information can also be accessed by the user. Refer to the remainder of this section to learn how to access and read this information.

The VPD values for a specific board are unique for that board and should not be used on any other board. Hardware and software developers, as well as factory analysis technicians, may need to change certain VPD values, but non-technical users should not, since improper modifications can impair board operation, functionality, or prevent access to firmware prompts.

Note If a firmware prompt is not available, the Safe Start option should be used to bring up a prompt on the system console, from which the VPD can be manually restored.

How to Read VPD Information

VPD information is stored in a fixed address portion of memory, usually SROM or EEPROM. It can be viewed by entering the following MOTLoad command:

vpdDisplay

If the VPD is valid, **vpdDisplay** provides a formatted output of all the VPD packets in the SROM. The VPD specification should be referenced to determine the meaning of each field of the various packet types.

For most hardware products, the following elements are defined at the factory:

- ❏ Product Identifier (PRPMC880-xxxx)
- ❏ Manufacturing Assembly Number (01-W3830F0x)
- ❏ Serial Number (of the assembled board product)
- ❏ Processor Family Number (7447)
- ❏ Hardware clock frequencies (internal, external, fixed, PCI bus)
- ❏ Component configuration information (connectors, Ethernet address(es), other addresses, flash bank ID, L2 cache ID)
- ❏ Security Information (VPD type, version and revision data, 32-bit crc protection)

How to Archive VPD Information

Prior to modifying any elements of VPD, first create an archive copy of the initial VPD contents. Use the archive copy later to restore the VPD to its original state, if necessary.

Although the VPD information should not be altered by the typical user, there may be a need to do so.The procedure below shows how to archive the current VPD contents.

1. Read the VPD into the default user area of memory with a command similar to:

   sromRead -d/dev/i2c0/srom/A8 -n400

2. Create a file of the original VPD with a command similar to: **tftpPut -n0x400 -cBOARD_IP_HERE -fpath_and_filename -d/dev/enet2 -sSERVER_IP_HERE**

Note The command lines shown above must be customized for the board being used. The VPD SROM device string passed to **sromRead** must match the board. The Ethernet device string must also be for that board, as well as the IP addresses being used. The *-n* (size) option should match the MOTLoad SROM size.

The resulting file (path_and_filename) will be a binary file whose length is determined by the *-n* (size) option. Save this binary file, to use later to restore the board VPD if necessary.

Restoring the Archive

As mentioned in the previous section, before you modify any elements of the VPD, create an archive copy of the initial VPD contents to use later to restore the VPD to its previous contents.

Take extreme care when writing to the VPD SROM. Incorrect VPD values can prevent a board from reaching the MOTLoad command prompt. If this occurs, Safe Start, a jumper option on some hardware products, should be used.

The following sequence shows how to restore the archived VPD contents. A list of MOTLoad commands can be found in the *MOTLoad Firmware Package User's Manual*, listed in Appendix C, *Related Documentation* of this manual.

tftpGet -n0x400 -cBOARD_IP_HERE -fpath_and_filename -d/dev/enet2 -sSERVER_IP_HERE
sromWrite -d/dev/i2c0/srom/A8 -n400

Note The command lines shown above must be changed to reflect the specific board being used. The VPD SROM device string passed to sromWrite needs to match the board. The Ethernet device string needs to be appropriate for the board, as do the IP addresses being used. It is very important to use the data file for the exact board to which the restoration is being done. The *-n* (size) option should match the MOTLoad SROM size, which is defined by the Vital Product Data specification.

Modifying the VPD

The MOTLoad **vpdEdit** command allows the VPD to be modified. Make sure that the proper safeguards have been put in place prior to modifying. For example, the VPD should be both understood, and archived, prior to

applying any changes. Incorrect VPD values can prevent a board from reaching the MOTLoad command promp.

The edit session prompts the user with each byte currently in VPD, and provides the option of changing the byte by typing in a new value (a byte in hexadecimal). Or the existing value can be kept by entering a **Return**.

The following editing session entries have special meaning:

^ (caret)

reverse edit order. This is helpful if the byte that needs to be modified has been passed up during the edit session.

v (lowercase v)

edit in “normal” order again. This is handy after having used the ^, described above.

. (period)

stop editing and query user as to whether edits are to be saved in SROM.

Below is an example of an editing session. Note that the addresses increment until the ^ is entered, then decrement until the “v” is entered.

```
vpdEdit 00A67000 4D?
 00A67001 4F?
 00A67002 54?
 00A67003 4F?
 00A67004 52?
 00A67005 4F? ^
 00A67004 52?
 00A67003 4F?
 00A67002 54? V.
 00A67003 4F?
 00A67004 52?
 00A67005 4F? .

Program VPD SROM (Y/N)? n
```

If you answer YES to the Program VPD SROM (Y/N)?, then the modifications are written to the VPD SROM. A new checksum is

calculated and written. Answering NO prevents any change to the existing SROM contents.

Correcting VPD Problems

If for some reason the VPD information becomes corrupted, the following occurs:

- ❏ A warning is displayed in the startup banner
- ❏ The firmware ignores the VPD contents and attempts to acquire information from other sources
- ❏ Some device drivers will not work
- ❏ Some diagnostic tests fail
- ❏ The board runs much slower than usual

How to Fix Corrupted VPD Information

Use the **sromRead**, **sromWrite**, **vpdEdit** or **vpdDisplay** command to edit and fix the corrupted VPD.

What if Your Board Has the Wrong VPD?

If for some reason your board has the wrong VPD information, the following occurs:

- ❏ No warning is displayed
- ❏ The firmware believes the incorrect VPD information
- ❏ The board may hang during startup (no-start condition)
- ❏ The board may be very unstable if it reaches the prompt
- ❏ Device drivers, diagnostic tests, and firmware commands may hang or fail in unexpected ways

How to Fix Wrong VPD Information

If you suspect that your board has problems as a result of wrong VPD information, use the **sromRead**, **sromWrite**, **vpdEdit** or **vpdDisplay** command to edit and fix the corrupted VPD

The data listed in the following tables are for general reference information. It is divided into two major sections: *VPD Packet Types* which define VPD packet formats, and *VPD Content Information* which includes information on what is actually contained in the VPD.

VPD Data Definitions

The second 256KB of the EEPROM contain Vital Product Data (VPD) configuration information specific to the PrPMC880. Typical information that may be present in the EEPROM include: manufacturer, board revision, build version, date of assembly, memory present, options present, etc.

VPD Packet Types

The following sections describe possible values of the onboard VPD for various configurations required to properly configure the PrPMC880. For further information on Vital Product Data, refer to the *MOTLoad Firmware Package User's Manual*, listed in Appendix C, *Related Documentation*.

The following table describes and lists the currently assigned packet identifiers. Note: Additional packet identifiers may be added to this list as future versions of the VPD are released.

Table 8-1. VPD Packet Types

ID#	Size	Packet Description	Data Type	Notes
00	N/A	Guaranteed Illegal	N/A	
01	Variable	Product Identifier (PrPMC880)	ASCII	1
02	Variable	Factory Assembly Number ("01-WxxxFxxx", etc.)	ASCII	1
03	Variable	Serial Number Packet	ASCII	1

Notes

1. The data size is variable. Its actual size is dependent upon the product configuration/type.
2. Integer values are formatted/stored in big-endian byte ordering.
3. This packet may be omitted if the Ethernet interface is nonexistent, or the Ethernet interface has an associative SROM.
4. This packet may contain an additional byte following the address data. This additional byte indicates the Ethernet interface number. This additional byte would be specified in applications where the host product supports multiple Ethernet interfaces. For each Ethernet interface present, the instance number would be incremented by one starting with zero.

