ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

Custom engineering
so your equipment works exactly as you specify.

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

Expert team | Trust guarantee | 100% satisfaction

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

Find the *Abaco Systems / SBS VMIVME-7050-100001* at our website: *Click HERE*

VMIVME-7050

PowerPC-Based VMEbus Single-Board Computer

Product Manual

A GE Fanuc Company

12090 South Memorial Parkway
Huntsville, Alabama 35803-3308, USA
(256) 880-0444 ◆ (800) 322-3616 ◆ Fax: (256) 882-0859

500-007050-000 Rev. C

COPYRIGHT AND TRADEMARKS

© Copyright 2004. The information in this document has been carefully checked and is believed to be entirely reliable. While all reasonable efforts to ensure accuracy have been taken in the preparation of this manual, VMIC assumes no responsibility resulting from omissions or errors in this manual, or from the use of information contained herein.

VMIC reserves the right to make any changes, without notice, to this or any of VMIC's products to improve reliability, performance, function, or design.

VMIC does not assume any liability arising out of the application or use of any product or circuit described herein; nor does VMIC convey any license under its patent rights or the rights of others.

For warranty and repair policies, refer to VMIC's Standard Conditions of Sale.

AMXbus, BITMODULE, COSMODULE, DMAbus, IOMax**,** IOWorks Foundation, IOWorks Manager, IOWorks Server, MAGICWARE, MEGAMODULE, PLC ACCELERATOR (ACCELERATION), Quick Link, RTnet, Soft Logic Link, SRTbus, TESTCAL, "The Next Generation PLC", The PLC Connection, TURBOMODULE, UCLIO, UIOD, UPLC, Visual Soft Logic Control(ler), ***VMEaccess*****,** VMEbus Access***, VMEmanager*****, VMEmonitor*****,** VMEnet, VMEnet II, and ***VMEprobe*** are trademarks and The I/O Experts, The I/O Systems Experts, The Soft Logic Experts, and The Total Solutions Provider are service marks of VMIC.



(I/O man figure)



(IOWorks man figure)









The I/O man figure, IOWorks, IOWorks man figure, UIOC, Visual IOWorks and the VMIC logo are registered trademarks of VMIC.

ActiveX, Microsoft, Microsoft Access, MS-DOS, Visual Basic, Visual C++, Win32, Windows, Windows NT, and XENIX are registered trademarks of Microsoft Corporation.

Celeron and MMX are trademarks, and Intel and Pentium are registered trademarks of Intel Corporation.

PICMG and CompactPCI are registered trademarks of PCI Industrial Computer Manufacturers' Group.

Other registered trademarks are the property of their respective owners.

VMIC

All Rights Reserved

This document shall not be duplicated, nor its contents used for any purpose, unless granted express written permission from VMIC.

Table of Contents

Overview 1 3

PowerPC RISC Microprocessor 15
L1 Data Cache 15
L2 Cache 15
Bus Pipelining 15
Marvell MV64360 System Controller 16
Ethernet Controllers 17
Gigabit Ethernet Switch 17
Serial Port (COM1 and COM2) 17
References 19
Safety Summary 21
Warnings, Cautions and Notes 22
Notation and Terminology 23

Chapter 1 - Installation and Setup 25

Unpacking Procedures 26
Hardware Setup 27
VMEbus Switch S1 30
Non-Volatile Memory Write Protection 30
Installation 32
Installing PMC cards 33
PMC Sites (VIO) 33
Front/Rear Panel Connectors 35
LED Definition 36
RS-232/422/485 Serial Port (COM1) 37
Using the Serial Port (Debug) 41
PMC Connector Pinouts 42
PMC Site#1 (J11) Connector and Pinout 42
PMC Site#1 (J12) Connector and Pinout 43

PMC Site#1 (J13) Connector and Pinout 44
PMC Site#1 (J14) Connector and Pinout 45
PMC Site#2 (J21) Connector and Pinout 46
PMC Site#2 (J22) Connector and Pinout 47
PMC Site#2 (J23) Connector and Pinout 48
PMC Site#2 (J24) Connector and Pinout 49
Serial Connector and Pinout 50
Ethernet Connector Pinout (J1 and J2) 51
Optional P0 Connector Pinout 52
VMEbus Connector Pinout 53

Chapter 2 - PowerPC Programming 55

Physical Memory 56
PowerPC Registers and Programming Model 57
Instruction Set 58
PowerPC Instruction Set 59
PowerPC Instruction Set (Continued) 60
On-Chip Cache Implementation 61
PowerPC Cache Model 61
PowerPC 750FX/GX Microprocessor Cache Implementation 61
Description of 750FX/GX L1 Caches 61
Description of 750FX/GX L2 Cache 62
L2 Cache Operation 62
Exception Model 64
PowerPC Exception Model 64
Microprocessor Exception Implementation 65
Memory Management 68
PowerPC Memory Management Model 68
PowerPC 750FX/GX Microprocessor Memory Management Implementation 69
Instruction Timing 70
Power Management 72
Full-On 72
Doze 73
Nap 73
Sleep 73
Thermal Management 74
Performance Monitor 75
BootROM Monitor (PMON) 76
PMON/2000 On-line Documentation 76
PMON R01.00 Readme.txt 76
Requirements 76

Requirements (Continued) 77
Flash Device Selection 77
Setting The IP Address For gtx0/NIC0 77
Loading an Image on the Soldered-Down Flash 77
PMON Startup Screens (For Reference Only) 78
PMON Commands by Catagory 80
Example of using PMON for Assembly-level debugging 81
Example: Set Terminal Options 85
stty 85
Format 85
Functional Description 86
Example: Write/Erase Flash Memory 87
Flash 87
Format 87
Flash the 64 MB part using PMON: 87
Release Notes 89
What's New 89
VMIVME-7050 89
Notes/ Errata 90
PMON R01.00 Release Notes - End 90

Maintenance 91

Maintenance Prints 92

List of Figures

Figure 1 VMIVME-7050 Block Diagram 18
Figure 1-1 VMIVME-7050 Board Layout 28
Figure 1-2 VMIVME-7050 Front Panels 29
Figure 1-3 Installing PMC Cards on the VMIVME-7050 34
Figure 1-4 Front Panel LED Positions 35
Figure 1-5 Example: Desktop Windows-Based OS 37
Figure 1-6 Hyperterminal New Connection Window 38
Figure 1-7 COM Port Window 39
Figure 1-8 COM1 Property Window 40
Figure 1-9 VMIVME-7050 Serial Port Connection 41
Figure 1-10 Serial Connector Pinouts 50
Figure 1-11 Ethernet Connector and Pinout 51
Figure 1-12 P0 Connector and Pinout 52
Figure 1-13 VMEbus Connector Diagram 53
Figure 2-1 PowerPC Instruction Set 58
Figure 2-2 750FX/GX Power States 72

List of Tables

Table 1-1	CPU Board Connectors and Switches	30
Table 1-2	VMEbus Switch S1 Configuration	30
Table 1-3	PMC Site#1 (J11) Connector Pinout	42
Table 1-4	PMC Site#1 (J12) Connector Pinout	43
Table 1-5	PMC Site#1 (J13) Connector Pinout	44
Table 1-6	PMC Site#1 (J14) Connector Pinout	45
Table 1-7	PMC Site#1 (J21) Connector Pinout	46
Table 1-8	PMC Site#1 (J22) Connector Pinout	47
Table 1-9	PMC Site#2 (J23) Connector Pinout	48
Table 1-10	PMC Site#1 (J24) Connector Pinout	49
Table 1-11	VMEbus Connector Pinout	53
Table 2-1	Microprocessor Exception Classifications	66
Table 2-2	Exceptions and Conditions	66
Table 2-3	Physical Address Map	79

Overview

Introduction

The VMIVME-7050 is the flagship of VMIC's PowerPC-based VMEbus line of SBCs. The following features represent a robust product in a single-slot VMEbus form factor. The VMIVME-7050 is compliant with the VMEbus Specification Rev. C.1 and features a PCI-to-VMEbus bridge, allowing the board to function in multi-CPU systems.

The VMIVME-7050 provides features typically found on desktop systems, such as:

- High Performance PowerPC-based single-board computer (SBC)
- IBM 750FX/GX PowerPC Processor (733 MHz to 1.0 GHz)
- Marvell MV64360 system controller with 2 MB integrated SRAM
- Dual 64-bit PCI-X on PMC expansion sites (backward compatible) with support for double-wide PMC card

Additional features include:

- Up to 2 GB of DDR266 SDRAM, PC2100 (without losing any PMC expansion capabilities)
- 1MB and 512 KB on-chip L2 cache
- 64.5 MB total Bootable Flash (64 Mbyte soldered down, 512 KB in PLCC32 socket)
- Up to 1 GB of bootable CompactFlash (optional)
- Ultra-DMA IDE Controller to CompactFlash Socket (Type 1 or 2)
- 32 KB NVRAM
- Real-time clock
- Built-in self-test
- Two high-performance (up to 10Mb/s per channel) 16550-compatible serial ports (supporting synch/asynch RS-232, RS-422 and RS-485)
- Dual 10/100/1000 Ethernet (two RJ-45 connectors on the front panel , two out of the P2 connector)
- Four 32-bit programmable timers
- One 32-bit Watchdog Timer (dual stage for NMI and board RESET)

Additional features include (continued)

- Full 64-pin PMC I/O support for both PMC sites through VMEbus P0 & P2 (P0 offers controlled-impedance for high-speed connectivity such as Firewire and Ultra320 SCSI)
- 64-bit PCI board-to-board connector for PMC expansion
- On-board monitoring of temperature and voltages
- Optional 1101.10 front panel

Additionally, the VMIVME-7050 supports IEEE P1386 Common Mezzanine Card specification with two mezzanine card expansion sites. Two single or one "Double-wide" PMC module can be accommodated. Both sites automatically support every possible PMC device configuration from "5V I/O, 32-bit/33 MHz" through "3.3V I/O 64-bit/133MHz PCI-X." Proper VIO and bus capabilities are automatically set.

Support for high-speed I/O is accomplished by routing all 64 pins of PMC#1's J14 to the VMEbus P0 in "P4V0-64" fashion ("P4V0-64" is a mnemonic used in the ANSI/VITA-35-2000 specification to show a compliant PMC-P4 to VME-P0 mapping on the host). Unlike the standard VME P1/P2 open-wire connectors, P0 is a high-density, high-quality, impedance-controlled connector which maintains the integrity of extended bandwidth signals. Ultra320 differential SCSI, firewire and Gbit Ethernet are just a few examples of the interfaces that can benefit from using this high-quality pathway. Since this pathway is simply PMC#1's I/O channel, changing high speed interfaces is as easy as changing out a PMC card.

The VMIVME-7050 is suitable for use in a variety of applications, such as: telecommunications, simulation, instrumentation, industrial control, process control and monitoring, factory automation, automated test systems, data acquisition systems and anywhere that the higher performance processor in a single VMEbus slot is desired.

PowerPC RISC Microprocessor

The microprocessor used on the VMIVME-7050 is a 32-bit PowerPC 750FX/GX architecture in a 0.13 micron CMOS technology with six levels of copper interconnect. The performance enhancements included in the microprocessor are:

- On-board L2 cache with locking by way (512KB to 1MB)
- L1 data cache availability
- Bus pipelining (consecutive data reads)
- Additional floating point unit (FPU) reservation station and improved reciprocal estimates

L1 Data Cache

The 750FX/GX L1 data cache supports miss-under-miss access, meaning that with one miss outstanding, the cache can continue to be accessed until a second miss occurs. Additionally, the 750FX L1 data cache allows the second miss to initiate a transaction in the bus interface unit, while the first miss is pending. In previous PowerPC 750 microprocessor implementations the data bus width for bus interface unit (BIU) accesses of the L1 data cache array was 64 bits. To cast out or to reload a 256-bit cache line required four access cycles. On the 750FX/GX, this bus has been expanded to 256 bits. As a result, cache line data bursts can be read from or written to the cache array in a single cycle, reducing cache contention between the BIU and the load-store unit.

L2 Cache

The 750FX/GX processor has an internal L2 cache capacity of 512 Kbyte. The cache is two-way set associative; each way contains 4096 blocks and each block consists of two 32-byte sectors. Array read and write operations execute in one processor cycle. Writes to the array are 64 bits wide, while reads are 256 bits wide.

In addition, the L2 cache has an 8-bit ECC for every 64-bit word in memory that can be used to correct a majority of single bit errors and detect multiple bit errors. The L2 tags also support parity and locking by way.

Bus Pipelining

The 60x bus has decoupled address and data buses, allowing transactions to be pipelined on the bus. That is, the address operation for a second transaction can be initiated before the data operation for a previous transaction is complete. The 750FX/GX BIU allows pipelined read transactions to a depth of two back-to-back reads.

Marvell MV64360 System Controller

The VMIVME-7050 incorporates the Marvell MV64360 PowerPC System Controller. The MV64360 has two 64-bit PCI / PCI-X buses, dual Gigabit Ethernet LAN controllers, four 32-bit Timers / Counters and one Watchdog Timer. The system controller has 2 Mbits of general purpose SRAM on-board, accessible from the CPU or any of the other interfaces. The SRAM can be used for off loading DRAM traffic, and as a fast CPU access memory (6-cycle latency).

The MV64360 supports PowerPC CPUs with a maximum frequency of 133 MHz, with a transfer rate in excess of 8 Gbps.

The MV64360 also features two Multi-Protocol Serial Controllers (MPSCs), that support synchronous, asynchronous RS-232 and RS-422 / 485 protocols.

Each serial channel has an independent SDMA to support baud rates of up to 10 Mbps / channel (RS-232 @ 230 Kbps).

Ethernet Controllers

The VMIVME-7050 implements two Gigabit Ethernet controllers operating at 10, 100 or 1000 Mbps. The Ethernet controllers are software configurable. The port includes an IEEE 802.3 compliant Media Access Controller (MAC), supporting GMII with an external PHY device. The port speed, duplex and 802.3 flow control can be auto-negotiated, according to IEEE standards 802.3U and 802.3x. Backpressure is supported for half-duplex mode when operating at 10 or 100 Mbyte speeds.

Gigabit Ethernet Switch

The VMIVME-7050 uses two Gigabit LAN switches. The device multiplexes the Gigabit transceiver to one of two corresponding RJ-45 connectors (front and rear).The PHY poles the front panel RJ-45 connection for a link and if none is established within two seconds, it switches to the rear panel to pole for two seconds. Each LAN channel is independent of the other. As soon as a link is established, that channel stays locked. If connections are made to both front and rear LAN connections, there is no guarantee as to which will “win” the locked selection.

Serial Port (COM1 and COM2)

One serial port (COM1) is available as a “debug” port for remote console (async RS-232 only) at the front panel through an RJ-45 connector, and all supported modes at the rear P2 connector. The second port (COM2) supports all listed serial modes and is available through the rear I/O P2 connector only.