Table 8-1. VPD Packet Types (continued)

ID#	Size	Packet Description	Data Type	Notes
04	10	Product Configuration Options Data The data in this packet further describes the board configuration (header population, I/O routing, etc.). Its exact contents is dependent upon the product configuration/type, refer to Table 8-2 on page 8-11	Binary	
05	05	MPU Internal Clock Frequency in Hertz 1,000,000,000 decimal, etc.), refer to Table 8-3 on page 8-13	Integer (4-byte)	2
06	05	MPU External Clock Frequency in Hertz (133,000,000 decimal, etc.). This is also called the local processor bus frequency, refer to Table 8-4 on page 8-14	Integer (4-byte)	2
07	05	Reference Clock Frequency in Hertz (133,000,000 decimal, etc.). This value is the frequency of the crystal driving the baudout bit, refer to Table 8-5 on page 8-14	Integer (4-byte)	2
08	07	Ethernet MAC Address, refer to Table 8-7 on page 8-15	Binary	3, 4
09	Variable	MPU Type, refer to Table 8-8 on page 8-16.	ASCII	1
0A	04	EEPROM CRC This packet is required. When computing the CRC, this field (for example, 4 bytes) is set to zero.This CRC only covers the range as specified in the size field, refer to Table 8-9 on page 8-16	Integer (4-byte)	2
0B	0C	Flash Memory Configuration, refer to Table 8-13 on page 8-21	Binary	

Notes

1. The data size is variable. Its actual size is dependent upon the product configuration/type.
2. Integer values are formatted/stored in big-endian byte ordering.
3. This packet may be omitted if the Ethernet interface is nonexistent, or the Ethernet interface has an associative SROM.
4. This packet may contain an additional byte following the address data. This additional byte indicates the Ethernet interface number. This additional byte would be specified in applications where the host product supports multiple Ethernet interfaces. For each Ethernet interface present, the instance number would be incremented by one starting with zero.

Table 8-1. VPD Packet Types (continued)

ID#	Size	Packet Description	Data Type	Notes
0C	00	VLSI Device Revisions/Versions, refer to *VLSI Device Revision Packet, 0x0c* on page 8-17	Binary	
0D	05	Host PCI-Bus clock frequency in Hertz (33,333,333 decimal, etc.), refer to *Host PCI Bus Clock Frequency Packet, 0x0d* on page 8-17	Integer (4-byte)	2
0E	0F	L2 Cache Configuration Packet, refer to Table 8-11 on page 8-18		
0F	04	VPD Revision, refer to Table 8-18 on page 8-36	Binary	
10	Variable	Obsolete. Not for use with MOTLoad firmware	Binary	
14	18	Reserved		
1a		Variable mesh packet		
1b	FD	Reserved (do not use)		
FE		Varies redefined packet		
FF	N/A	VPD termination marker		

Notes

1. The data size is variable. Its actual size is dependent upon the product configuration/type.
2. Integer values are formatted/stored in big-endian byte ordering.
3. This packet may be omitted if the Ethernet interface is nonexistent, or the Ethernet interface has an associative SROM.
4. This packet may contain an additional byte following the address data. This additional byte indicates the Ethernet interface number. This additional byte would be specified in applications where the host product supports multiple Ethernet interfaces. For each Ethernet interface present, the instance number would be incremented by one starting with zero.

Product Configuration Options Data

The product configuration options data packet consists of general component and connector population information.The data section consists of an array of bits which correspond to a specific device instance, the first bit of the first byte is bit 0 (for example, PowerPC bit numbering). An option is present when the assigned bit is a one. The following table further describes the product configuration options VPD data packet:

Table 8-2. PrPMC880 Product Configuration Options Data

Bit	Mnemonic	Description
0	PCO_PCI0_CONN1	PCI bus 0 connector 1
1	PCO_PCI0_CONN2	PCI bus 0 connector 2
2	PCO_PCI0_CONN3	PCI bus 0 connector 3
3	PCO_PCI0_CONN4	PCI bus 0 connector 4
4	PCO_PCI1_CONN1	PCI bus 1 connector 1
5	PCO_PCI1_CONN2	PCI bus 1 connector 2
6	PCO_PCI1_CONN3	PCI bus 1 connector 3
7	PCO_PCI1_CONN4	PCI bus 1 connector 4
8	PCO_ISA_CONN1	ISA bus connector 1
9	PCO_ISA_CONN2	ISA bus connector 2
10	PCO_ISA_CONN3	ISA bus connector 3
11	PCO_ISA_CONN4	ISA bus connector 4
12	PCO_EIDE1_CONN1	IDE/EIDE controller 1 connector 1
13	PCO_EIDE1_CONN2	IDE/EIDE controller 1 connector 2
14	PCO_EIDE2_CONN1	IDE/EIDE controller 2 connector 1
15	PCO_EIDE2_CONN2	IDE/EIDE controller 2 connector 2
16	PCO_ENET1_CONN	Ethernet controller 1 connector
17	PCO_ENET2_CONN	Ethernet controller 2 connector
18	PCO_ENET3_CONN	Ethernet controller 3 connector

Table 8-2. PrPMC880 Product Configuration Options Data (continued)

Bit	Mnemonic	Description
19	PCO_ENET4_CONN	Ethernet controller 4 connector
20	PCO_SCSI1_CONN	SCSI controller 1 connector
21	PCO_SCSI2_CONN	SCSI controller 2 connector
22	PCO_SCSI3_CONN	SCSI controller 3 connector
23	PCO_SCSI4_CONN	SCSI controller 4 connector
24	PCO_SERIAL1_CONN	Serial port 1 connector
25	PCO_SERIAL2_CONN	Serial port 2 connector
26	PCO_SERIAL3_CONN	Serial port 3 connector
27	PCO_SERIAL4_CONN	Serial port 4 connector
28	PCO_FLOPPY_CONN1	Floppy disk controller connector 1
29	PCO_FLOPPY_CONN2	Floppy disk controller connector 2
30	PCO_PARALLEL1_CONN	Parallel port 1 connector
31	PCO_PARALLEL2_CONN	Parallel port 2 connector
32	PCO_PMC1_IO_CONN	PMC slot 1 I/O connector
33	PCO_PMC2_IO_CONN	PMC slot 2 I/O connector
34	PCO_USB0_CONN	USB channel 0 connector
35	PCO_USB1_CONN	USB channel 1 connector
36	PCO_KEYBOARD_CONN	Keyboard controller connector
37	PCO_MOUSE_CONN	Mouse controller connector
38	PCO_VGA1_CONN	VGA controller 1 connector
39	PCO_SPEAKER_CONN	Speaker timer connector - 12 - 3/3/2003
40	PCO_VME_CONN	VME backplane connector
41	PCO_CPCI_CONN	Compact PCI backplane connector
42	PCO_ABORT_SWITCH	Abort switch
43	PCO_BDFAIL_LIGHT	Board fail light

Table 8-2. PrPMC880 Product Configuration Options Data (continued)

Bit	Mnemonic	Description
44	PCO_SWREAD_HEADER	Software readable header
45	PCO_MEMMEZ_CONN	Memory mezzanine connector
46	PCO_PCI0_EXP_CONN	PCI bus 0 expansion connector
47	[Reserved]	[Reserved]
48	PCO_DIMM1_CONN	DIMM slot 1 connector
49	PCO_DIMM2_CONN	DIMM slot 2 connector
50	PCO_DIMM3_CONN	DIMM slot 3 connector
51	PCO_DIMM4_CONN	DIMM slot 4 connector
52 -127	Reserved	Reserved

Internal Processor Clock Frequency Packet, 0x05

The internal processor clock frequency packet contains the internal processor clock frequency, which is usually a fractional multiple of the external processor bus frequency in Hertz. the VPD integers are stored in big-endian format.

Table 8-3. Internal Processor Clock Frequency Packet

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x05
2	UCHAR	Size of the following data section in bytes must be set to 0x05
3 - 6	UINT	Internal processor clock frequency in Hertz
7	UCHAR	Which processor does this packet describe: 0x01: 1st Processor 0x02: 2nd Processor 0x04: 3rd Processor 0x08: 4th Processor, etc.

External Processor Clock Frequency Packet, 0x06

The external processor clock frequency packet contains the external processor bus frequency in Hertz. The Vital Product Data integers are stored in big-endian format.

Table 8-4. External Processor Clock Frequency Packet, 0x06

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x06
2	UCHAR	Size of the following data section in bytes must be set to 0x05
3 - 6	UINT	External processor clock frequency in Hertz
7	UCHAR	0x01: 1st Processor 0x02: 2nd Processor 0x04: 3rd Processor 0x08: 4th Processor, etc.

Reference Clock Frequency Packet, 0x07

The reference clock frequency packet contains the architecture -specific frequency value for a software readable reference timer or register bit. See the appropriate architecture description in Table 8-6 for the default value implied by the absence of the packet. The Vital Product Data integers are stored in big-endian format.

Table 8-5. Reference Clock Frequency Packet, 0x07

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x07
2	UCHAR	Size of the following data section in bytes must be set to 0x05
3 - 6	UINT	Reference clock frequency in Hertz, refer to the following table.
7	UCHAR	I/O bus-specific instance number (start at 0, increment by 1 for each additional instance)

Table 8-6. Architecture Description for Reference Clock

Architecture	Default Value	Description
PowerPlus I	14,318,180 Hz	Frequency of the ISA bridge 8254 timer oscillator
PowerPlus I I	153,600 Hz	Frequency of the HAWK_PX_BASE + 0x8080 BAUDOUT bit for a UART running at 9600 baud
PowerPlus III	28,800 Hz	Frequency of the HARRIER_XCSR_BASE + 0x00D0 XTAL64 bit
MPC106	14,318,180 Hz	Frequency of the ISA bridge 8254 timer oscillator
Power Chap I	153,600 Hz	Frequency of the CHAP_PX_BASE + 0x0000 REF_CLK bit for a UART running at 9600 baud

Ethernet MAC Address Packet, 0x08

This packet contains the MAC address for a single Ethernet controller. Each Ethernet controller that doesn't have an SROM-based MAC address should have an Ethernet MAC address packet in VPD. Ethernet MAC addresses are in big-endian format.

Table 8-7. Ethernet MAC Address Packet, 0x08

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x08
2	UCHAR	Size of the following data section in bytes must be set to 0x05
3 - 8	UCHAR[6]	Ethernet MAC address
9	UCHAR	Ethernet controller instance number (starts at 0, increments by 1 for each additional controller)

Processor Identifier Packet, 0x09

This packet contains the ASCII characters used in the board's processor identifier string. The VPD ASCII strings do not contain null-terminators.

Table 8-8. Processor Identifier Packet, 0x09

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x09
2	UCHAR	Size of the following data section in bytes must be set to *n*
3	UCHAR	ASCII character 1
4	UCHAR	ASCII character 2
...	...	...
N=2	UCHAR	ASCII character *n*

EEPROM CRC Packet, 0x0a

This packet contains the 32-bit CRC value for the VPD SROM block. This packet must be present is a valid PowerPC VPD block. VPD data integers are stored in big-endian format.

Note For information of calculating the VPD, refer to *vpdGenerateCRC* on page 8-38.

Table 8-9. EEPROM CRC Packet, 0x0a

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x0a
2	UCHAR	Size of the following data section in bytes must be set to 0x04
3 - 6	UINT	VPD block CRC value (set to 0 when computing CRC value)

VLSI Device Revision Packet, 0x0c

The VLSI device revision packet is not implemented in this release.

Host PCI Bus Clock Frequency Packet, 0x0d

The host PCI bus clock frequency packet contains the host PCI bus frequency in Hertz. This packet shouldn't be present on boards that don't control the host PCI bus frequency. The VPD integers are stored in big-endian format.

Table 8-10. Host PCI Bus Clock Frequency Packet, 0x0d

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x0d
2	UCHAR	Size of the following data section in bytes must be set to 0x05
3 - 6	UINT	Host PCI bus clock frequency in Hertz
7	UCHAR	PCI host bridge instance number (starts at 0, increments by 1 for each bridge)

L2 Cache Configuration Packet, 0x0e

The L2 cache configuration packet contains multiple fields that specify cache configuration parameters for a single L2 cache memory array. TheL2 cache is either internal or external. Each external L2 cache memory array should have a:

- ❏ Configuration packet in VPD with all elements shown in the next table
- ❏ Configuration packet with only elements marked "always" included)

If the configuration packet is missing from the VPD for a board that has an L2 controller, the L2 cache does not initialize or enable.

The VPD integers are stored in big-endian format.

Table 8-11. L2 Cache Configuration Packet, 0x0e

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x0e (always included)
2	UCHAR	Size of the following data section in bytes must be set to 0x0f (always included)
3 – 4	USHORT	Vendor identifier 0x0000 – 0xFFFE: manufacturer assigned identifier 0xFFFF: unknown vendor identifier
5 – 6	USHORT	Device identifier 0x0000 – 0xFFFE: manufacturer assigned identifier 0xFFFF: unknown device identifier
7	UCHAR	Single device data width in bits
8	UCHAR	Total number of devices or sockets present
9	UCHAR	Number of interleave columns (contiguous data bytes alternate through these columns at the following width)
10	UCHAR	Column width in bits (always a multiple of the device data width)
11	UCHAR	L2 cache type: 0x00: MPC750/MPC7400/etc. backside cache at SPR1017 0x01: glance/double take cache at HAWK_PX_BASE+0x8000 0x02: IBM15-C700A in-line cache at HAWK_PX_BASE+0x8X00 0x03 – 0xFF: [unused]
12	UCHAR	0x02: 2nd Processor 0x04: 3rd Processor 0x08: 4th Processor, etc.
13	UCHAR	Write operation capabilities: 0x00: write-through and write-back (software configurable) 0x01: write-through and write-back (hardware configurable) 0x02: write-through only 0x03: write-back only 0x04 – 0xFF: [unused]
14	UCHAR	Error detection type (always included) 0x00: none 0x01: parity 0x02: ECC 0x03 – 0xFF: [unused]

Table 8-11. L2 Cache Configuration Packet, 0x0e (continued)

Byte	Field Type	Description
15	UCHAR	L2 cache memory array size [(1 << n) * 256KB] 0x00: 256KB 0x01: 512KBs 0x02: 1MB 0x03: 2MB 0x04: 4MB
16	UCHAR	Backside cache configuration: 0x00: late write sync, 1ns hold, differential clock, parity 0x01: pipelined sync burst, .5ns hold, no differential clock, parity 0x02: late write sync, 1ns hold, differential clock, no parity 0x03: pipelined sync burst, .5ns hold, no differential clock, no parity 0x04 – 0xFF: [unused]
17	UCHAR	Internal processor frequency / backside cache frequency ratio 0x00: L2 clock and DLL disabled 0x01: 1:1 (1.0) 0x02: 3:2 (1.5) 0x03: 2:1 (2.0) 0x04: 5:2 (2.5) 0x05: 3:1 (3.0) 0x06: 7:2 (3.5) 0x07: 4:1 (4.0) 0x08 – 0xFF: [unused]

Flash Memory Configuration Packet, 0x0b

The flash memory configuration packet contains multiple fields that specify memory configuration parameters for a single bank of flash memory. Each flash memory bank should have a corresponding flash memory configuration packet in VPD. If a bank has multiple packets, the packet with the highest address in the SROM is used by the MOTLoad flash driver. VPD data integers are stored in big-endian format.