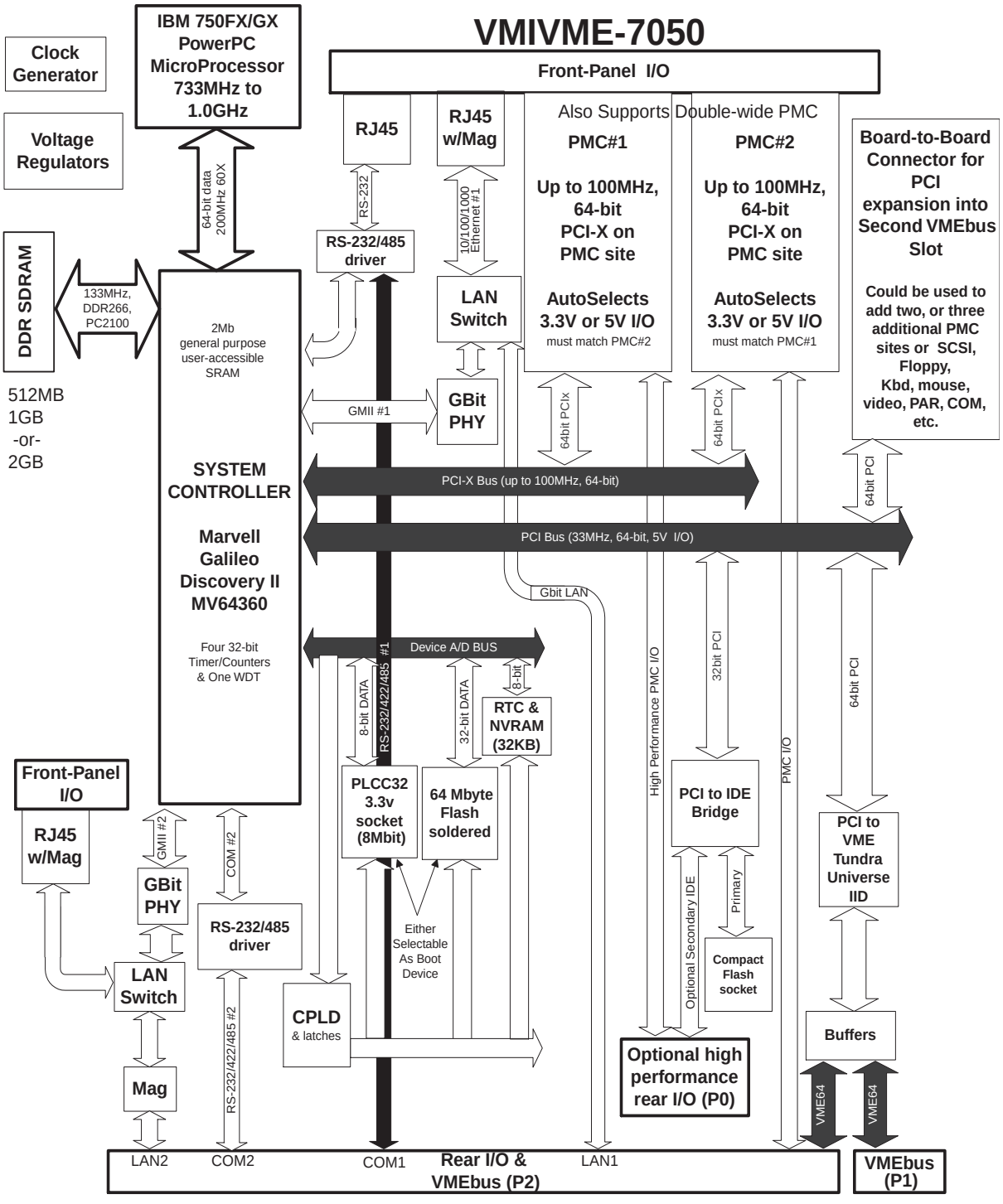


Figure 1 VMIVME-7050 Block Diagram

References

IBM PowerPC 750FX/GX RISC Microprocessor
IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351
www.ibm.com or
www-3.ibm.com/chips/

Marvell MV64360/1/2 System Controller for PowerPC
Marvell Semiconductor
700 First Ave.
Sunnyvale, CA 94089
(408) 222-2500
www.marvell.com

BroadCom BCM5421/5461 10/100/1000BaseT Gigabit Copper Transceiver
BroadCom Corporation
16215 Alton Pkwy
P.O. Box 57013
Irvine, CA 92619-7013
(949) 450-8700
FAX: 949-450-8710
www.broadcom.com

CMD PCI-649 Bus Master Ultra DMA PCI-IDE/ATA Chip
CMD Technology, Inc
5901 Jeronimo Rd
Irvine, CA 92618
(949) 454-0800
www.cmd.com

Tundra Universe II VME-to-PCIbus Bridge
Doc. No. 500-000211-000
VMIC
12090 South Memorial Pkwy.
Huntsville, AL 35803-3308
(800) 322-3616
www.vmic.com

Pericom PI3L301D 16- to 8-bit, Mux/Demux Gigabit Ethernet LAN Switch
Pericom Semiconductor Corporation
2380 Dering Dr.
San Jose, CA 95131
(800) 435-2336
FAX: 408-435-1100
www.pericom.com

PCI Local Bus Specification, Rev. 2.1

PCI Special Interest Group
P.O. Box 14070
Portland, OR 97214
(800) 433-5177 (U.S.)
(503) 797-4207 (International)
(503) 234-6762 (FAX)

CMC Specification, P1386/Draft 2.0 from:

IEEE Standards Department
Copyrights and Permissions
445 Hoes Lanes, P.O. Box 1331
Piscataway, NJ 08855-1331, USA

PMC Specification, P1386.1/Draft 2.0 from:

IEEE Standards Department
Copyrights and Permissions
445 Hoes Lanes, P.O. Box 1331
Piscataway, NJ 08855-1331, USA

VMISFT-9420 IOWorks Access User's Guide

Doc. No. 520-009420-910
VMIC
12090 South Memorial Pkwy.
Huntsville, AL 35803-3308
(800) 322-3616
www.vmic.com

For a detailed description and specification of the VMEbus, please refer to:

VMEbus Specification Rev. C. and the VMEbus Handbook

VMEbus International Trade Assoc. (VITA)
7825 East Gelding Dr.
Suite 104
Scottsdale, AZ 85260
(602) 951-8866
(602) 951-0720 (FAX)
www.vita.com

Safety Summary

The following general safety precautions must be observed during all phases of the operation, service and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture and intended use of this product.

VMIC assumes no liability for the customer's failure to comply with these requirements.

Ground the System

To minimize shock hazard, the chassis and system cabinet must be connected to an electrical ground. A three-conductor AC power cable should be used. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet.

Do Not Operate in an Explosive Atmosphere

Do not operate the system in the presence of flammable gases or fumes. Operation of any electrical system in such an environment constitutes a definite safety hazard.

Keep Away from Live Circuits

Operating personnel must not remove product covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

Do Not Service or Adjust Alone

Do not attempt internal service or adjustment unless another person capable of rendering first aid and resuscitation is present.

Do Not Substitute Parts or Modify System

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to VMIC for service and repair to ensure that safety features are maintained.

Dangerous Procedure Warnings

Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

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

Warnings, Cautions and Notes

STOP informs the operator that a practice or procedure should not be performed. Actions could result in injury or death to personnel, or could result in damage to or destruction of part or all of the system.

WARNING denotes a hazard. It calls attention to a procedure, practice or condition, which, if not correctly performed or adhered to, could result in injury or death to personnel.

CAUTION denotes a hazard. It calls attention to an operating procedure, practice or condition, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the system.

NOTE denotes important information. It calls attention to a procedure, practice or condition which is essential to highlight.

Notation and Terminology

This product bridges the traditionally divergent worlds of Intel-based PC's and Motorola-based VMEbus controllers; therefore, some confusion over "conventional" notation and terminology may exist. Every effort has been made to make this manual consistent by adhering to conventions typical for the Motorola/VMEbus world; nevertheless, users in both camps should review the following notes:

- Hexadecimal numbers are listed Motorola-style, prefixed with a dollar sign: $F79, for example. By contrast, this same number would be signified 0F79H according to the Intel convention, or 0xF79 by many programmers. Less common are forms such as $F79_h$ or the mathematician's $F79_{16}$.
- An 8-bit quantity is termed a "byte," a 16-bit quantity is termed a "word," and a 32-bit quantity is termed a "longword." The Intel convention is similar, although their 32-bit quantity is more often called a "doubleword."
- Motorola programmers should note that Intel processors have an I/O bus that is completely independent from the memory bus. Every effort has been made in the manual to clarify this by referring to registers and logical entities in I/O space by prefixing I/O addresses as such. Thus, a register at "I/O $140" is not the same as a register at "$140," since the latter is on the memory bus while the former is on the I/O bus.
- Intel programmers should note that addresses are listed in this manual using a linear, "flat-memory" model rather than the old segment:offset model associated with Intel Real Mode programming. Thus, a ROM chip at a segment:offset address of C000:0 will be listed in this manual as being at address $C0000. For reference, here are some quick conversion formulas:

Segment:Offset to Linear Address

Linear Address = (Segment × 16) + Offset

Linear Address to Segment:Offset

Segment = ((Linear Address ÷ 65536) – *remainder*) × 4096

Offset = *remainder* × 65536

Where *remainder* = the fractional part of (Linear Address ÷ 65536)

Note that there are many possible segment:offset addresses for a single location. The formula above will provide a unique segment:offset address by forcing the segment to an even 64 Kbyte boundary, for example, $C000, $E000, etc. When using this formula, make sure to round the offset calculation properly!

CHAPTER
1

Installation and Setup

Contents

Unpacking Procedures 26
Hardware Setup 27
Installation 32
Installing PMC cards 33
Front / Rear Panel Connectors 35
RS-232 / 422 / 485 Serial Port (COM1) 37
PMC Connector Pinouts 42
Serial Connector and Pinout 50
Ethernet Connector Pinout (J1 and J2) 51
Optional P0 Connector Pinout 52
VMEbus Connector Pinout 53

Introduction

This chapter describes the hardware jumper settings, connector descriptions, installation, system setup and operation of the VMIVME-7050.

Unpacking Procedures

Any precautions found in the shipping container should be observed. All items should be carefully unpacked and thoroughly inspected for damage that might have occurred during shipment. The board(s) should be checked for broken components, damaged printed circuit board(s), heat damage and other visible contamination. All claims arising from shipping damage should be filed with the carrier and a complete report sent to Customer Care along with a request for advice concerning the disposition of the damaged item(s).

CAUTION: Some of the components assembled on the products may be sensitive to electrostatic discharge and damage may occur on boards that are subjected to a high energy electrostatic field. When the board is placed on a bench for configuring, etc., it is suggested that conductive material be inserted under the board to provide a conductive shunt. Unused boards should be stored in the same protective boxes in which they were shipped.

Hardware Setup

The VMIVME-7050 is factory populated with user-specified options as part of the VMIVME-7050 ordering information. The CPU speed, RAM size and flash memory size are not user-upgradable. To change CPU speeds or RAM/Flash size, contact Customer Care to receive a Return Material Authorization (RMA).

Customer Care is available at: 1-800-240-7782.

Or E-mail us at **support.embeddedsystems@gefanuc.com**.

The VMIVME-7050 is tested for system operation and shipped with factory-installed header jumpers. The physical location of the switches and connectors for the single board computer are illustrated in Figure 1-1 on page 28. The definitions of the switches and connectors are included in Table 1-1.

CAUTION: All jumpers marked factory configured should not be modified by the user.

Care must be taken when making jumper modifications to ensure against improper settings or connections. Improper settings may result in damage to the unit.

Modifying any jumper not marked “User Configurable” will void the Warranty and may damage the unit.