Table 8-12. Flash Memory Configuration Packet, 0x0b

Byte	Field Type	Description
1	UCHAR	Unique VPD packet identifier must be set to 0x0b
2	UCHAR	Size of the following data section in bytes must be set to 0x0c
3 - 4	USHORT	Vendor identifier 0x0000 - 0xFFFE: manufacturer assigned identifier 0xFFFF: unknown vendor identifier, or removable device
5 - 6	USHORT	Device identifier 0x0000 - 0xFFFE: manufacturer assigned identifier 0xFFFF: unknown or removable device identifier
7	UCHAR	Single device data width in bits
8	UCHAR	Total number of devices or sockets present
9	UCHAR	Number of interleave columns (contiguous data bytes alternate through these columns at the following width)
10	UCHAR	Column width in bits (always a multiple of the device data width)
11	UCHAR	Minimum write/erase data width in bits (multiple FLASH memory devices must be programmed in parallel when this value exceeds the data width of a single device)
12	UCHAR	Flash bank number 0x00 - 0x03: system memory controller 0 banks A - D 0x10 - 0x13: system memory controller 1 banks A - D 0x20 - 0x23: system memory controller 2 banks A - D 0x30 - 0x33: system memory controller 3 banks A - D 0x40 - 0xFF, etc.: [unused]

Table 8-12. Flash Memory Configuration Packet, 0x0b (continued)

Byte	Field Type	Description
13	UCHAR	ROM access speed in nanoseconds
14	UCHAR	Total bank size [(1 << n) * 256KB] 0x00: 256KB 0x01: 512KB 0x02: 1MB 0x03: 2MB 0x04: 4MB 0x05: 8MB

Flash Memory Configuration Data, 0x0b

The flash memory configuration data packet consists of byte fields that indicate the size organization and type of the flash memory array. The next table an example VPD data packet.

Table 8-13. Example Flash Memory Configuration Data Packet, 0x0b

Byte Offset	Field size	Field Mnemonic	Field Description
00	2	FMC_MID	Manufacturer's Identifier (0089 = Intel)
02	2	FMC_DID	Manufacturer's Device Identifier(0018 for 28F128J3A)
04	1	FMC_DDW	Device data width (16-bits)
05	1	FMC_NOD	Number of device sockets present (2)
06	1	FMC_NOC	Number of columns (interleaves) (1)
07	1	FMC_CW	Column width in bits (32) is always a multiple of the device's data width.
08	1	FMC_WEDW	Write/Erase data width (32) the flash memory devices must be programmed in parallel when the write/erase data width exceeds the device's data width.
09	1	FMC_BANK	Bank number of flash memory array (0 = Bank A)

Table 8-13. Example Flash Memory Configuration Data Packet, 0x0b

Byte Offset	Field size	Field Mnemonic	Field Description
0A	1	FMC_SPEED	ROM access speed in nanoseconds(150)
0B	1	FMC_SIZE	Total bank size (should agree w/the physical organization above): 05 = 8 MB 06 = 16 MB 07 = 32 MB 08 = 64 MB

VPD SROM Contents for 01-W3830Fxx

The next table lists the variable contents of the VPD that is programmed onto the SROM for each of the board models. In Table 8-14, the board numbers marked with an asterisk (*) must be assigned with a unique entity because the data is not static. Static contents for each board are listed in Table 8-15.

Table 8-14. Variable VPD Content Specifications

Offset	01-W3830F05*	01-W3830F06*	01-W3830F07*	01-W3830F08*
Field Description: Product Identifier (Packet ASCII)				
21 (0x15)	32	33	32	33
22 (0x16)	32	32	32	32
23 (0x17)	36	36	37	37
24 (0x18)	31	31	31	31
Field Description: Assembly Number (Packet ASCII)				
36 (0x24)	30	30	30	30
37 (0x25)	35	36	37	38
38 (0x26)	**xx**	**xx**	**xx**	**xx**
Note For the assembly number (**xx** = 01-W3830F**xx***'s **xx**				
Field Description: Product Configuration Options Data (Packet Binary)				

Table 8-14. Variable VPD Content Specifications (continued)

Offset	01-W3830F05*	01-W3830F06*	01-W3830F07*	01-W3830F08*
50 (0x32)	80	80	80	80
Field Description: Internal Processor Clock Frequency (Packet Binary)				
66 (0x42)	05	05	05	05
67 (0x43)	05	05	05	05
68 (0x44)	3B	3B	3B	3B
69 (0x45)	9A	9A	9A	9A
70 (0x46)	CA	CA	CA	CA
71 (0x47)	00	00	00	00
72 (0x48)	01	01	01	01
Field Description: MPU type (Packet ASCII)				
105 (0x69)	09	09	09	09
106 (0x6a)	07	07	07	07
107 (0x6b)	4D	4D	4D	4D
108 (0x6c)	50	50	50	50
109 (0x6d)	43	43	43	43
110 (0x6e)	37	37	37	37
111 (0x6f)	34	34	34	34
112 (0x70)	34	34	34	34
113 (0x71)	37	37	37	37

Table 8-15. Static VPD Content Specifications

Offset	Value	Field Type	Description
00 (0x00)	4D	ASCII	Eye Catcher (Motorola) **Note** Starting byte for the calculation of CRC.
01 (0x01)	4F		

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
02 (0x02)	54		
03 (0x03)	4F		
04 (0x04)	52		
05 (0x05)	4F		
06 (0x06)	4C		
07 (0x07)	41		
08 (0x08)	02	Binary	Size of VPD area is bytes. The size is viewed as logical, it is not the size of the EEPROM
09 (0x09)	00		
10 (0x0a)	01	Packet ASCII	Product Identifier {PrPMC880-xxx}. Table 8-14 for xx values. See Notes 1 and 3
11 (0x0b)	0D		
12 (0x0c)	50		
13 (0x0d)	72		
14 (0x0e)	50		
15 (0x0f)	4D		
16 (0x10)	43		
17 (0x11)	38		
18 (0x12)	38		
19 (0x13)	30		
20 (0x14)	2D		
21 (0x15)	xx		
22 (0x16)	xx		
23 (0x17)	xx		
24 (0x18)	xx		
25 (0x19)	02	Packet ASCII	Factory assembly number (01-W3830Fxx*)

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
26 (0x1a)	0C		
27 (0x1b)	30		
28 (0x1c)	31		
29 (0x1d)	2D		
30 (0x1e)	57		
31 (0x1f)	33		
32 (0x20)	38		
33 (0x21)	33		
34 (0x22)	30		
35 (0x23)	46		
36 (0x24)	**xx**		To be filled in at BS. See Table 8-14 for xx values. See Notes 1, 2
37 (0x25)	**xx**		
38 (0x26)	**xx**		
39 (0x27)	03		Serial number to be filled in at BS
40 (0x28)	07		
41 (0x29)	**xx**		
42 (0x2a)	**xx**		
43 (0x2b)	**xx**		
44(0x2c)	**xx**		
45 (0x2d)	**xx**		
46 (0x2e)	**xx**		
47 (0x2f)	**xx**		
48 (0x30)	04	Packet binary	Product configuration options data. Refer to Table 8-14 for xx values
49 (0x31)	10		

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
50 (0x32)	80		
51 (0x33)	00		
52 (0x34)	80		
53 (0x35)	80		
54(0x36)	00		
55 (0x37)	18		
56 (0x38)	00		
57 (0x39)	00		
58 (0x3a)	00		
59 (0x3b)	00		
60 (0x3c)	00		
61 (0x3d)	00		
62 (0x3e)	00		
63 (0x3f)	00		
64 (0x40)	00		
65 (0x41)	00		
66 (0x42)	05	Packet binary	Internal Processor Clock Frequency, refer to Table 8-14 for xx values
67 (0x43)	05		
68 (0x44)	**xx**		
69 (0x45)	**xx**		
70 (0x46)	**xx**		
71 (0x47	**xx**		
72 (0x48)	**xx**		
73 (0x49)	06	Packet Binary	External Processor Clock Frequency
74 (0x4a)	05		

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
75 (0x4b)	07		
76 (0x4c)	F2		
77 (0x4d)	81		
78 (0x4e)	55		
79 (0x4f)	01		
80 (0x50)	07	Packet binary	Reference Clock Frequency
81 (0x51	05		
82 (0x52)	07		
83 (0x53	F2		
84 (0x44)	81		
85 (0x55)	55		
86 (0x56)	00		
87 (0x57)	08	Packet binary	GEnet MAC 1 Address
88 (0x58)	07		
89 (0x59)	**xx**		
90 (0x5a)	**xx**		
91 (0x5b)	**xx**		
92 (0x5c)	**xx**		
93 (0x5d	**xx**		
94 (0x4e)	**xx**		
95 (0x5f)	00		
96 (0x60)	08	Packet binary	GEnet MAC2 Address
97 (0x61)	07		
98 (0x62)	**xx**		
99 (0x63)	**xx**		

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
100 (0x64)	**xx**		
101 (0x65)	**xx**		
102 (0x66)	**xx**		
103 (0x67)	**xx**		
104 (0x68)	01		
105 (0x69)	09	Packet ASCII	MPU Type
106 (0x6a)	07		
107 (0x6b)	4D		
108 (0x6c)	50		
109 (0x6d)	43		
110 (0x6e)	37		
111 (0x6f)	34		
112 (0x70)	34		
113 (0x71)	37		
114 (0x72)	0A	Packet binary	EPROM CRC
115 (0x73)	04		
116 (0x74)	**xx**		
117 (0x75)	**xx**		
118 (0x76)	**xx**		
119 (0x77)	**xx**		
120 (0x78)	0B	Packet binary	Flash Memory Configuration (Bank A)
121 (0x79)	0C		
122 (0x7a)	00		
123 (0x7b	89		Intel driver
124 (0x7c)	88	Intel flash type	Intel 28F256k3, 32MB flash chip, total 64MB flash board

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
125 (0x7d)	03		
126 (0x7e)	10		
127 (0x7f)	02		
128 (0x80)	01		
129 (0x81)	40		
130 (0x82)	40		
131 (0x83)	00		
132 (0x84)	8C		
133 (0x85)	08		
134 (0x86)	0D	Packet binary	
135 (0x87)	05		
136 (0x88)	01		
137 (0x89)	FC		
138 (0x8a)	A0		
139 (0x8b)	55		
140 (0x8c)	00		
141 (0x8d)	0E	Packet binary	L2 Cache Configuration
142 (0x8e)	0F		
143 (0x8f)	FF		
144 (0x90)	FF		
145 (0x91)	FF		
146 (0x92)	FF		
147 (0x93)	FF		
148 (0x94)	FF		
149 (0x95)	FF		

Table 8-15. Static VPD Content Specifications (continued)

Offset	Value	Field Type	Description
150 (0x96)	FF		
151 (0x97)	FF		
152 (0x98)	01		
153 (0x99)	FF		
154 (0x9a)	01		
155 (0x9b)	FF		
156 (0x9c)	FF		
157 (0x9d)	FF		
158 (0x9e)	0F	Packet binary	VPD Revision
159 (0x9f)	04		
160 (0xa0)	00		
161 (0xa1)	03		
162 (0xa2)	00		
163 (0xa3)	00		
164 (0xa4)	FF	Binary	Reserved for future expansion
. .	. .		
511 (0x1FF)	FF		Reserved for future expansion. **Note** End byte for the calculation of CRC

SPD Contents for 01-W3830Fxx Boards

The next table describes the static and variable contents of the SPD that are programmed onto the board.

Notes

1. 256MB SDRAM. For F05, F06 boards, the SPD for 256MB using one bank of nine 8-bit wide, 256MB devices. These have a CAS latency of 2.5 at 11ns clock cycle. CL2.5 affects byte 23 decimal. In this case a c0 meaning CL of 2 requires 12ns.
2. 512MB SDRAM. for F07, F08 boards, the SPD for 512MB using one bank of nine 8-bit wide, 512MB devices. These have a CAS latency of 2.5 at 11ns clock cycle. CL2.5 affects byte 23 decimal. In this case it is a c0 meaning CL of 2 requires 12ns.

Table 8-16. Variable SPD SROM Configuration Specifications

Assembly Number	**01-W3830F05* 01-W3830F06***	**01-3830F07* 01-W3830F08***
SDRAM Part Number	**52NW9663B61**	**51NW9663C05**
Offset		
03 (0x03)	0D	0D
04 (0x04)	0A	0B
09 (0x9)	75	75
23 (0x17)	75	75
24 (0x18)	75	75
28 (0x1c)	3C	3C
30 (0x1e)	2D	2D

Table 8-16. Variable SPD SROM Configuration Specifications

Assembly Number	01-W3830F05* 01-W3830F06*	01-3830F07* 01-W3830F08*
31 (0x1f)	40	80
63 (0x3F)	8E	D9

Notes

1. This is typically programmed as 128 bytes.
2. This is typically programmed as 256 bytes.
3. High order bit defines if assembly requires redundant addressing (if set to 1, redundant addressing is required).
4. From datasheet.
5. High order bit is Self Refresh flag. If set to 1 the assembly supports self refresh.
6. The JEDEC specification states that these bytes are optional for 66 MHz applications. If they are not included then the SPD revision level (byte 62) is set at revision 1 (01h).

Table 8-17. Static SPD SROM Configuration Specifications

Offset	Value	Field Type	Description
00 (0x00)	80		Number of serial PD bytes written during module production. See Note 1
01 (0x01)	08		Total number of bytes in serial PD device. See Note 2
02 (0x02)	07		Fundamental memory type (FPM, EDO, SDRAM)

Notes

1. This is typically programmed as 128 bytes.
2. This is typically programmed as 256 bytes.
3. High order bit defines if assembly requires redundant addressing (if set to 1, redundant addressing is required).
4. From datasheet.
5. High order bit is Self Refresh flag. If set to 1 the assembly supports self refresh.
6. The JEDEC specification states that these bytes are optional for 66 MHz applications. If they are not included then the SPD revision level (byte 62) is set at revision 1 (01h).

Table 8-17. Static SPD SROM Configuration Specifications (continued)

Offset	Value	Field Type	Description
03 (0x03)	**xx**		Number of row addresses on this assembly. See Note 3, Table 8-16 for xx values
04 (0x04)	**xx**		Number of column addresses on this assembly. See Note 3, Table 8-16 for xx values
05 (0x05)	01		Number of DIMM banks
06 (0x06)	48		Data width of this assembly
07 (0x07)	00		Data width of this assembly
08 (0x08)	04		Voltage interface level of this assembly
09 (0x09)	**xx**		SDRAM cycle time at max supported CAS latency (CL), CL=X. Note 4, Table 8-16 for X values
10 (0x0a)	75		SDRAM access from clock
11 (0x0b)	02		DIMM configuration type (nonparity, parity, or ECC)
12 (0x0c)	82		Refresh rate/type. Notes 4, 5
13 (0x0d)	08		Primary SDRAM width
14 (0x0e)	08		Error checking SDRAM width
15 (0x0f)	01		SDRAM device attributes: min clock delay, back-to-back random column access
16 (0x10)	0E		SDRAM device attributes: burst lengths supported
17 (0x11)	04		SDRAM device attributes: number of banks on SDRAM device. Note 4

Notes

1. This is typically programmed as 128 bytes.
2. This is typically programmed as 256 bytes.
3. High order bit defines if assembly requires redundant addressing (if set to 1, redundant addressing is required).
4. From datasheet.
5. High order bit is Self Refresh flag. If set to 1 the assembly supports self refresh.
6. The JEDEC specification states that these bytes are optional for 66 MHz applications. If they are not included then the SPD revision level (byte 62) is set at revision 1 (01h).