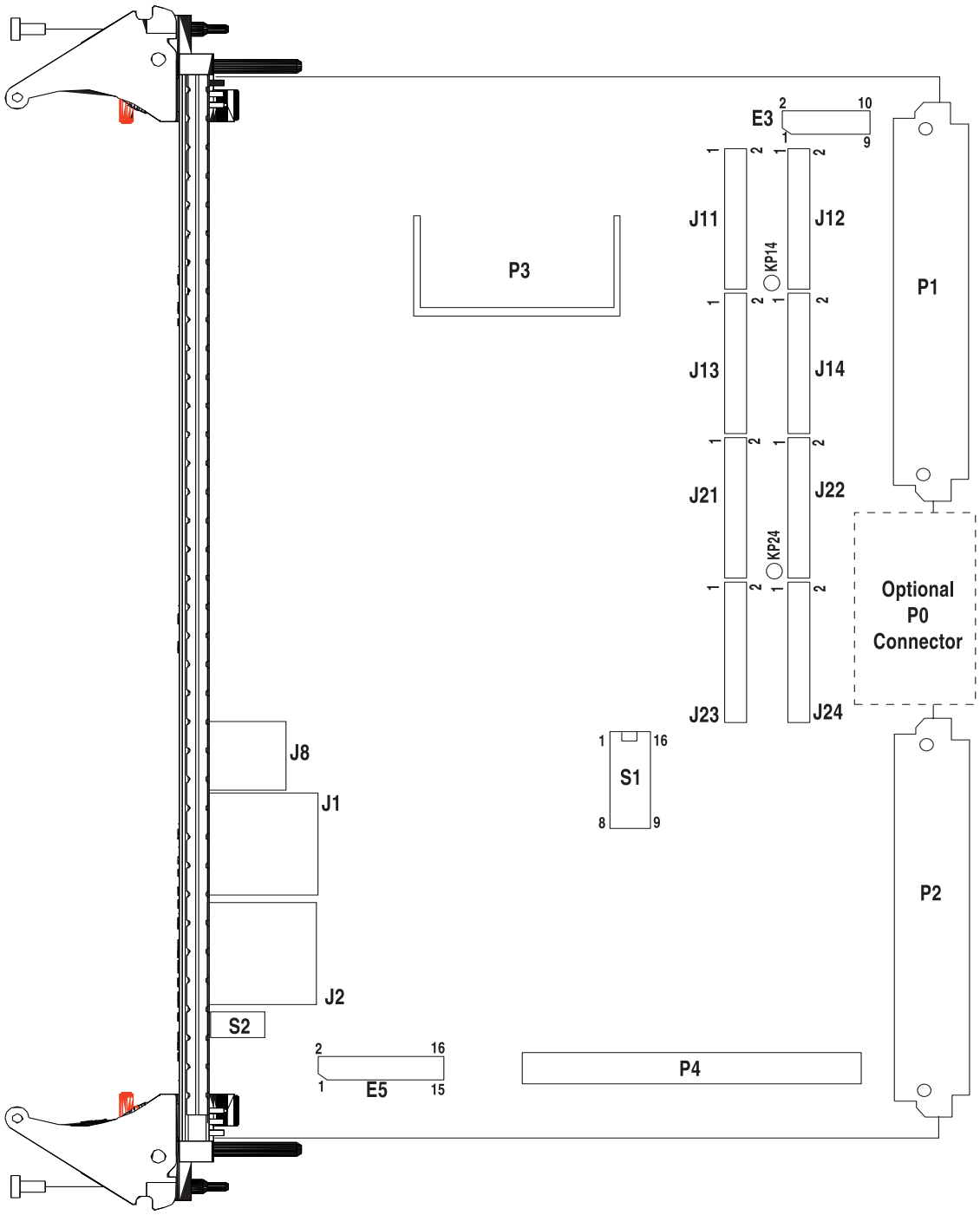


Figure 1-1 VMIVME-7050 Board Layout

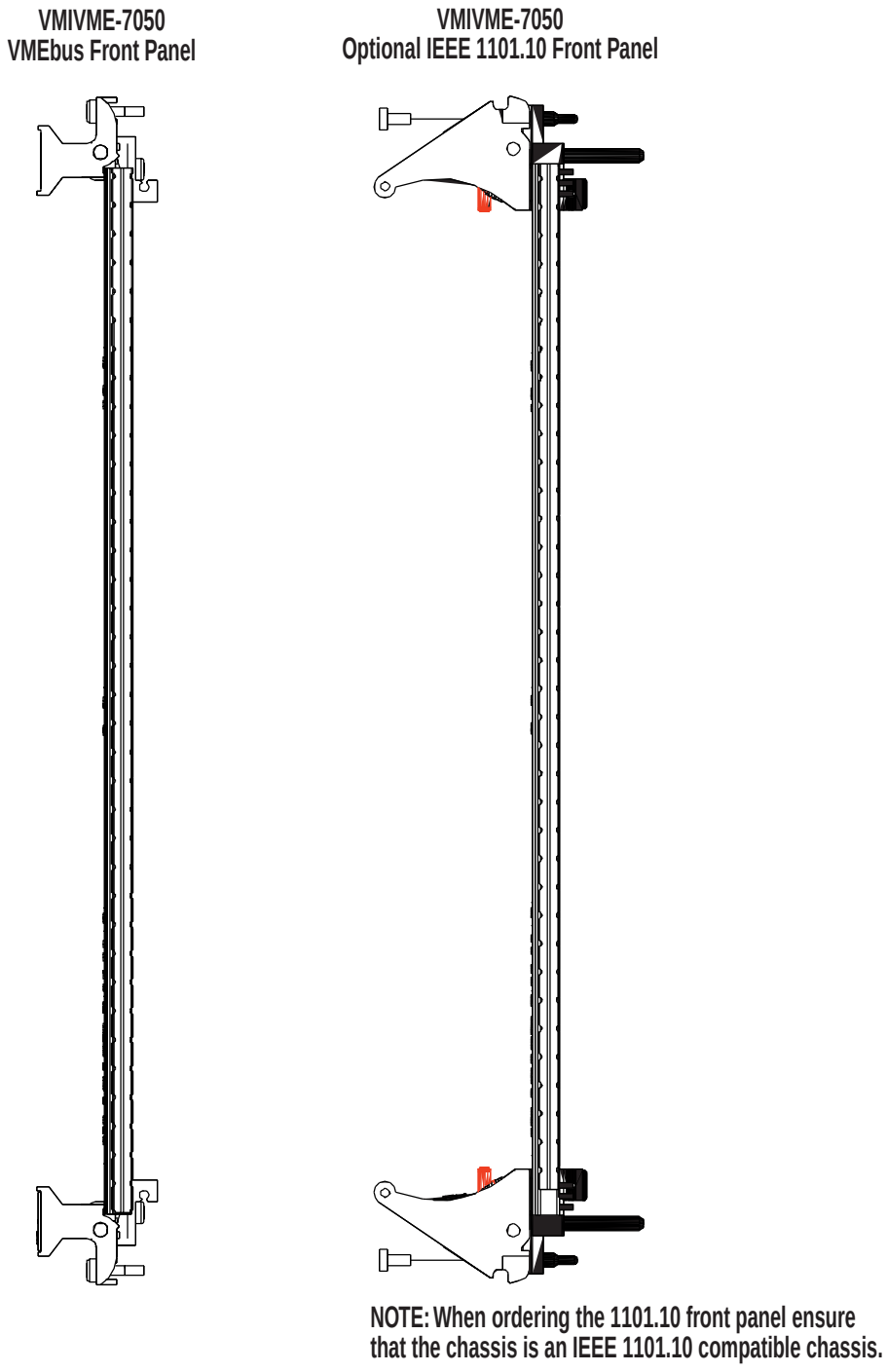


Figure 1-2 VMIVME-7050 Front Panels

Table 1-1 CPU Board Connectors and Switches

Connector	Function
E3	CPLD Program Header
E5	JTAG Interface Header (RTC calibration, E5 pin 14)
S1	VMEbus Switch
S2	Front Panel Reset Switch
J8	COM1 (Debug)
J1	LAN 1
J2	LAN 0
J11, J12, J13, J14	PMC Site#1 Connectors
J21, J22, J23, J24	PMC Site#2 Connectors
P1, P2, (optional P0)	VMEbus Backplane Connectors
P3	IDE CompactFlash
P4	PCI Board-to-Board Connector
KP14	PMC Site#1 Keypin (5V VIO)
KP24	PMC Site#2 Keypin (5V VIO)

VMEbus Switch S1

Table 1-2 VMEbus Switch S1 Configuration (Factory Default is “OFF” for all Positions)

Pin#	Position	Function
1	On	Boot from solder-down Flash (64MB)
	Off	Boot from PLCC32
2	On	Write protect I^2C EPROM (U78)
3	On	Write protect all other non-volatile memory
4	On	WDE# Resets the board
5	On	COM Port 0 (COM0) RS-422/485 mode
	Off	COM Port 0 (COM0) RS-232 mode
6	On	COM0 loopback Check
	Off	COM0 normal Operation
7	On	COM Port 1 (COM1) RS422/485 mode
	Off	COM Port 1 (COM1) RS-232 mode
8	On	COM1 loopback
	Off	COM1 normal

Non-Volatile Memory Write Protection

Users of the VMIVME-7050 can write protect the non-volatile memory by means of a hardware switch. This is a hardware function used to prevent the storage of sensitive data into non-volatile locations. This function cannot be overridden by software. Write protection can be switched ON or OFF by using switch S1 positions 2 and 3. With S1 position 2 in the OFF position, any writes to the I^2C EPROM will be denied. Switch S1 position 3 denies writes to the three other non-volatile locations, including the PLCC32 socketed flash (U28), two soldered-down 64MB TSOP flashes (U51 and

U52) and the real-time clock's NVRAM (U59). If write access to any of the devices is needed, the user can simply move the switch to the OFF position, complete the writes, and if desired move the switch to the ON position to enable write protection when finished (all without powering-down the VMIVME-7050). See Table 1-2 on page 30 above for switch S1 configuration and Figure 1-1 on page 28 for switch location.

Installation

The VMIVME-7050 conforms to the VMEbus physical specification for a single slot 6U Eurocard (dual height). It can be plugged directly into any standard chassis accepting this type of board.

CAUTION: Do not install or remove the board while power is applied.

The following steps describe the recommended method for the VMIVME-7050 installation and power-up:

1. Make sure power to the equipment is off.
2. Choose chassis slot. The VMIVME-7050 can be installed as a slot one controller, or installed in any VMEbus backplane slot.

NOTE: If the VMIVME-7050 has the P0 option, ensure that the chassis is a P0 compatible chassis.

NOTE: The VMIVME-7050 requires forced air cooling Air flow requirement as measured at output side of heatsink is to be greater than 200LFM.

It is advisable to install blank panels over any exposed VMEbus slots. This will allow for better air flow over the VMIVME-7050.

3. Install all PMC cards before placing the VMIVME-7050 in the VMEbus chassis. (Refer to *Installing PMC cards* on page 33 for the proper installation.) (See "PMC Sites (VIO)" on page 33.)
4. Insert the VMIVME-7050 into the chosen VMEbus chassis slot. While ensuring that the board is properly aligned and oriented in the supporting board guides, slide the board smoothly forward against the mating connector until firmly seated.
5. Connect all needed peripherals to the front panel. Each connector is clearly labeled on the front panel. Minimally, an RS-232 connection to another PC is required if the user has not previously configured the system. See "VMIVME-7050 Serial Port Connection" on page 41.
6. Apply power to the system. Several messages are displayed on the screen, including names, versions and copyright dates for PMON.

Installing PMC cards

The VMIVME-7050 supports the IEEE P1386 common Mezzanine Card specification with two Mezzanine Card expansion sites. Two single, or one “Double-wide” PMC card can be accommodated. Both sites automatically support every possible PMC device configuration from “5V I/O, 32-bit/33 MHz” through “3.3V I/O 64-bit/133MHz PCI-X.” The proper VIO and bus capabilities are automatically set.

Support for high-speed I/O is accomplished by routing all 64 pins of PMC#1’s J4 to the VMEbus P0 in “P4V0-64” fashion (“P4V0-64” is a mnemonic used in the ANSI/VITA-35-2000 specification to show a compliant PMC-P4 to VME-P0 mapping on the host). Unlike the standard VME P1/P2 open-wire connectors, P0 is a high-density, high-quality, impedance-controlled connector which maintains the integrity of extended bandwidth signals. Ultra320 differential SCSI, firewire and Gbit Ethernet are just a few examples of the interfaces that can benefit from using this high-quality pathway. Since this pathway is simply PMC#1’s I/O channel, changing high speed interfaces is as easy as changing out a PMC card.

PMC Sites (VIO)

VIO is the I/O voltage used for PCI bus termination. The PCI bus interface requires strong parallel (pull-up) termination. Initially, the voltage for the pull-ups was +5V. Over time, with the need for power reduction and higher speeds, +3V devices became prevalent. +3V devices can easily drive TTL signaling. Higher speed PCI buses utilize +3V termination.

KP14 is PMC#1's VIO keypin position (removed for 3.3V VIO).

KP24 is PMC#2's VIO keypin position (removed for 3.3V VIO).

Since PMC#1 and #2 are on the same PCI bus, they must utilize the same VIO. Therefore, make sure to properly mount the keypins into the proper holes. If the VIO keypin position for both PMC sites are not properly matched, then the keypin position detection circuitry will apply +3.3V to VIO. The keypin is installed for +5.0V and removed for +3.3V.

NOTE: For "Px_M66ENA".... per the PCI specification, 66MHz PCI is only allowed on +3.3V VIO systems. Therefore, if the keypins are loaded such that +5V VIO is enabled, 66MHz is not allowed.

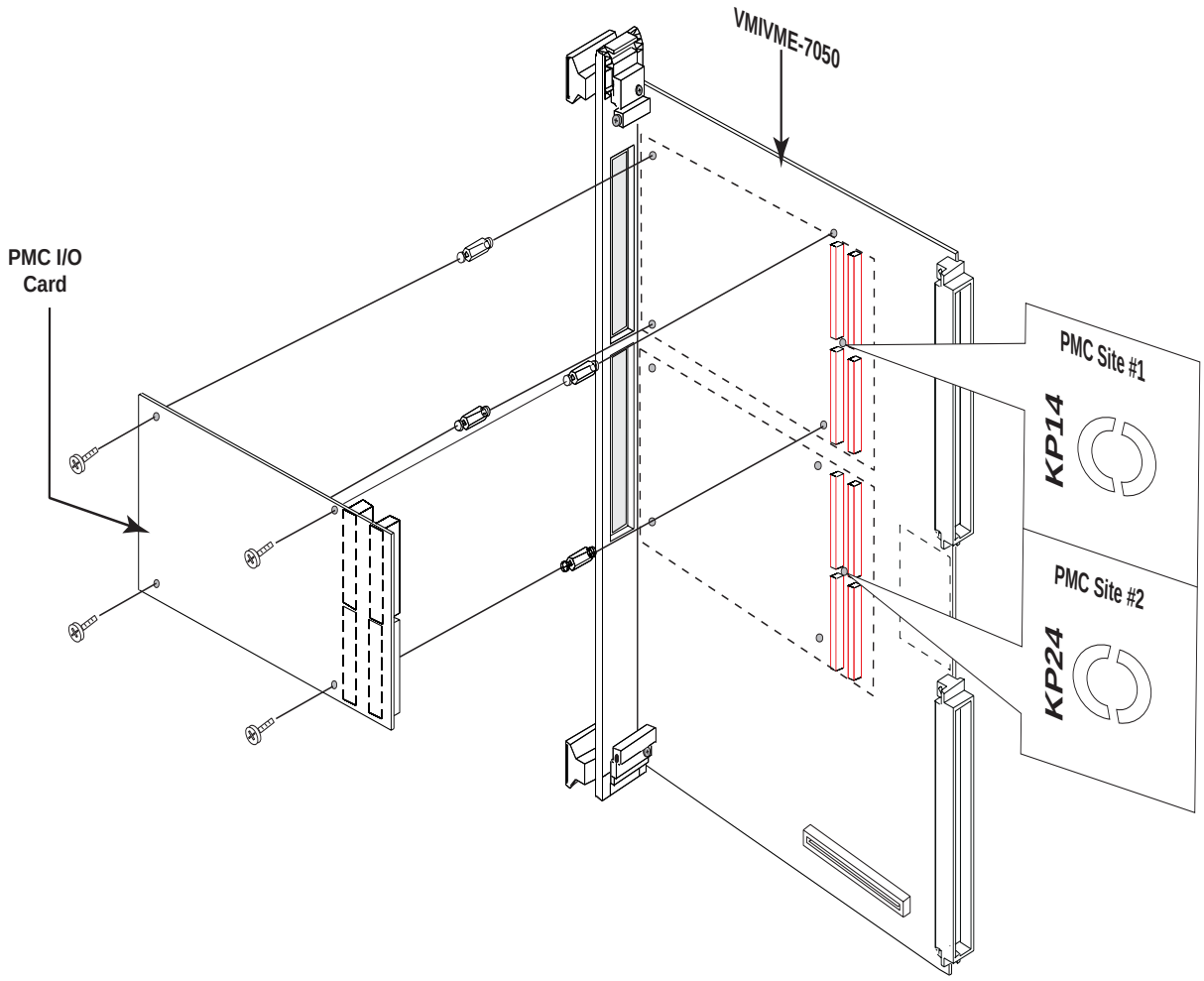


Figure 1-3 Installing PMC Cards on the VMIVME-7050

Front/Rear Panel Connectors

The VMIVME-7050 provides front-panel access to two PMC expansion sites, two Ethernet ports, both are 10/100/1000 Mbit RJ-45 connectors, the reset switch, COM 1 and the status LEDs. A drawing of the VMIVME-7050 front-panel is shown in Figure 1-4 on page 35. The front-panel connectors and indicators are labeled as follows:

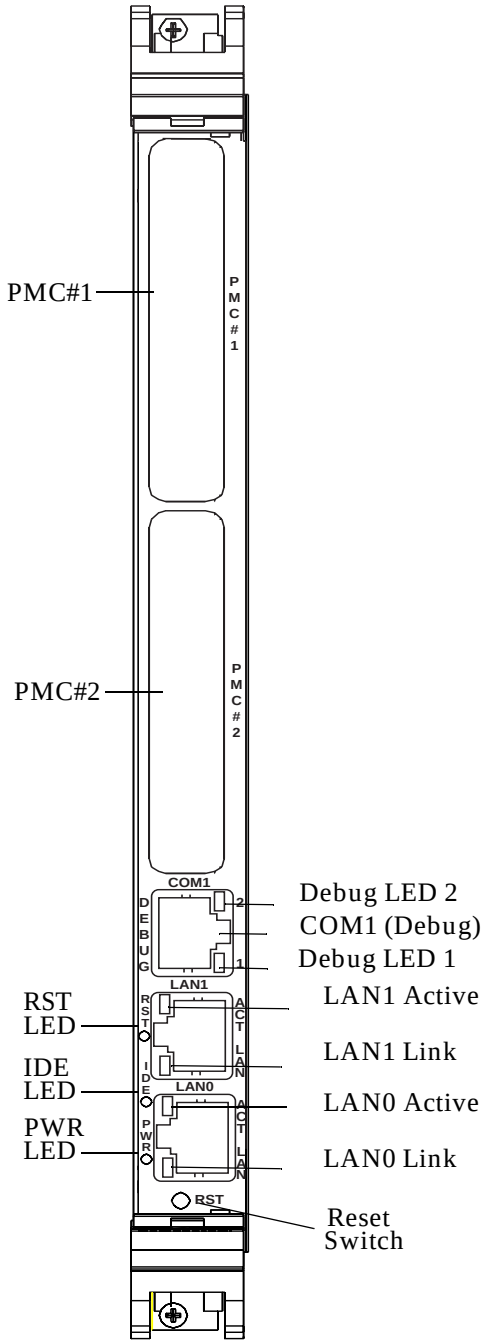
- LAN 0 — 10/100/1000 Mbit Ethernet connector
- LAN 1 — 10/100/1000 Mbit Ethernet connector
- RST — Manual reset switch
- COM 1 — COM port 1 (Debug)
- PMC#1 and #2 — PMC sites #1 and #2
- RST — Reset LED
- IDE — IDE Activity LED
- PWR — Power status LED

The VMIVME-7050 provides rear I/O support for two COM ports, two Gigabit Ethernet ports, two PMC I/O ports and one secondary IDE interface. These signals are accessed using a rear-panel transition board (the VMIACC-7050, 7050A, 7050B and the 7050C), which terminates into industry standard connectors.

The front panel connectors, including connector pinouts and orientation, for the VMIVME-7050 are described later in this chapter. Rear I/O panel connections are defined in the appropiate rear I/O transition utility board Installation Guide. See the VMIVME-7050 product specification for the VMIACC-7050 compatible rear panel transistion utility board offered by VMIC.

Figure 1-4 Front Panel LED Positions

LED Definition



RST	*Reset Switch* - Allows the system to be reset from the front panel.
RST LED	*Reset LED* - Indicates when the board is in reset (Red LED).
PWR LED	*Power* - Indicates when +5v power is applied to the board, (Green LED).
IDE LED	*IDE "Activity" Indicator* - Indicates when IDE activity is occurring, (Green LED).
Debug LED 1	*VMESYSFAIL* - LED #1 lights green when MPP24 is an output and driven high. The LED lights yellow when MPP24 is an output and driven Low, or if a VMEbus SYSFAIL occurs. The LED is not lit when MPP24 is an input, or if a SYSFAIL occurs while MPP24 is driven High.
Debug LED 2	*Debug2* - LED #2 lights green when MPP25 is an output and driven high. LED #2 Lights yellow when MPP25 is an output and driven Low. The LED is not lit when MPP25 is an input.
LAN1 Active	Indicates the Ethernet is active, (Yellow LED).
LAN1 Link	Indicates link speed, Yellow LED indicates 100BaseTx, Green LED indicates 1000BaseT and no LED indicates 10BaseT.
LAN0 Active	Indicates the Ethernet is active, (Yellow LED).
LAN0 Link	Indicates link speed, Yellow LED indicates 100BaseTx, Green LED indicates 1000BaseT and no LED indicates 10BaseT.

RS-232/422/485 Serial Port (COM1)

The serial port can be used to control, monitor and set up the VMIVME-7050. Using a second PC as a remote terminal with a Windows-based OS, go to Accessories, then to Hyperterminal and click on *Hyperterminal*. When using a Linux-based OS, go to the root directory, at the command prompt type in Minicom.



Figure 1-5 Example: Desktop Windows-Based OS

1. The following window will appear for Windows-based OS. From the Connection window, type in the name (7050). You can also choose an icon for the connection. Click OK.

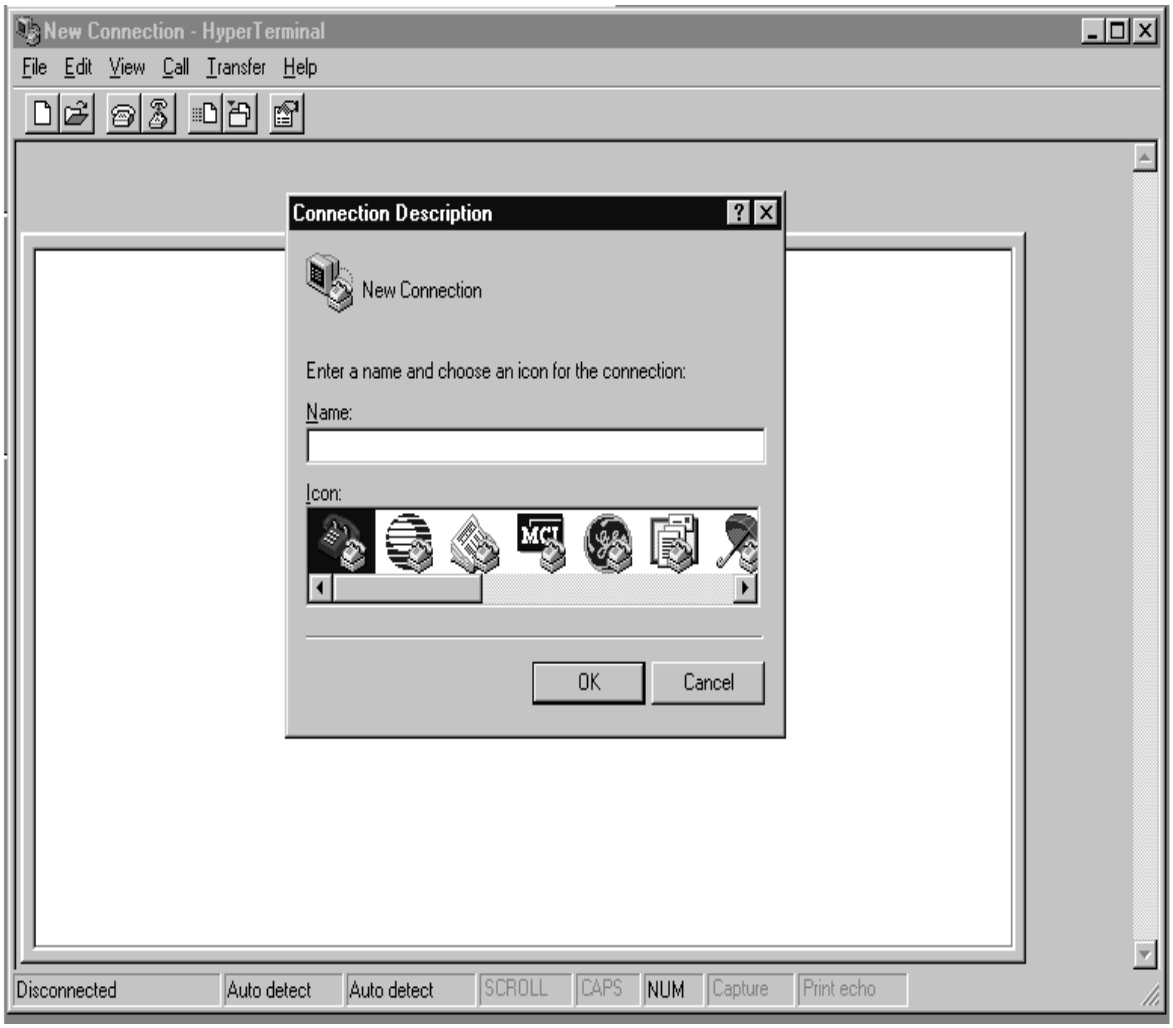


Figure 1-6 Hyperterminal New Connection Window

2. Select the COM port you will be using. After selecting the COM port click OK.

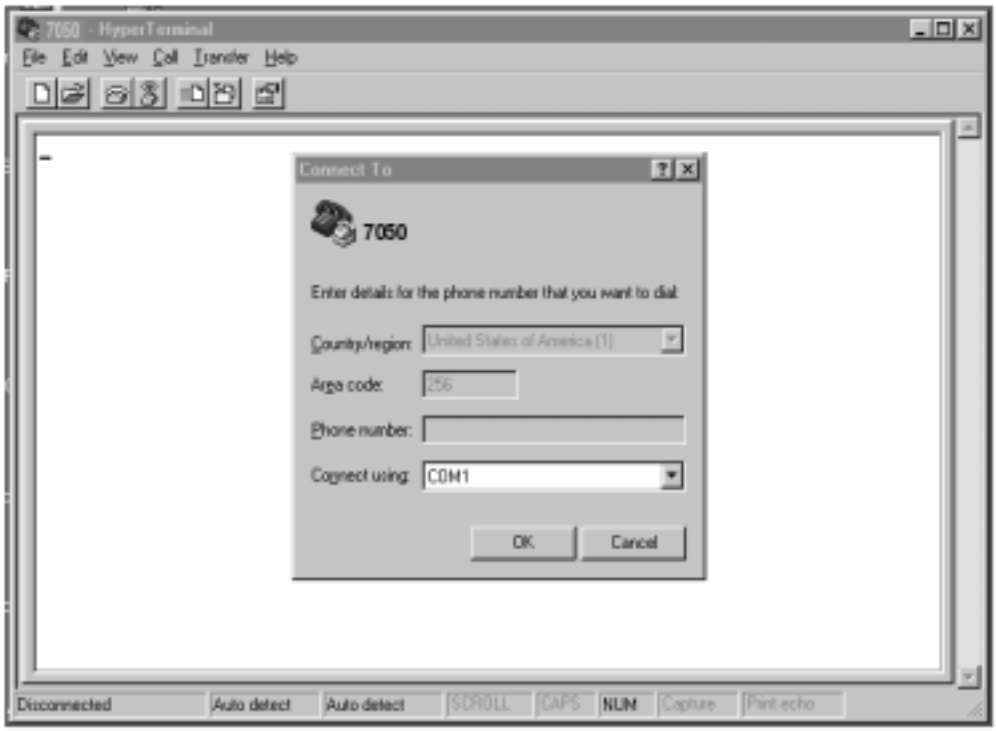


Figure 1-7 COM Port Window

3. The COM port properties window will appear. Select the Baud rate (bits per second). The default Baud rate for the VMIVME-7050 is 9600. Click OK.

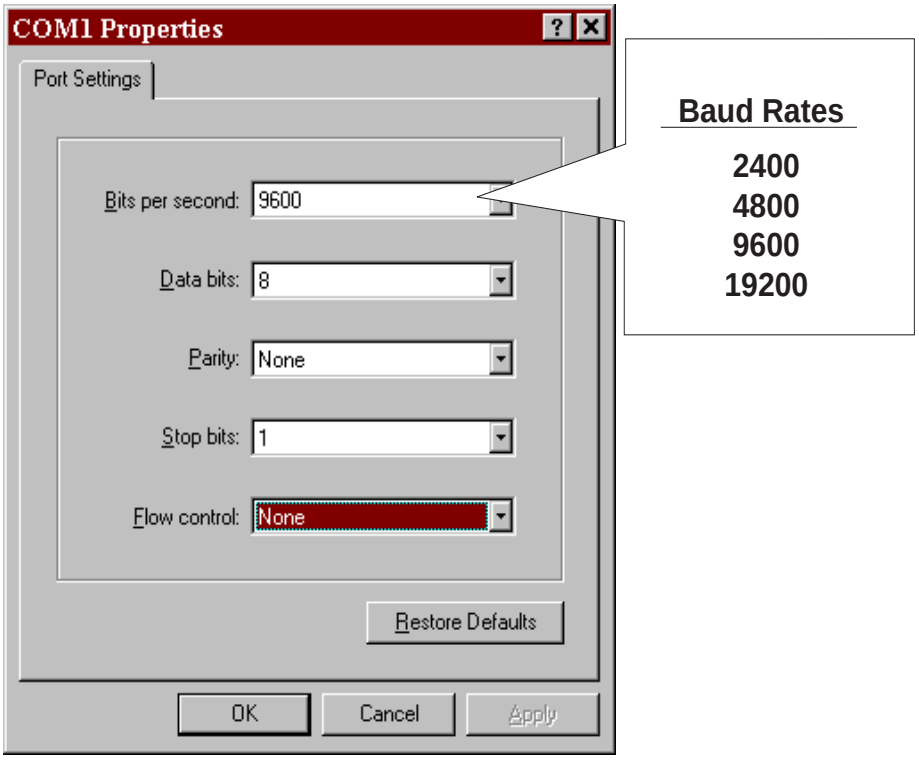


Figure 1-8 COM1 Property Window

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Using the Serial Port (Debug)

The RS-232 serial port is used to monitor and control the VMIVME-7050. The VMIVME-7050 can be cabled to a standard PC with an available serial port. See Figure 1-9 below for an example of connecting the VMIVME-7050 with a PC using the RS-232 serial port and cable.

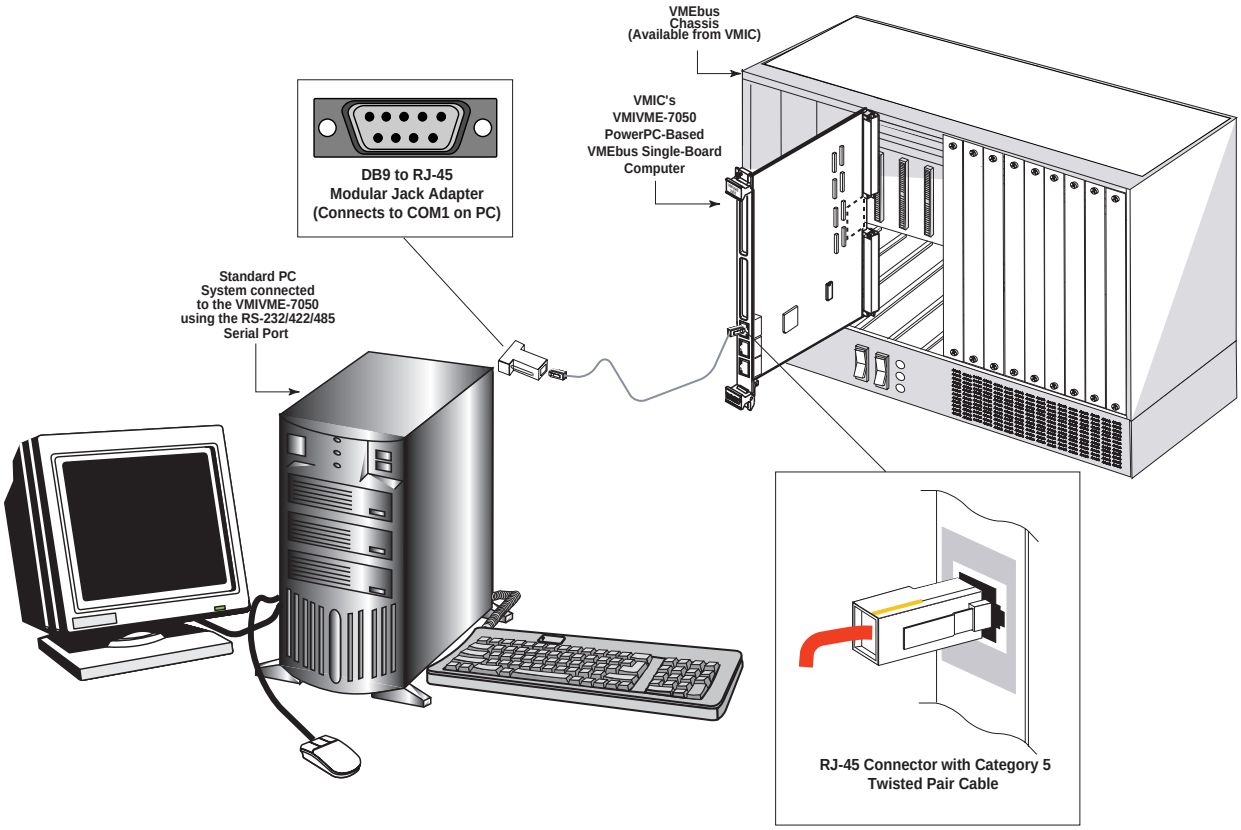


Figure 1-9 VMIVME-7050 Serial Port Connection

RJ-45 to DB9 COM Port Adapter	
PN#	**Type**
360-010030-001	DB9 Male
360-010030-002	DB9 Female
NOTE: One adapter is included with the VMIVME-7050. Both can be ordered separately.	

PMC Connector Pinouts

PMC Site#1 (J11) Connector and Pinout

The PCI Mezzanine Card (PMC) carries the same signals as the PCI standard; however, the PMC standard uses a completely different form factor. Tables 1-3 through 1-6 are the pinouts for the PMC #1 connectors (J11, J12, J13 and J14).

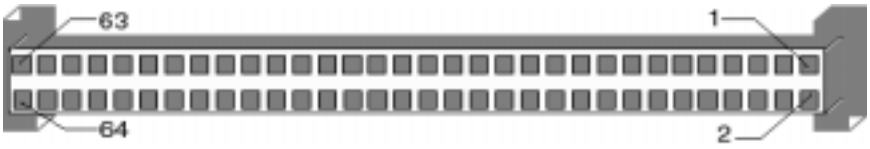


Table 1-3 PMC Site#1 (J11) Connector Pinout

PMC Connector (J11)				PMC Connector (J11)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	GND	2	-12 V	33	FRAME#	34	GND
3	GND	4	INTA#	35	GND	36	IRDY#
5	INTB#	6	INTA#	37	DEVSEL#	38	+5.0V
7	PRSNT#	8	+5.0V	39	CAP	40	LOCK#
9	INTB#	10	NC	41	SDONE#	42	SB0#
11	GND	12	NC	43	PAR	44	GND
13	CLK_Px	14	GND	45	VIO	46	AD[15]
15	GND	16	GNT0#	47	AD[12]	48	AD[11]
17	REQ0#	18	+5.0V	49	AD[9]	50	+5.0 V
19	VIO	20	AD[31]	51	GND	52	C/BE[0]#
21	AD[28]	22	AD[27]	53	AD[6]	54	AD[5]
23	AD[25]	24	GND	55	AD[4]	56	GND
25	GND	26	C/BE[3]#	57	VIO	58	AD[3]
27	AD[22]	28	AD[21]	59	AD[2]	60	AD[1]
29	AD[19]	30	+5.0V	61	AD[0]	62	+5.0 V
31	VIO	32	AD[17]	63	GND	64	REQ64#

PMC Site#1 (J12) Connector and Pinout

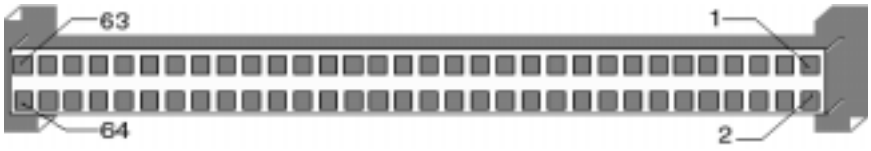


Table 1-4 PMC Site#1 (J12) Connector Pinout

PMC Connector (J12)				PMC Connector (J12)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	+12.0V	2	N/C	33	GND	34	N/C
3	GND	4	N/C	35	TRDY#	36	+3.3V
5	N/C	6	GND	37	GND	38	STOP#
7	GND	8	NC	39	PERR#	40	GND
9	N/C	10	NC	41	+3.3V	42	SERR#
11	BMODE2#	12	+3.3V	43	C/BE[1]	44	GND
13	RST#	14	BMODE3#	45	AD[14]	46	AD[13]
15	+3.3V	16	BMODE4#	47	M66ENA	48	AD[10]
17	N/C	18	GND	49	AD[8]	50	+3.3V
19	AD[30]	20	AD[29]	51	AD[7]	52	N/C
21	GND	22	AD[26]	53	+3.3V	54	N/C
23	AD[24]	24	+3.3V	55	N/C	56	GND
25	IDSEL	26	AD[23]	57	N/C	58	N/C
27	+3.3V	28	AD[20]	59	GND	60	N/C
29	AD[18]	30	GND	61	ACK64#	62	+3.3V
31	AD[16]	32	C/BE[2]#	63	GND	64	NC

PMC Site#1 (J13) Connector and Pinout

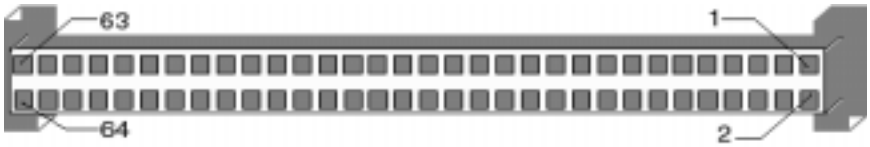


Table 1-5 PMC Site#1 (J13) Connector Pinout

PMC Connector (J13)				PMC Connector (J13)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	NC	2	GND	33	GND	34	AD[48]
3	GND	4	C/BE[7]#	35	AD[47]	36	AD[46]
5	C/BE[6]#	6	C/BE[5]#	37	AD[45]	38	GND
7	C/BE[4]#	8	GND	39	VIO	40	AD[44]
9	VIO	10	PAR64	41	AD[43]	42	AD[42]
11	AD[63]	12	AD[62]	43	AD[41]	44	GND
13	AD[61]	14	GND	45	GND	46	AD[40]
15	GND	16	AD[60]	47	AD[39]	48	AD[38]
17	AD[59]	18	AD[58]	49	AD[37]	50	GND
19	AD[57]	20	GND	51	GND	52	AD[36]
21	VIO	22	AD[56]	53	AD[35]	54	AD[34]
23	AD[55]	24	AD[54]	55	AD[33]	56	GND
25	AD[53]	26	GND	57	VIO	58	AD[32]
27	GND	28	AD[52]	59	NC	60	NC
29	AD[51]	30	AD[50]	61	NC	62	GND
31	AD[49]	32	GND	63	GND	64	NC

PMC Site#1 (J14) Connector and Pinout

All 64-pins of PMC I/O (J14) are routed to the VMEbus backplane connector P0.

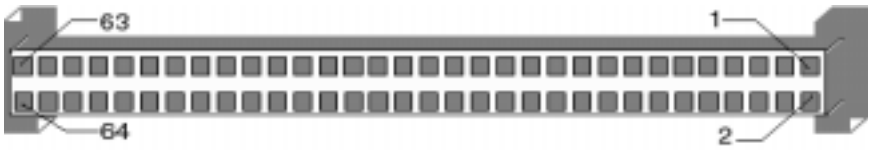


Table 1-6 PMC Site#1 (J14) Connector Pinout

PMC Connector (J14)				PMC Connector (J14)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	J14[1]	2	J14[2]	33	J14[33]	34	J14[34]
3	J14[3]	4	J14[4]	35	J14[35]	36	J14[36]
5	J14[5]	6	J14[6]	37	J14[37]	38	J14[38]
7	J14[7]	8	J14[8]	39	J14[39]	40	J14[40]
9	J14[9]	10	J14[10]	41	J14[41]	42	J14[42]
11	J14[11]	12	J14[12]	43	J14[43]	44	J14[44]
13	J14[13]	14	J14[14]	45	J14[45]	46	J14[46]
15	J14[15]	16	J14[16]	47	J14[47]	48	J14[48]
17	J14[17]	18	J14[18]	49	J14[49]	50	J14[50]
19	J14[19]	20	J14[20]	51	J14[51]	52	J14[52]
21	J14[21]	22	J14[22]	53	J14[53]	54	J14[54]
23	J14[23]	24	J14[24]	55	J14[55]	56	J14[56]
25	J14[25]	26	J14[26]	57	J14[57]	58	J14[58]
27	J14[27]	28	J14[28]	59	J14[59]	60	J14[60]
29	J14[29]	30	J14[30]	61	J14[61]	62	J14[62]
31	J14[31]	32	J14[32]	63	J14[63]	64	J14[64]

PMC Site#2 (J21) Connector and Pinout

The PCI Mezzanine Card (PMC) carries the same signals as the PCI standard; however, the PMC standard uses a completely different form factor. Tables 1-7 through 1-10 are the pinouts for the PMC #2 connectors (J21, J22, J23 and J24).

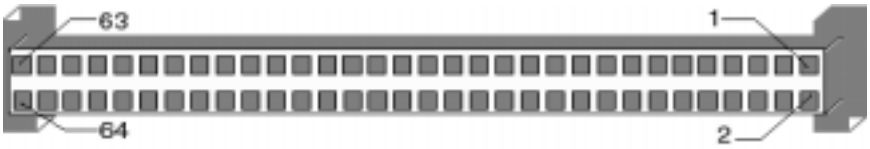


Table 1-7 PMC Site#1 (J21) Connector Pinout

PMC Connector (J21)				PMC Connector (J21)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	GND	2	-12 V	33	FRAME#	34	GND
3	GND	4	INTB#	35	GND	36	IRDY#
5	INTA#	6	INTB#	37	DEVSEL#	38	+5.0V
7	PRSNT#	8	+5.0V	39	CAP	40	LOCK#
9	INTA#	10	NC	41	SDONE#	42	SB0#
11	GND	12	NC	43	PAR	44	GND
13	CLK_Px	14	GND	45	VIO	46	AD[15]
15	GND	16	GNT4#	47	AD[12]	48	AD[11]
17	REQ4#	18	+5.0V	49	AD[9]	50	+5.0 V
19	VIO	20	AD[31]	51	GND	52	C/BE[0]#
21	AD[28]	22	AD[27]	53	AD[6]	54	AD[5]
23	AD[25]	24	GND	55	AD[4]	56	GND
25	GND	26	C/BE[3]#	57	VIO	58	AD[3]
27	AD[22]	28	AD[21]	59	AD[2]	60	AD[1]
29	AD[19]	30	+5.0V	61	AD[0]	62	+5.0 V
31	VIO	32	AD[17]	63	GND	64	REQ64#

PMC Site#2 (J22) Connector and Pinout

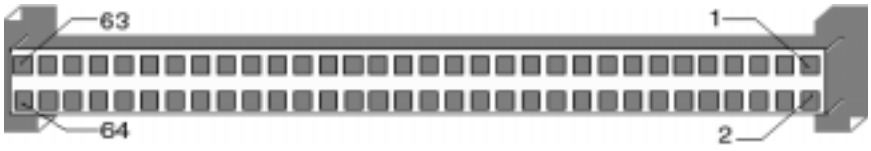


Table 1-8 PMC Site#1 (J22) Connector Pinout

PMC Connector (J22)				PMC Connector (J22)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	+12.0V	2	N/C	33	GND	34	N/C
3	GND	4	N/C	35	TRDY#	36	+3.3V
5	N/C	6	GND	37	GND	38	STOP#
7	GND	8	N/C	39	PERR#	40	GND
9	N/C	10	N/C	41	+3.3V	42	SERR#
11	BMODE2#	12	+3.3V	43	C/BE[1]	44	GND
13	RST#	14	BMODE3#	45	AD[14]	46	AD[13]
15	+3.3V	16	BMODE4#	47	M66ENA	48	AD[10]
17	NC	18	GND	49	AD[8]	50	+3.3V
19	AD[30]	20	AD[29]	51	AD[7]	52	N/C
21	GND	22	AD[26]	53	+3.3V	54	N/C
23	AD[24]	24	+3.3V	55	N/C	56	GND
25	IDSEL	26	AD[23]	57	N/C	58	N/C
27	+3.3V	28	AD[20]	59	GND	60	N/C
29	AD[18]	30	GND	61	ACK64#	62	+3.3V
31	AD[16]	32	C/BE[2]#	63	GND	64	N/C

PMC Site#2 (J23) Connector and Pinout

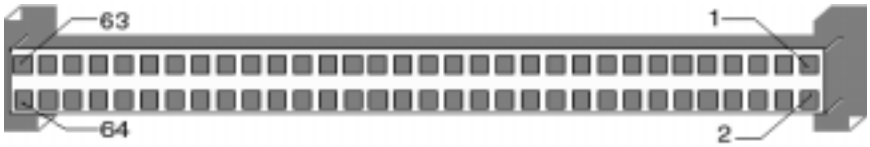


Table 1-9 PMC Site#2 (J23) Connector Pinout

PMC Connector (J23)				PMC Connector (J23)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	NC	2	GND	33	GND	34	AD[48]
3	GND	4	C/BE[7]#	35	AD[47]	36	AD[46]
5	C/BE[6]#	6	C/BE[5]#	37	AD[45]	38	GND
7	C/BE[4]#	8	GND	39	VIO	40	AD[44]
9	VIO	10	PAR64	41	AD[43]	42	AD[42]
11	AD[63]	12	AD[62]	43	AD[41]	44	GND
13	AD[61]	14	GND	45	GND	46	AD[40]
15	GND	16	AD[60]	47	AD[39]	48	AD[38]
17	AD[59]	18	AD[58]	49	AD[37]	50	GND
19	AD[57]	20	GND	51	GND	52	AD[36]
21	VIO	22	AD[56]	53	AD[35]	54	AD[34]
23	AD[55]	24	AD[54]	55	AD[33]	56	GND
25	AD[53]	26	GND	57	VIO	58	AD[32]
27	GND	28	AD[52]	59	NC	60	NC
29	AD[51]	30	AD[50]	61	NC	62	GND
31	AD[49]	32	GND	63	GND	64	NC

PMC Site#2 (J24) Connector and Pinout

All 64-pins of PMC I/O (J24) are routed to the VMEbus backplane connector P2.