Table 8-17. Static SPD SROM Configuration Specifications (continued)

Offset	Value	Field Type	Description
18 (0x12)	0C		SDRAM device attributes: CAS latency. Note 4
19 (0x13)	01		SDRAM device attributes: CS latency. Note 4
20 (0x14)	02		SDRAM device attributes: write latency. Note 4
21 (0x15)	24		SDRAM module attributes
22 (0x16)	00		SDRAM device attributes: general. Note 4
23 (0x17)	**xx**		Minimum clock cycle at CLX-1. Note 4, Table 8-16 for xx values
24 (0x18)	**xx**		Maximum data access time (t AC) from clock at CLX-2. Note 4, Table 8-16 for xx values
25 (0x19)	00		Minimum clock cycle at CLX-2. Note 4
26 (0x1a)	00		Maximum data access time (t AC) from clock at CLX-2. Note 4
27 (0x1b)	50		Minimum row precharge time (t RP). Note 4
28 (0x1c	**xx**		Minimum row active to row active delay (t RRD). See Note 4, Table 8-16 for xx values.
29 (0x1d	50		Minimum RAS to CAS delay (t RCD). Note 4
30 (0x1e	**xx**		Minimum AS pulse width (t RAS). Note 4, Table 8-16 for xx values
31 (0x1f	**xx**		Module bank density

Notes

1. This is typically programmed as 128 bytes.
2. This is typically programmed as 256 bytes.
3. High order bit defines if assembly requires redundant addressing (if set to 1, redundant addressing is required).
4. From datasheet.
5. High order bit is Self Refresh flag. If set to 1 the assembly supports self refresh.
6. The JEDEC specification states that these bytes are optional for 66 MHz applications. If they are not included then the SPD revision level (byte 62) is set at revision 1 (01h).

Table 8-17. Static SPD SROM Configuration Specifications (continued)

Offset	Value	Field Type	Description
32 (0x20)	90		Address and command setup time before clock. Note 6
33 (0x21)	90		Address and command hold time after clock. Note 6
34 (0x22)	50		Data input setup time before clock. Note 6
35 (0x23)	50		Data input hold time after clock. Note 6
36 (0x24)	00		Reserved for VCSDRAM
. .	. .	. .	. .
40 (0x28)	00		Reserved for VCSDRAM
41 (0x29)	41		Minimum active to active/auto refresh time
42 (0x2a)	4B		Minimum auto-refresh to active/auto refresh command period
43 (0x2b)	30		Maximum cycle time
44(0x2c)	32		SQS-DQ skew for DQS and associated DQ signals
45 (0x2d)	00		Read data hold skew factor
46 (0x2e)	00		Superset information
. .	. .	. .	. .
61 (0x3d	00		Superset information

Notes

1. This is typically programmed as 128 bytes.
2. This is typically programmed as 256 bytes.
3. High order bit defines if assembly requires redundant addressing (if set to 1, redundant addressing is required).
4. From datasheet.
5. High order bit is Self Refresh flag. If set to 1 the assembly supports self refresh.
6. The JEDEC specification states that these bytes are optional for 66 MHz applications. If they are not included then the SPD revision level (byte 62) is set at revision 1 (01h).

Table 8-17. Static SPD SROM Configuration Specifications (continued)

Offset	Value	Field Type	Description
62 (0x3e)	00		SPD revision
63 (0x3f	**xx**		Checksum for bytes 0-62. See Table 8-16 for xx values.
64 (0x40)	FF		Reserved for future expansion
. .	. .	. .	. .
255	FF		

Notes

1. This is typically programmed as 128 bytes.
2. This is typically programmed as 256 bytes.
3. High order bit defines if assembly requires redundant addressing (if set to 1, redundant addressing is required).
4. From datasheet.
5. High order bit is Self Refresh flag. If set to 1 the assembly supports self refresh.
6. The JEDEC specification states that these bytes are optional for 66 MHz applications. If they are not included then the SPD revision level (byte 62) is set at revision 1 (01h).

VPD Revision Data, 0x0f

The VPD revision data packet consists of byte fields that indicate the type, version, and revision of the vital product data. The following table describes the VPD revision data packet.

Table 8-18. VPD Revision Data, 0x0f

Byte Offset	Field Type	Field Description
1	UCHAR	Unique VPD packet identifier must be set to 0x0f
2	UCHAR	Size of the following data section in bytes must be set to 0x04

A product must have exactly one VPD revision packet.

Table 8-18. VPD Revision Data, 0x0f (continued)

3	UCHAR	Board Type: 0x00: processor board or mezzanine 0x01: brainless baseboard 0x02: I/O transition module 0x03: I/O card or mezzanine (IPMC, etc.) 0x04-0xFF: Unused
4	UCHAR	Architecture revision must be set to 0x03
5	UCHAR	Board build revision (starts at zero and increments whenever the VPD data for the board is modified
6	UCHAR	Revision reason flags (these flags are ORed together and are cumulative across revisions 0x00: initial release 0x01: changed boards's artwork identifier revision letter 0x02: added one or more VPD packets 0x04: removed one or more existing VPD packets 0x08: extended one or more existing VPD packets 0x10: changed one or more existing VPD field values 0x20: repaired invalid VPD condition 0x40: unused 0x80: unused
A product must have exactly one VPD revision packet.		

VPD Content Information

The information on the following pages includes content information of the VPD for the initial release of the PrPMC880 and applies to the following boards and components.

Table 8-19. PrPMC880 VPD Models/Part Numbers

Description	Part Number
PrPMC880-2261 256MB DDR SDRAM, Front/Rear gigabit Ethernet and Serial Port	01-W3830F05
PrPMC880-2271 512MB DDR SDRAM, Front/Rear gigabit Ethernet and Serial Port	01-W3830F07
PrPMC880-3261 256MB DDR SDRAM, Dual Rear gigabit Ethernet and Serial Port	01-W3830F06
PrPMC880-3271 512MB DDR SDRAM, Dual Rear gigabit Ethernet and Serial Port	01-W3830F08

vpdGenerateCRC

The EEPROM CRC packet contains the 32-bit CRC value for the VPD SROM block. This packet must be present in a valid PowerPC Vital Product Data block. Note that VPD integers are stored in big-endian format.The following sample code demonstrates the CRC calculation process. Note that the CRC value stored in the CRC VPD packet must be zeroed prior to running the calculation in order to obtain the correct CRC value. The CRC needs to be computed for all bytes contained in the SROM data block. This is the same quantity as is defined by the second element, following the “MOTOROLA” eyecatcher, in the VPD Header Section.

```
/*

* vpdGenerateCRC - generate CRC data for the passed buffer
* description:
*    This function's purpose is to generate the CRC for the
*    passed VPD SROM buffer.
* call:
*    argument #1 = buffer pointer
*    argument #2 = number of elements
* return:
*    CRC data

*/


unsigned int
vpdGenerateCRC(pVpdBuffer, vpdSromSize)
unsigned char *pVpdBuffer;
unsigned int vpdSromSize;
{
   unsigned int crcValue;
   unsigned int crcValueFlipped;
   unsigned char dataByte;
   unsigned int index, dataBitValue, msbDataBitValue;


crcValue = 0xffffffff;
for (index = 0; index < vpdSromSize; index++) {
   dataByte = *pVpdBuffer++;

   for (dataBitValue = 0; dataBitValue < 8; dataBitValue++) {
      msbDataBitValue = (crcValue >> 31) & 1;
      crcValue <<= 1;

      if (msbDataBitValue ^ (dataByte & 1)) {
         crcValue ^= 0x04c11db6;
         crcValue |= 1;
      }
      dataByte >>= 1;
   }
}
crcValueFlipped = 0;
for (index = 0; index < 32; index++) {
```

```
   crcValueFlipped <<= 1;
   dataBitValue = crcValue & 1;
   crcValue >>= 1;
   crcValueFlipped += dataBitValue;
}
crcValue = crcValueFlipped ^ 0xffffffff;

return (crcValue);
}
```

Serial Presence Detect (SPD) Checksum Calculation

The checksum field (byte 63) designates the checksum for checking data integrity (similar to parity) for bytes 0-62. It is written during board production and can be used to verify the data integrity for these bytes.