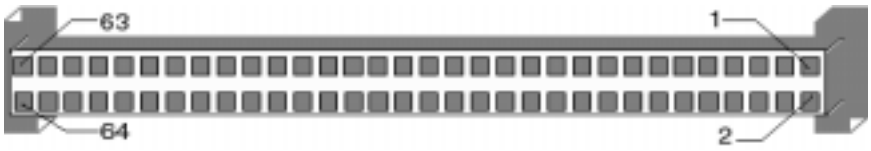
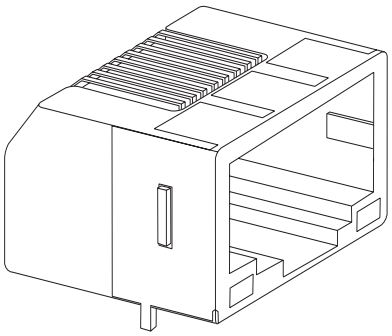


Table 1-10 PMC Site#1 (J24) Connector Pinout

PMC Connector (J24)				PMC Connector (J24)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	J24[1]	2	J24[2]	33	J24[33]	34	J24[34]
3	J24[3]	4	J24[4]	35	J24[35]	36	J24[36]
5	J24[5]	6	J24[6]	37	J24[37]	38	J24[38]
7	J24[7]	8	J24[8]	39	J24[39]	40	J24[40]
9	J24[9]	10	J24[10]	41	J24[41]	42	J24[42]
11	J24[11]	12	J24[12]	43	J24[43]	44	J24[44]
13	J24[13]	14	J24[14]	45	J24[45]	46	J24[46]
15	J24[15]	16	J24[16]	47	J24[47]	48	J24[48]
17	J24[17]	18	J24[18]	49	J24[49]	50	J24[50]
19	J24[19]	20	J24[20]	51	J24[51]	52	J24[52]
21	J24[21]	22	J24[22]	53	J24[53]	54	J24[54]
23	J24[23]	24	J24[24]	55	J24[55]	56	J24[56]
25	J24[25]	26	J24[26]	57	J24[57]	58	J24[58]
27	J24[27]	28	J24[28]	59	J24[59]	60	J24[60]
29	J24[29]	30	J24[30]	61	J24[61]	62	J24[62]
31	J24[31]	32	J24[32]	63	J24[63]	64	J24[64]

Serial Connector and Pinout

The RS-232 serial port connector is an RJ-45 connector as shown in Figure 1-10. An adapter to convert to a standard D9 connector is included (also sold separately). Please refer to the product specification sheet for ordering information.

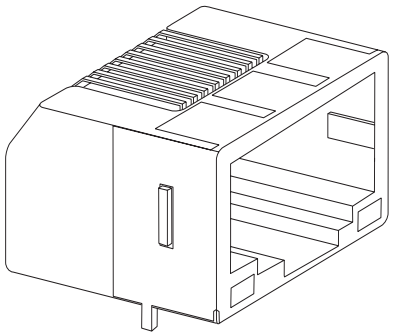


COM 1 Serial Port Connector			
RJ-45 Pin#	**DIR**	**RS-232 Signal**	**Function**
1			Data Carrier Detect (Not Used)
2	Out	RTS	Request to Send
3		CGND	Chassis Ground
4	Out	TxD	Transmit Data
5	In	RxD	Receive Data
6		GND	Ground
7	In	CTS	Clear to Send
8	Out	DTR	Data Terminal Ready

Figure 1-10 Serial Connector Pinouts

Ethernet Connector Pinout (J1 and J2)

The pinout diagram for the Ethernet 10BaseT, 100BaseTx and 1000BaseT connectors are shown in Figure 1-11.

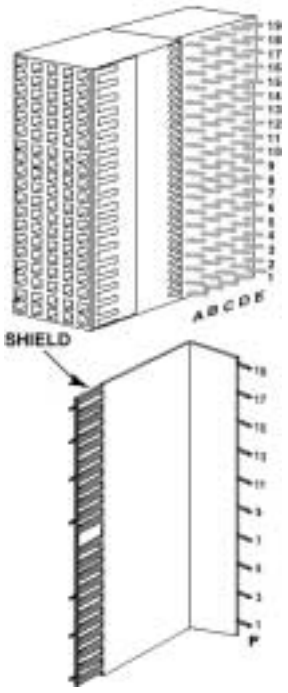


Ethernet Connectors (10BaseT, 100BaseTx and 1000BaseT)	
PIN	**Signal Name**
1	MX0+
2	MX0-
3	MX1+
4	MX2+
5	MX2-
6	MX1-
7	MX3+
8	MX3-

Figure 1-11 Ethernet Connector and Pinout

Optional P0 Connector Pinout

The P0 connector is a 5 row, 19 pins each, 2mm “Hard Metric” impedance matching connector. An additional external metal shield is also used, labeled row F. Figure 1-12 illustrates the P0 connector and the connector pinout. This connector is used to route the I/O signals of the PMC Site#1 and the secondary IDE signals to the backplane I/O.



Pin No.	Row A	Row B	Row C	Row D	Row E	Row F
19	IDES_DA2	J14[2]	J14[6]	J14[1]	J14[5]	GND
18	IDES_DA1	J14[4]	J14[8]	J14[3]	J14[7]	GND
17	IDES_DA0	J14[10]	J14[14]	J14[9]	J14[13]	GND
16	IDES_D[15]	J14[12]	J14[16]	J14[11]	J14[15]	GND
15	IDES_D[14]	J14[18]	J14[22]	J14[17]	J14[21]	GND
14	IDES_D[13]	J14[20]	J14[24]	J14[19]	J14[23]	GND
13	IDES_D[12]	J14[26]	J14[30]	J14[25]	J14[29]	GND
12	IDES_D[11]	J14[28]	J14[32]	J14[27]	J14[31]	GND
11	IDES_D[10]	J14[34]	J14[38]	J14[33]	J14[37]	GND
10	IDES_D[9]	J14[36]	J14[40]	J14[35]	J14[39]	GND
9	IDES_D[8]	J14[42]	J14[46]	J14[41]	J14[45]	GND
8	IDES_D[7]	J14[44]	J14[48]	J14[43]	J14[47]	GND
7	IDES_D[6]	J14[50]	J14[54]	J14[49]	J14[53]	GND
6	IDES_D[5]	J14[52]	J14[56]	J14[51]	J14[55]	GND
5	IDES_D[4]	J14[58]	J14[62]	J14[57]	J14[61]	GND
4	IDES_D[3]	J14[60]	J14[64]	J14[59]	J14[63]	GND
3	IDES_D[2]	NC	NC	IDES_DIOW#	IDES_DMACK	GND
2	IDES_D[1]	IDES_CS1#	IDES_INTRQ	IDES_DMARQ	IDES_IORDY	GND
1	IDES_D[0]	IDES_CS0#	IDES_CBLIO#	IDES_RESET#	IDES_DIOR#	GND

Figure 1-12 P0 Connector and Pinout

NOTE: Backplane designs should route P0 signals straight through (not bused). The rear transition board can then properly utilized the high-speed PMC#1 I/O signals.

VMEbus Connector Pinout

Figure 1-13 shows the location of the VMEbus P1 and P2 connectors and their orientation on the VMIVME-7050. Table 1-11 shows the pin assignments for the VMEbus connectors. Note that only Row B of connector P2 is used; all other pins on P2 are reserved and should not be connected.

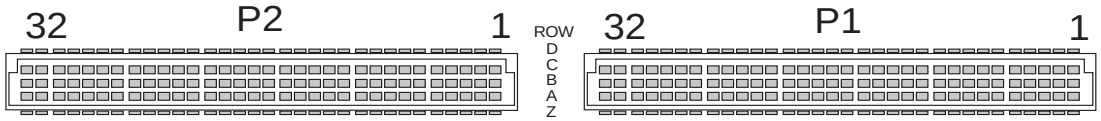


Figure 1-13 VMEbus Connector Diagram

Table 1-11 VMEbus Connector Pinout

Pin #	P1 Row Z Signal	P1 Row A Signal	P1 Row B Signal	P1 Row C Signal	P1 Row D Signal	P2 Row Z Signal	P2 Row A Signal	P2 Row B Signal	P2 Row C Signal	P2 Row D Signal
1	N/C	D00	BBSY	D08	N/C	CONN [2]	J24[47]	+5 V	SP0_RxD+	J24[1]
2	N/C	D01	BCLR	D09	N/C	GND	J24[48]	GND	SP0_RxD-	J24[3]
3	N/C	D02	ACFAIL	D10	N/C	CONN [5]	J24[49]	Reserved	SP0_CTS+	J24[4]
4	N/C	D03	BG0IN	D11	N/C	GND	J24[50]	A24	SP0_CTS-	J24[6]
5	N/C	D04	BG0OUT	D12	N/C	CONN [8]	J24[51]	A25	SP1_RTS-	J24[7]
6	N/C	D05	BG1IN	D13	N/C	GND	J24[52]	A26	SP1_RTS+	J24[9]
7	N/C	D06	BG1OUT	D14	N/C	CONN [11]	G0_0+	A27	SP1_TxD+	J24[10]
8	N/C	D07	BG2IN	D15	N/C	GND	G0_0-	A28	SP1_TxD-	J24[12]
9	N/C	GND	BG2OUT	GND	GA5	CONN [14]	G0_1+	A29	SP0_TxD-	J24[13]
10	N/C	SYSCLK	BG3IN	SYSFAIL	GA0	GND	G0_1-	A30	SP0_TxD+	J24[15]
11	N/C	GND	BG3OUT	BERR	GA1	CONN [17]	G0_2+	A31	SP0_RTS+	J24[16]
12	N/C	DS1#	BR0	SYSRESET	N/C	GND	G0_2-	GND	SP0_RTS-	J24[18]
13	N/C	DS0#	BR1	LWORD	GA2	CONN [20]	G0_3+	+5 V	SP1_CTS-	J24[19]
14	N/C	WRITE	BR2	AM5	N/C	GND	G0_3-	D16	SP1_CTS+	J24[21]
15	N/C	GND	BR3	A23	GA3	CONN [23]	G1_0+	D17	SP1_RxD-	J24[22]
16	N/C	DTACK	AM0	A22	N/C	GND	G1_0-	D18	SP1_RxD+	J24[24]
17	N/C	GND	AM1	A21	GA4	CONN [26]	G1_1+	D19	N/C	J24[25]
18	N/C	AS#	AM2	A20	N/C	GND	G1_1-	D20	J24[63]	J24[27]
19	N/C	GND	AM3	A19	N/C	CONN [29]	G1_2+	D21	J24[64]	J24[28]
20	N/C	IACK	GND	A18	N/C	GND	G1_2-	D22	N/C	J24[30]

Table 1-11 VMEbus Connector Pinout (Continued)

Pin #	P1 Row Z Signal	P1 Row A Signal	P1 Row B Signal	P1 Row C Signal	P1 Row D Signal	P2 Row Z Signal	P2 Row A Signal	P2 Row B Signal	P2 Row C Signal	P2 Row D Signal
21	I2C_RST#	IACKIN	I2C_CL	A17	N/C	CONN [32]	G1_3+	D23	N/C	J24[31]
22	N/C	IACKOUT	I2C_DA	A16	N/C	GND	G1_3-	GND	N/C	J24[33]
23	N/C	AM4	GND	A15	I2C_SMI#	CONN [35]	J24[53]	D24	N/C	J24[34]
24	N/C	A07	IRQ7	A14	N/C	GND	J24[54]	D25	N/C	J24[36]
25	N/C	A06	IRQ6	A13	N/C	CONN [38]	J24[55]	D26	N/C	J24[37]
26	N/C	A05	IRQ5	A12	N/C	GND	J24[56]	D27	N/C	J24[39]
27	N/C	A04	IRQ4	A11	N/C	CONN [41]	J24[57]	D28	MPP24	J24[40]
28	N/C	A03	IRQ3	A10	N/C	GND	J24[58]	D29	MPP25	J24[42]
29	N/C	A02	IRQ2	A09	N/C	CONN [44]	J24[59]	D30	P2_C29	J24[43]
30	N/C	A01	IRQ1	A08	N/C	GND	J24[60]	D31	P2_C30	J24[45]
31	N/C	-12 V	+5 V STDBY	+12 V	N/C	CONN [46]	J24[61]	GND	N/C	GND
32	N/C	+5 V	+5 V	+5 V	N/C	GND	J24[62]	+5 V	N/C	Reserved

CHAPTER 2

PowerPC Programming

Contents

Physical Memory 56
Instruction Set 58
On-Chip Cache Implementation 61
Exception Model 64
Memory Management 68
Instruction Timing 70
Power Management 72
Thermal Management 74
Performance Monitor 75
BootROM Monitor (PMON) 76
Example of using PMON for Assembly-level debugging 81

Introduction

The VMIVME-7050 is a PowerPC processor-based single board computer.

Physical Memory

The VMIVME-7050 provides dual date rate (DDR) Synchronous DRAM (SDRAM) as on-board system memory. Memory can be accessed as bytes, words or longwords.

The VMIVME-7050 accepts one SDRAM SODIMM for a maximum capacity of 2GB of PC2100 DDR memory with ECC support. The on-board SDRAM is dual-ported to the VMEbus through the PCI-to-VMEbus bridge and is addressable by the local processor, as well as the VMEbus slave interface by another VMEbus master. Caution must be used when sharing memory between the local processor and the VMEbus to prevent a VMEbus master from overwriting the local processor's operating system.

The VMIVME-7050 includes 32 KB of non-volatile SRAM which can be accessed by the CPU at any time, and is used to store system data that must not be lost during power-off conditions.

NOTE: Memory capacity may be extended as higher density parts become available.

PowerPC Registers and Programming Model

The PowerPC Architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction itself. The three-register instruction formats allow specification of a target register distinct from the two source operands. Only load and store instructions transfer data between registers and memory.

PowerPC processors have two levels of privilege—supervisor mode of operation (typically used by the operating system) and user mode of operation (used by the application software, it is also called problem state).

The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers. Each PowerPC microprocessor also has its own unique set of hardware implementation dependent (HID) registers.

While running in supervisor mode, the operating system is able to execute all instructions and access all registers defined in the PowerPC Architecture. In this mode the operating system establishes all address translations and protection mechanisms, loads all processor state registers, and sets up all other control mechanisms defined on the PowerPC 750FX/GX processors. While running in user mode (problem state) many of these registers and facilities are not accessible and any attempt to read or write these register results in a program exception.

Figure 2-1 on page 58 shows all the PowerPC 750FX/GX registers available at the user and supervisor level. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands to access the register.