The process for calculating the checksum includes the following:

1. Convert the binary information in byte locations 0-62 to decimal.
2. Add together (sum) all decimal values for addresses 0-62.
3. Divide the sum by 256.
4. Convert the remainder to binary (will be less than 256).
5. Store the result (single byte) in address 63 as “checksum.”

Note The same result can be obtained by adding the binary values in addresses 0-62 and eliminating all but the low order byte. The low order byte is the checksum.

Calculation of Checksum for the Configuration Information

This field designates the checksum for checking data integrity (similar to parity) for bytes 0 - 62. It is written during module production and can be used to verify the data integrity for these bytes.

Table 8-20. Example of a Checksum Calculation for Bytes 0-62

SPD Byte Address	Serial PD			Convert to Decimal
0	0010	0100		36
1	1111	1110		+254
2	0000	0000		+ 0
3	0000	0000		+ 0
				+ 0
				+ 0
60	0000	0000		+ 0
61	0000	0000		+ 0
62	0000	0000		+ 0
Decimal Total				290
Divide by 256				1
Remainder				34
Convert to binary	0010	0010		34
63	(Checksum) 0010	0010		

Specifications

This appendix provides general specifications including mechanical, environmental, and electrical for the PrPMC880.

Table A-1. PrPMC880 Specifications

Characteristics	Specifications
+3.3Vdc	for MPC7447 @ 1 GHz/133 MHz bus 5.8A, typical
Operating temperature (refer to Cooling Requirements section)	0º to 55º C (32º to131º F) at point of exit of forced air
Storage Temperature	-40º to +85º C (-40º to 185º F)
Relative Humidity	10% to 90% (noncondensing)
Vibration	10.5 Gs RMS (operating) 20-2000 Hz random 6 Gs RMS (non operating) 20-2000 Hz random
Physical Dimensions	Width: 74mm (2.91 inches) Length: 149mm (5.87 inches) Standard: 13.5mm (0.53 inches
MTBF	100,000 hours

PrPMC880 Electrical Characteristics

The PrPMC880 requires only a +3.3V (±5%) voltage input. The estimated 3.3V current draw and power requirements are shown in the following table.

Table A-2. Estimated Power Requirements

Module Configuration	Typical Watts @ +3.3V	Maximum Watts @ +3.3V
133 MHz bus with MPC7447 processor @ 1 GHz	7.5	
9 SDRAM devices	6.3	
6 flash devices	0.2	0.35
Core power supply		
EPLD	0.43	
Ethernet	2.3	
Other components	1	2
Power Consumption Summary	19.3	
+3.3V power consumption is 5.8A typical		

EMC Compliance

The PrPMC880 is an add-on module meant to be used in conjunction with standard VME or CompactPCI baseboard applications. As such, it is the responsibility of the OEM to meet the regulatory guidelines as determined by their application.

The PrPMC880 has been tested in conjunction with a standard MCG baseboard and chassis for CE certification and meets the requirements for

EN55022 Class B equipment. Compliance was achieved under the following conditions:

- ❏ Shielded cables on all external I/O ports.
- ❏ Cable shields connected to earth ground via metal shell connectors bonded to a conductive module front panel.
- ❏ Conductive chassis rails connected to earth ground. This provides the path for connecting shields to earth ground.
- ❏ Front panel screws properly tightened.

For minimum RF emissions, it is essential that the conditions above be implemented. Failure to do so could compromise the EMC compliance of the equipment containing the module.

Thermal Validation

B

Board component temperatures are affected by ambient temperature, air flow, board electrical operation, and software operation. In order to evaluate the thermal performance of a circuit board assembly, it is necessary to test the board under actual operating conditions. These operating conditions vary depending on system design.

While Motorola Computer Group performs thermal analysis in a representative system to verify operation within specified ranges (see Appendix A, *Specifications*), you should evaluate the thermal performance of the board in your application.

This appendix provides systems integrators with information which can be used to conduct thermal evaluations of the board in their specific system configuration. It identifies thermally significant components and lists the corresponding maximum allowable component operating temperatures. It also provides example procedures for component-level temperature measurements.

Thermally Significant Components

The following table summarizes components that exhibit significant temperature rises. These are the components that should be monitored in order to assess thermal performance. The table also supplies the component reference designator and the maximum allowable operating temperature.

Versions of the board that are not fully populated may not contain some of these components.

The preferred measurement location for a component may be *junction*, *case*, or *air* as specified in the table. Junction temperature refers to the temperature measured by an on-chip thermal device. Case temperature

refers to the temperature at the top, center surface of the component. Air temperature refers to the ambient temperature near the component.

Table B-1. Thermally Significant Components

Reference Designator	Generic Description	Max. Allowable Component Temperature (deg. C)	Measurement Location
U81-U82	ER82559 Ethernet Controller	70	Case
U69-U77	SDRAM	70	Case
U40	MPC7447	105	Case
U64	MV64360	120	Case

Component Temperature Measurement

The following sections outline general temperature measurement methods. For the specific types of measurements required for thermal evaluation of this board, see Table B-1.

Preparation

We recommend 40 AWG (American wire gauge) thermocouples for all thermal measurements. Larger gauge thermocouples can wick heat away from the components and disturb air flowing past the board.

Allow the board to reach thermal equilibrium before taking measurements. Most circuit boards will reach thermal equilibrium within 30 minutes. After the warm up period, monitor a small number of components over time to assure that equilibrium has been reached.

Measuring Junction Temperature

Some components have an on-chip thermal measuring device such as a thermal diode. For instructions on measuring temperatures using the on-board device, refer to the *PrPMC880 Processor PMC Programmer's*

Reference Guide and to the component manufacturer's documentation listed in Appendix C, *Related Documentation*.

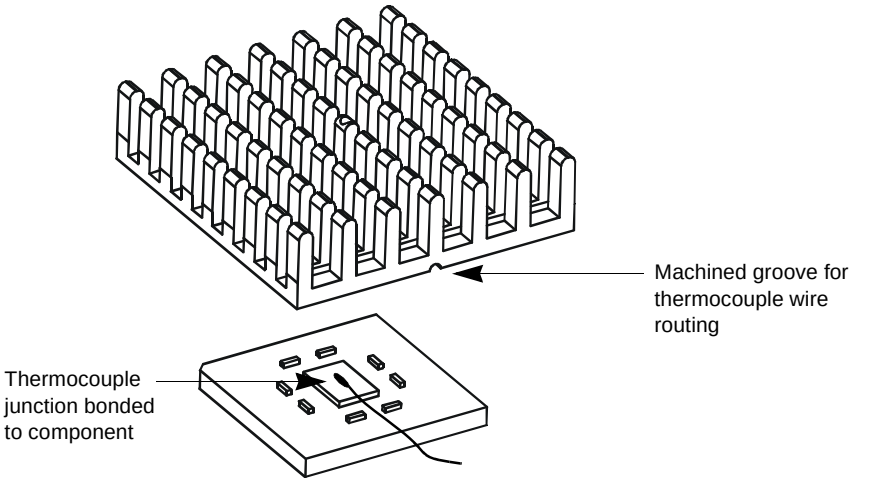
Measuring Case Temperature

Measure the case temperature at the center of the top of the component. Make sure there is good thermal contact between the thermocouple junction and the component. We recommend you use a thermally conductive adhesive such as Loctite 384.

If components are covered by mechanical parts such as heatsinks, you will need to machine these parts to route the thermocouple wire. Make sure that the thermocouple junction contacts *only* the electrical component. Also make sure that heatsinks lay flat on electrical components. The following figure shows one method of machining a heatsink base to provide a thermocouple routing path.

Note Machining a heatsink base reduces the contact area between the heatsink and the electrical component. You can partially compensate for this effect by filling the machined areas with thermal grease. The grease should not contact the thermocouple junction.

B



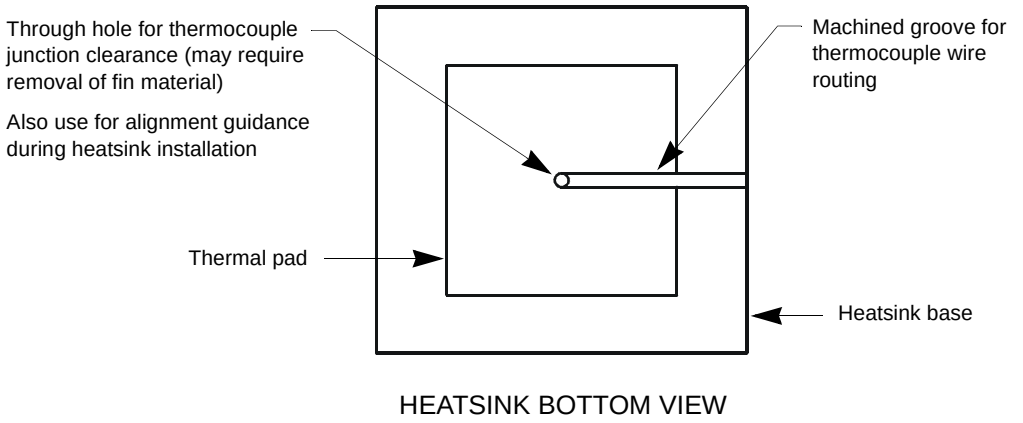


Figure B-1. Mounting a Thermocouple Under a Heatsink

Measuring Local Air Temperature

Measure local component ambient temperature by placing the thermocouple downstream of the component. This method is conservative since it includes heating of the air by the component. The following figure illustrates one method of mounting the thermocouple.

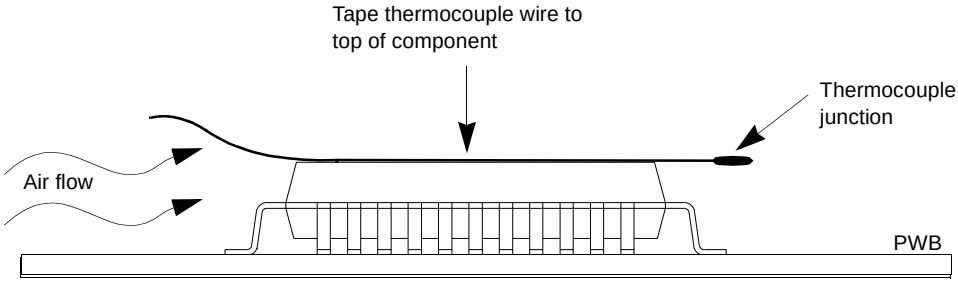


Figure B-2. Measuring Local Air Temperature

Related Documentation

C

Motorola Computer Group Documents

The Motorola publications listed below are referenced in this manual. You can obtain paper or electronic copies of Motorola Computer Group publications by:

- ❏ Contacting your local Motorola sales office
- ❏ Visiting Motorola Computer Group's World Wide Web literature site, http://www.motorola.com/computer/literature

Table C-1. Motorola Computer Group Documents

Document Title	Publication Number
MOTLoad Firmware Package User's Manual	MOTLODA/UM

To obtain the most up-to-date product information in PDF or HTML format, visit http://www.motorola.com/computer/literature

Manufacturers' Documents

The next table lists the manufacturers' data sheets and other useful manuals. Please note that in many cases, the information is preliminary

and the revision levels of the documents are subject to change without notice.

C

Table C-2. Manufacturers' Documents

Document Title and Source	Publication Number
MPC7450 RISC Microprocessor Family User's Manual	MPC7450/UM
MPC7457 RISC Microprocessor Hardware Specifications	MPC7457EC
Programming Environments Manual for 32-Bit Implementations of the PowerPC Architecture	MPCFPE32B/AD
PowerPC Apollo Microprocessor Implementation Definition Book IV Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 Web Site: http://e-www.motorola.com/webapp/sps/library/prod_lib.jsp Email: ldcformotorola@hibbertco.com	Addendum to SC-Vger Book IV Version 1.0
MV64360 System Controller for PowerPC Processors Marvell Corp. http://www.marvell.com/ To register and request a login to their Publications site, visit http://www.marvell.com/login	MV64360 Product Brief MV64360 Data Sheet
ATMEL Nonvolatile Memory Data Book Must request documentation at: http://www.atmel.com/atmel/support/	AT24C04 0180C 0336F
Broadcom http://www.broadcom.com	BCM5421S Product Brief
MAX6900 Real Time Clock — Data Sheet Maxim Corporation; http://www.maxim-ic.com	MAX6900.pdf

Related Specifications

This table lists the product's related specifications. The appropriate source for the listed document is also provided. Please note that in many cases, the information is preliminary and the revision levels of the documents are subject to change without notice.

Table C-3. Related Specifications

Document Title and Source	Publication Number
IEEE http://standards.ieee.org/catalog/	
IEEE - Common Mezzanine Card Specification (CMC) Institute of Electrical and Electronics Engineers, Inc.	P1386 Draft 2.0
IEEE - PCI Mezzanine Card Specification (PMC) Institute of Electrical and Electronics Engineers, Inc.	P1386.1 Draft 2.0
IEEE Standard for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Institute of Electrical and Electronics Engineers, Inc.	IEEE 802.3
PCI Industrial Manufacturers Group (PICMG) http://www.picmg.com/	
CompactPCI Hot Swap Specification, Revision 2.0 Peripheral Component Interconnect (PCI) Local Bus Specification, Revision 2.0, 2.1, 2.2 Embedded PCI-X (ePCI-X) for Standard Form Factor PCI Specification	PICMG 2.1 R2.0 PCI Local Bus Specification PICMG 1.2 R1.0
PCI Special Interest Group (PCI SIG) http://www.pcisig.com/	
Electronic Industries Alliance http://www.eia.org/	

C

Table C-3. Related Specifications (continued)

Document Title and Source	Publication Number
Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange; Electronic Industries Alliance; http://global.ihs.com/index.cfm (for publications)	TIA/EIA-232 Standard
PowerPC Reference Platform (PRP) Specification, Third Edition, Version 1.0, Volumes I and II; International Business Machines Corporation http://www.ibm.com	MPR-PPC-RPU-02
VITA http://www.vita.com/	
Proposed Standard Physical and Environmental Layers for PCI Processor Mezzanine Cards: PPMC	VITA32 2002, Rev. 0.9

Index

D

E

F

G

N

O

P

R

S

T

U

V

INDEX

W

INDEX