Instruction Set

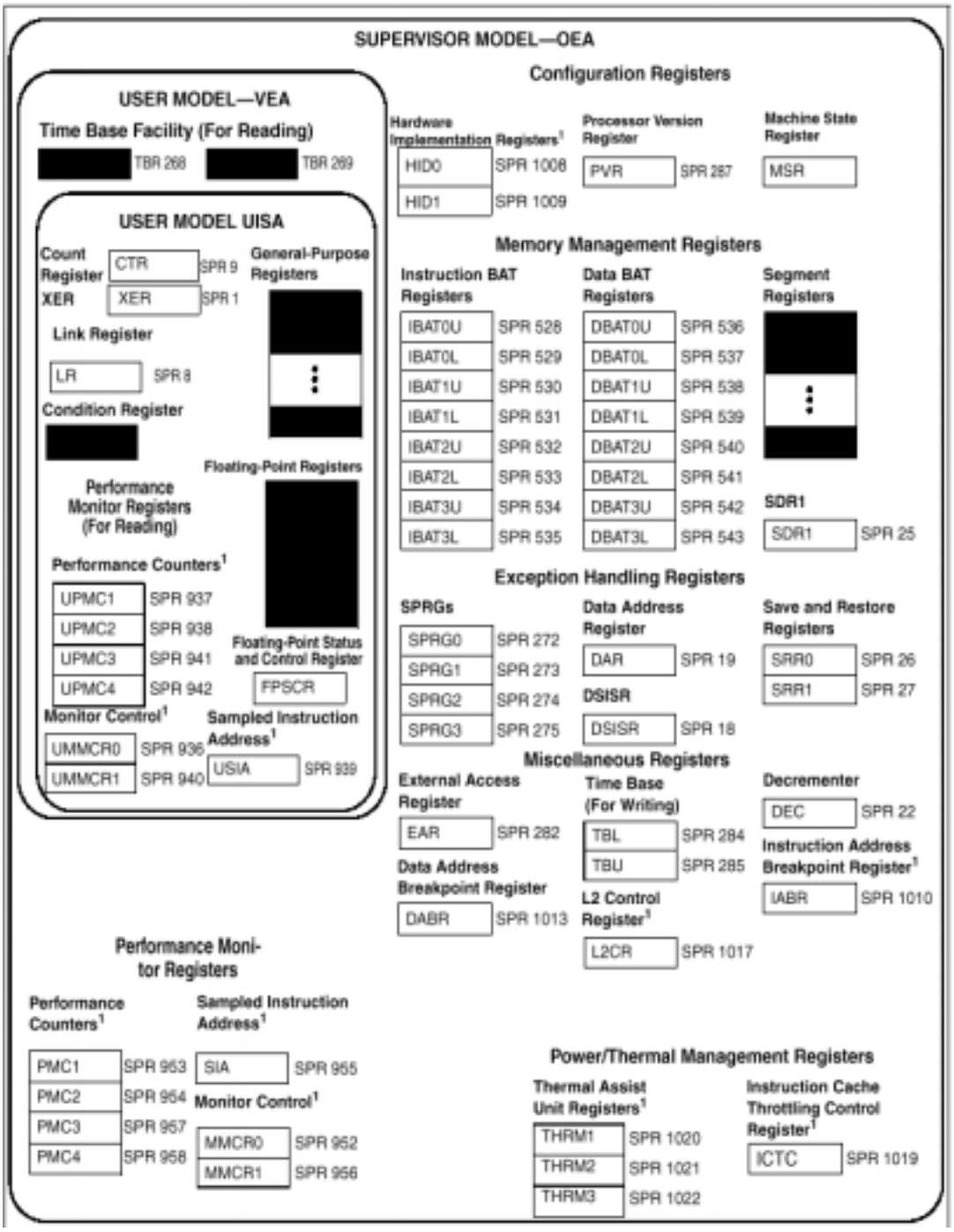


Figure 2-1 PowerPC Instruction Set

All PowerPC instructions are encoded as single-word (32-bit) instructions. Instruction formats are consistent among all instruction types (primary op-code is always 6 bits, register operands always specified in the same bit fields in the instruction), permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplify instruction pipelining.

PowerPC Instruction Set

The PowerPC instructions are divided into the following categories.

1. Integer instructions: These include computational and logical instructions.
   - Integer arithmetic instructions
   - Integer compare instructions
   - Integer logical instructions
   - Integer rotate and shift instructions
2. Floating-point instructions: These include floating-point computational instructions, as well as instructions that affect the FPSCR.
   - Floating-point arithmetic instructions
   - Floating-point multiply/add instructions
   - Floating-point rounding and conversion instructions
   - Floating-point compare instructions
   - Floating-point status and control instructions
3. Load/store instructions: Include integer and floating-point load and store instructions.
   - Integer load and store instructions
   - Integer load and store multiple instructions
   - Floating-point load and store
   - Primitives used to construct atomic memory operations (lwarx and stwcx. instructions)
4. Flow control instructions: Include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
   - Branch and trap instructions
   - Condition register logical instructions (sets conditions for branches)
   - System Call
5. Processor control instructions: These instructions are used for synchronizing memory accesses and management of caches, TLBs and the segment registers.
   - Move to/from SPR instructions
   - Move to/from MSR
   - Synchronize (processor and memory system)
   - Instruction synchronize
   - Order loads and stores

PowerPC Instruction Set (Continued)

6. Memory control instructions: To provide control of caches, TLBs and SRs.
   - Supervisor-level cache management instructions
   - User-level cache instructions
   - Segment register manipulation instructions
   - Translation lookaside buffer management instructions

This grouping does not indicate the execution unit that executes a particular instruction or group of instructions.

Integer instructions operate on byte, word and Longword operands. Floating-point instructions operate on single-precision (one word) and double-precision (two words) floating-point operands. The PowerPC Architecture uses instructions that are four bytes long and word-aligned. It provides for integer byte, word and Longword operand loads and stores between memory and a set of 32 GPRs. It also provides for single and double-precision loads and stores between memory and a set of 32 floating-point registers (FPRs).

Computational instructions do not access memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location using three or more instructions.

PowerPC processors follow the program flow when they are in the normal execution state; however, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event.

Either type of exception will cause the associated exception handler to be invoked.

Effective address computations for both data and instruction accesses use 32-bit signed two's complement binary arithmetic. A carry from Bit 0 and overflow are ignored.

On-Chip Cache Implementation

This section describes the PowerPC Architecture treatment of cache in general, and the PowerPC 750FX/GX-specific implementation. The PowerPC 750GX is very similar to the 750FX, with the exception that the L2 cache is twice as large (1MB compared to 512KB).

PowerPC Cache Model

The PowerPC Architecture does not define hardware aspects of cache implementations. For example, PowerPC processors can have unified caches, separate instruction and data caches (Harvard architecture), or no cache at all. PowerPC microprocessors control the following memory access modes on a virtual page or block (BAT) basis.

- Write-back / write-through mode
- Caching-inhibited mode
- Memory coherency

The caches are physically addressed, and the data cache can operate in either write-back or write-through mode, as specified by the PowerPC Architecture.

The PowerPC Architecture defines the term ‘cache block’ as the cacheable unit. The VEA and OEA define cache management instructions that a programmer can use to affect cache contents.

PowerPC 750FX/GX Microprocessor Cache Implementation

Description of 750FX/GX L1 Caches

- 32 Kbyte, 8-way set associative, L1 instruction cache (pseudo LRU replacement)
- 32 Kbyte, 8-way set associative, L1 data cache (pseudo LRU replacement)
- 32 byte line size with each line having one valid bit and three possible states:
    - — Exclusive
    - — Modified
    - — Invalid
- Physically addressed cache and cache directory. Physical address tag stored in the cache directory.

Description of 750FX/GX L2 Cache

The 750FX / GX 512Kbyte L2 cache is implemented with an internal two-way set-associative tag memory with 4096 tags per way, and an internal 512 Kbyte SRAM for data storage. The tags are sectored to support two cache blocks per tag entry (two 32-byte sectors totalling 64 bytes). Each sector (32-byte L1 cache block) in the L2 cache has its own valid and modified bits. In addition, the SRAM includes an 8-bit ECC for every double-word.

The ECC logic corrects most single-bit errors and detects double-bit errors as data is read from the SRAM. The L2 cache maintains cache coherency through snooping and is normally configured to operate in copy-back mode.

The L2 cache control register (L2CR) allows control of the following:

- L2 cache configuration
- Double bit error machine check
- Global invalidation of L2 contents
- Write-through operation
- L2 test support
- L2 locking by way 1

L2 Cache Operation

The L2 cache for the 750FX / GX is a combined instruction and data cache that receives memory requests from both L1 instruction and L1 data caches independently. The L1 requests are generally the result of instruction fetch misses, data load or store misses, write-through operations, or cache management instructions. Each L1 request generates an address lookup in the L2 tags. If a hit occurs, the instructions or data are forwarded to the L1 cache. A miss in the L2 tags causes the L1 request to be forwarded to the 60x bus interface. The cache block received from the bus is forwarded to the L1 cache immediately, and is also loaded into the L2 cache with the tag marked valid and unmodified. If the cache block loaded into the L2 causes a new tag entry to be allocated and the current tag entry is marked valid modified, the modified sectors of the tag to be replaced are castout from the L2 cache to the 60x bus.

At any given time the L1 instruction cache may have one instruction fetch request, and the L1 data cache may have two loads and two stores requesting L2 cache access. The L2 cache also services snoop requests from the 60x bus. When there are multiple pending requests to the L2 cache, snoop requests have highest priority, followed by data load and store requests (serviced on a first-in, first-out basis). Instruction fetch requests have the lowest priority in accessing the L2 cache when there are multiple accesses pending.

If read requests from both the L1 instruction and L1 data caches are pending, the L2 cache can perform hit-under-miss and supplies the available instruction or data while a bus transaction for the previous L2 cache miss is being performed. The L2 cache supports miss under miss. Up to two outstanding misses are supported: one miss from the instruction cache and one from the data cache, or two data cache misses.

Requests that hit will be serviced even while misses are in progress on the bus.

All requests to the L2 cache that are marked cacheable (even if the respective L1 cache is disabled or locked) cause tag lookup and will be serviced if the instructions or data are in the L2 cache. Burst and single-beat read requests from the L1 caches that hit in the L2 cache are forwarded to the L1 caches as instructions or data, and the L2 LRU bit for that tag is updated. Burst writes from the L1 data cache due to a castout or replacement copyback are written only to the L2 cache, and the L2 cache sector is marked modified.

If the L2 cache is configured as write-through, the L2 sector is marked unmodified, and the write is forwarded to the 60x bus. If the L1 castout requires a new L2 tag entry to be allocated and the current tag is marked modified, any modified sectors of the tag to be replaced are cast out of the L2 cache to the 60x bus.

Single-beat read requests from the L1 caches that miss in the L2 cache do not cause any state changes in the L2 cache and are forwarded on the 60x bus interface. Cacheable single-beat store requests marked copy-back that hit in the L2 are allowed to update the L2 cache sector, but do not cause L2 cache sector allocation or deallocation. Cacheable, single-beat store requests that miss in the L2 are forwarded to the 60x bus. Single-beat store requests marked write-through (through address translation or through the configuration of L2CR[L2WT]) are written to the L2 cache if they hit and are written to the 60x bus independent of the L2 hit / miss status. If the store hits in the L2 cache, the modified / unmodified status of the tag remains unchanged. All requests to the L2 cache that are marked cache-inhibited by address translation (through either the MMU or by default WIMG configuration) bypass the L2 cache and do not cause any L2 cache tag state change.

Exception Model

The section describes the PowerPC exception model and the PowerPC 750FX/GX implementation.

PowerPC Exception Model

The PowerPC exception mechanism allows the processor to interrupt the instruction flow to handle certain situations caused by external signals, errors or unusual conditions arising from the instruction execution.

When exceptions occur, information about the state of the processor is saved to certain registers, and the processor begins execution at an address (exception vector) predetermined for each exception. System software must complete the saving of the processor state prior to servicing the exception. Exception processing proceeds in supervisor mode.

Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception — for example, the MSR, DSISR and the FPSCR contain status bits which farther identify the exception condition. Additionally, some exception conditions can be explicitly enabled or disabled by software.

The PowerPC Architecture requires that exceptions be handled in specific priority and program order; therefore, although a particular implementation may recognize exception conditions out of order, they are handled in program order. When an instruction-caused exception is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that are undispatched, are required to complete before the exception is taken, and any exceptions those instructions cause must also be handled first; likewise, asynchronous, precise exceptions are recognized when they occur but are not handled until the instructions currently in the completion queue successfully retire or generate an exception, and the completion queue is emptied.

Unless a catastrophic condition causes a system reset or machine check exception, only one exception is handled at a time. For example, if one instruction encounters multiple exception conditions, those conditions are handled sequentially in priority order. After the exception handler completes, the instruction processing continues until the next exception condition is encountered. Recognizing and handling exception conditions sequentially guarantees system integrity.

When an exception is taken, information about the processor state before the exception was taken is saved in SRR0 and SRR1. Exception handlers must save the information stored in SRR0 and SRR1 early to prevent the program state from being lost due to a system reset and machine check exception or due to an instruction caused exception in the exception handler, and before re-enabling external interrupts. The exception handler must also save and restore any GPR registers used by the handler.

The PowerPC Architecture supports four types of exceptions.

1. Synchronous, precise — These are caused by instructions. All instruction-caused exceptions are handled precisely; that is, the machine state at the time the exception occurs is known and can be completely restored. This means that (excluding the trap and system call exceptions) the address of the faulting instruction is provided to the exception handler and that neither the faulting instruction nor subsequent instructions in the code stream will complete execution before the exception is taken. Once the exception is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the exception handler). When an exception is taken due to a trap or system call instruction, execution resumes at an address provided by the handler.
2. Synchronous, imprecise — The PowerPC Architecture defines two imprecise floating-point exception modes, recoverable and nonrecoverable. Even though PowerPC 750FX/GX provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, enabled floating-point exceptions are always precise).
3. Asynchronous, maskable — The PowerPC Architecture defines external and decrementer interrupts as maskable, asynchronous exceptions. When these exceptions occur, their handling is postponed until the next instruction, and any exceptions associated with that instruction, completes execution. If no instructions are in the execution units, the exception is taken immediately upon determination of the correct restart address (for loading SRR0). As shown in the the PowerPC 750FX/GX implements additional asynchronous, maskable exceptions.
4. Asynchronous, nonmaskable — There are two nonmaskable asynchronous exceptions: system reset and the machine check exception. These exceptions may not be recoverable, or may provide a limited degree of recoverability. Exceptions report recoverability through the MSR[RI] bit.

Microprocessor Exception Implementation

The PowerPC 750FX/GX exception classes described above are shown in the Table 2-1 on page 66. Although exceptions have other characteristics, such as priority and recoverability, Table 2-1 describes categories of exceptions PowerPC 750FX/GX handles uniquely. Table 2-1 does not include synchronous imprecise exceptions; although the PowerPC Architecture supports imprecise handling of floating-point exceptions, PowerPC 750FX/GX implements these exception modes precisely.

Table 2-1 Microprocessor Exception Classifications

Synchronous/Asynchronous	Precise/Imprecise	Exception Type
Asynchronous, nonmaskable	Imprecise	Machine check, system reset
Asynchronous, maskable	Precise	External, decrementer, system management, performance monitor, and thermal management interrupts
Synchronous	Precise	Instruction-caused exceptions

Table 2-2 Exceptions and Conditions lists the PowerPC 750FX/GX exceptions and conditions that cause them. Exceptions specific to PowerPC 750FX/GX are indicated.

Table 2-2 Exceptions and Conditions

Exception Type	Vector Offset (hex)	Causing Conditions
Reserved	00000	—
System reset	00100	Assertion of either $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ or at power-on reset
Machine check	00200	Assertion of $\overline{\text{TEA}}$ during a data bus transaction, assertion of $\overline{\text{MCP}}$, an address, data or L2 double bit error. MSR[ME] must be set.
DSI	00300	As specified in the PowerPC Architecture. (e.g., page fault occurs)
ISI	00400	As defined by the PowerPC Architecture. (e.g., page fault occurs)
External interrupt	00500	MSR[EE] = 1 and INT is asserted.
Alignment	00600	A floating-point load/store, **stmw, stwcx, lmw**, **lwarx**, **eciwx** or **ecowx** instruction operand is not word-aligned. A multiple/string load/store operation is attempted in little-endian mode. The operand of **dcbz** is in memory that is write-through-required or caching-inhibited or the cache is disabled
Program	00700	As defined by the PowerPC Architecture.
Floating-point unavailable	00800	As defined by the PowerPC Architecture.
Decrementer	00900	As defined by the PowerPC Architecture, when the most significant bit of the DEC register changes from 0 to 1 and MSR[EE] = 1.
Reserved	00A00–00BFF	—
System call	00C00	Execution of the System Call (**sc**) instruction.
Trace	00D00	MSR[SE] = 1 or a branch instruction completes and MSR[BE] = 1. Unlike the architecture definition, **isync** does not cause a trace exception
Reserved	00E00	PowerPC 750FX/GX does not generate an exception to this vector. Other PowerPC processors may use this vector for floating-point assist exceptions.
Reserved	00E10–00EFF	—

Table 2-2 Exceptions and Conditions (Continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Performance monitor [1]	00F00	The limit specified in a PMC register is reached and MMCR0[ENINT] = 1
Instruction address breakpoint [1]	01300	IABR[0–29] matches EA[0–29] of the next instruction to complete, IABR[TE] matches MSR[IR], and IABR[BE] = 1.
Reserved	01400–016FF	—
Thermal management interrupt 1	01700	Thermal management is enabled, the junction temperature exceeds the threshold specified in THRM1 or THRM2, and MSR[EE] = 1.
Reserved	01800–02FFF	—
1. PowerPC 750FX / GX-specific		

Memory Management

This section describes the memory management features of the PowerPC Architecture, and the PowerPC 750FX/GX implementation.

PowerPC Memory Management Model

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses and to provide access protection on blocks and pages of memory. There are two types of accesses generated by the PowerPC 750FX/GX that require address translation—instruction fetches, and data accesses to memory generated by load, store and cache control instructions.

The PowerPC Architecture defines different resources for 32 and 64-bit processors; PowerPC 750FX/GX implements the 32-bit memory management model. The memory-management unit provides two types of memory access models: Block Address Translate (BAT) model and a virtual address model. The BAT block sizes range from 128Kbyte to 256Mbyte and are selectable from high order effective address bits and have priority over the virtual model. The virtual model employs a 52-bit virtual address space made up by a 24-bit segment address space and a 28-bit effective address space. The virtual model utilizes a demand paging method with a 4Kbyte page size. In both models address translation is done completely by hardware, in parallel with cache accesses, with no additional cycles incurred.

The PowerPC 750FX/GX MMU provides independent four-entry BAT arrays for instructions and data that maintain address translations for blocks of memory. These entries define blocks that can vary from 128Kbytes to 256Mbytes. The BAT arrays are maintained by system software. Instructions and data share the same virtual address model but could operate in separate segment spaces.

The PowerPC 750CX MMU and exception model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged implies that individual pages for data and instructions are loaded into physical memory from system disk only when they are required by an executing program. Infrequently used pages in memory are returned to disk or discarded if they have not been modified.

The hashed page table is a fixed-sized data structure (size should be determined by the amount of physical memory available to the system) that contains 8-byte entries (PTEs) that define the mapping between virtual pages and physical pages. The page table size is a power of 2 and is boundary aligned in memory based on the size of the table. The page table contains a number of page table entry groups (PTEGs). A PTEG contains eight page table entries (PTEs) of eight bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations. A given page translation can be found in one of two possible PTEG's. The size and location in memory of the page table is defined in the SDR1 register.

Setting MSR[IR] enables instruction address translations and MSR[DR] enables data address translations. If the bit is cleared, the respective effective address is used as the physical address.

PowerPC 750FX/GX Microprocessor Memory Management Implementation

The 750FX/GX implements a virtual memory management scheme compliant with the PowerPC specification for 32-bit microprocessors. The organization of the Memory Management Unit (MMU) hardware is as follows;

- 64-entry, 2-way set associative data TLB (total of 128)
- 64-entry, 2-way set associative instruction TLB (total of 128)
- Eight (8) entry, fully associative Instruction BAT
- Eight (8) entry, fully associative Data BAT
- Sixteen segment registers
- Hardware tablewalks

Instruction Timing

The PowerPC 750FX/GX is a pipelined, superscalar processor. A pipelined processor is one in which instruction processing is divided into discrete stages, allowing work to be done on multiple instructions in each stage. For example, after an instruction completes one stage, it can pass on to the next stage leaving the previous stage available to a subsequent instruction. This improves overall instruction throughput.

A superscalar processor is one that issues multiple independent instructions to separate execution units in a single cycle, allowing multiple instructions to execute in parallel. The PowerPC 750FX/GX has six independent execution units, two for integer instructions, and one each for floating-point instructions, branch instructions, load and store instructions, and system register instructions. Having separate GPRs and FPRs allows integer, floating-point calculations, and load and store operations to occur simultaneously without interference.

Additionally, rename buffers are provided to allow operations to post completed results for use by subsequent instructions without committing them to the architected FPR and GPR register files.

The common pipeline of the PowerPC 750FX/GX, has four stages through which all instructions must pass: fetch, decode/dispatch, execute, and complete/write back. Instructions flow sequentially through each stage. However, at dispatch a position is made available in the completion queue at the same time it enters the execution stage. This simplifies the completion operation when instructions are retired in program order. Both the load/store and floating-point units have multiple stages to execute their instructions. An instruction occupies only one stage at a time in all execution units. At each stage an instruction may proceed without delay or may stall. Stalls are caused by the requirement of additional processing or other events. For example divide instructions require multiple cycles to complete the operation, load and store instructions may stall waiting for address translation (TLB reload, page fault, and so forth).

This is a listing of the four major instruction pipeline stages in the PowerPC 750FX/GX.

1. Fetch — The fetch pipeline stage primarily involves fetching instructions from the memory system and keeping the instruction queue full. The BPU decodes branches after they are fetched and removes (folds out) those that do not update CTR or LR from the instruction stream. If the branch is taken or predicted as taken, the fetch unit is informed of the new address and fetching resumes along the taken patch. For branches not taken or predicted as not taken, sequential fetching continues.
2. Dispatch — The dispatch unit is responsible for taking instructions from the bottom two locations of the instruction queue and delivering them to an execution unit for further processing. Dispatch is responsible for decoding the instructions and determining which instructions can be dispatched. To qualify for dispatch, a reservation station, a rename buffer and a position in the completion queue all must be available. A branch instruction could be processed by the BPU on the same clock cycle for a maximum of three-instruction dispatch per cycle. The dispatch stage accesses operands, assigns a rename buffer for an operand(s) that updates an architected register(s) (GPR, FPR, CR, etc.) and

delivers the instruction to the reservation registers of the respective execution units. If a source operand is not available (a previous instruction is updating the item via a rename buffer) dispatch provides a tag that indicates which rename buffer will supply the operand when it becomes available. At the end of the dispatch stage, the instructions are removed from the instructions queue, latched into reservation stations at the appropriate execution unit and assigned positions in the completion buffers in sequential program order.

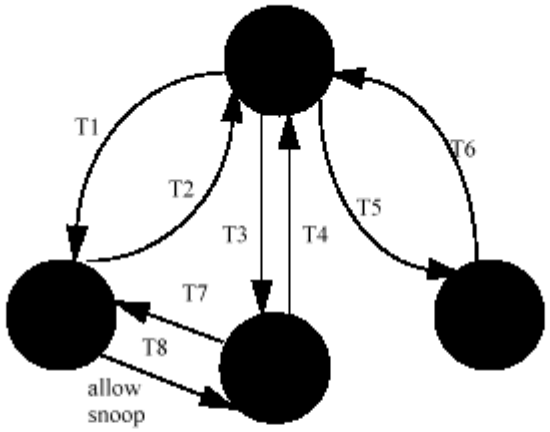
3. Execution — The execution units process instructions from their reservations stations using the operands provided from dispatch and notifies the completion stage when the instruction has finished execution. With the exception of multiply and divide integer instructions complete execution in a single cycle. FPU has three stages for processing floating-point arithmetic. The FPU stages are multiply, add and normalize. All single precision arithmetic (add, subtract, multiply and multiply / add) instructions are processed without stalls at each stage. They have a one cycle through put and a three cycle latency. Three different arithmetic instructions can be in execution at one time with one instruction completing execution each cycle. Double-precision arithmetic multiply requires two cycles in the multiply stage and one cycle in add, and one in normalize, yielding a two cycle throughput and a four cycle latency. All divide instructions require multiple cycles in the first stage for processing. The load / store unit has two reservation registers and two pipeline stages. The first stage is for effective address calculation and the second stage is for MMU translation and accessing the L1 data cache. Load instructions have a one cycle through put and a two cycle latency. In the case of an internal exception, the execution unit reports the exception to the completion pipeline stage and (except for the FPU) discontinues instruction execution until the exception is handled. The exception is not signaled until it is determined that all previous instructions have completed to a point where they will not signal an exception.
4. Completion — The completion unit retires instructions from the bottom two positions of the completion queue in program order. This maintains the correct architectural machine state and transfers execution results from the rename buffers to the GPRs and FPRs (and CTR and LR, for some instructions) as instructions are retired. If completion logic detects an instruction causing an exception, all following instructions are cancelled, their execution results in rename buffers being discarded and instructions are fetched from the appropriate exception vector.

Because the PowerPC Architecture can be applied to such a wide variety of implementations, instruction timing varies among PowerPC processors.

Power Management

The four power states for the 750FX/GX are shown in Figure 2-2 below. The three power saving modes provide different levels of power savings. Sleep mode provides the greatest power savings, followed by Nap mode and then Doze mode.

Use software to enter the 750FX/GX power saving modes by setting the appropriate control bits in the HID0 register and the Machine State Register (MSR) power management bit (MSR[POW]) as shown in Figure 2-2.



T1: HID0(Doze) = 1 and MSR[POW] transitions from MSR[POW] = 0 to MSR[POW] = 1
T2: HRESET_, SRESET_, INT_, SMI_, MCP_, DEC, PFM, machine check interrupts
T3: HID0(Nap) = 1 and MSR[POW] transitions from MSR[POW] = 0 to MSR[POW] = 1
T4:HRESET_, SRESET_, INT_, SMI_, MCP_, DEC
T5: HID0(Sleep) = 1 and MSR[POW] transitions MSR[POW] = 0 to MSR[POW] = 1
T6: HRESET_, SRESET_, INT_, SMI_, MCP_.
T7: QACK_ transitions from QACK_ = 0 to QACK_ = 1
T8: QACK_ transitions from QACK_ = 1 to QACK_ = 0

Figure 2-2 *750FX/GX Power States*

Full-On

Full-On is the default power state of the processor. The 750FX/GX is fully powered and the internal functional units are operating at full processor clock speed.

Doze

In this power saving mode, all the 750FX/GX functional units are disabled except for the Time Base/Decrementer, data cache, L2 cache, and the bus snooping logic. In Doze mode, the bus snooping logic, the data cache and the L2 cache are fully functional to maintain memory coherency. Once the Doze mode is entered, an external asynchronous interrupt, a system management interrupt, a decrementer interrupt, a reset (hard or soft), a performance monitor interrupt, thermal management interrupt or machine checks will bring 750FX/GX out of the Doze mode and into the Full-On state. In Doze mode, the PLL is required to be running and locked to SYSCLK. The transition to the Full-On state takes no more than a few processor cycles.

Nap

Further power savings can be achieved through the Nap mode in which only the Time Base and Thermal unit is operating. The processor will not enter Nap mode until QACK_ has been asserted. To maintain cache coherency, the system must de-assert QACK_ at least eight bus cycles before issuing a snoop to the processor.

Reasserting QACK_ after the snoop will put the processor back into Nap mode. 750FX/GX will wake up to the Full-On state by an external asynchronous interrupt, a system management interrupt, a decrementer interrupt, a reset (hard or soft) or a machine check input (MCP_). As in the Doze mode, the PLL is required to be running and locked to SYSCLK. The transition to the Full-On state takes no more than a few processor cycles.

Sleep

Sleep mode provides further power savings compared to the Nap mode. In sleep mode no functional units are operating and the PLL and SYSCLK input may be disabled by the system logic. The PLL can be disabled by configuring the PLL_CFG(0:4) pins into the PLL bypass mode. Note that, simply forcing SYSCLK to be static will not disable the PLL which will still be running at an undetermined frequency. When recovering from sleep mode with the PLL disabled, the system logic has to re-enable the PLL and SYSCLK first, and then assert an external asynchronous interrupt, a system management interrupt, a reset (hard or soft) or a machine check input (MCP_) after 100usec of PLL re-lock time. Select HRESET if PLL loses lock.

Power Management on page 72 and *Thermal Management* on page 74 provide information about power saving and thermal management modes for the PowerPC 750FX/GX.

Thermal Management

The PowerPC 750FX/GX's thermal assist unit (TAU) provides a way to control heat dissipation. This ability is particularly useful in portable computers, which, due to power consumption and size limitations, cannot use desktop cooling solutions such as fans. Therefore, better heat sink designs coupled with intelligent thermal management is critical for high performance portable systems.

Primarily, the thermal management system monitors and regulates the system's operating temperature. For example, if the temperature is about to exceed a set limit, the system can be made to slow down or even suspend operations temporarily in order to lower the temperature.

The thermal management facility also ensures that the processor's junction temperature does not exceed the operating specification. To avoid the inaccuracies that arise from measuring junction temperature with an external thermal sensor, PowerPC 750FX/GX on-chip thermal sensor and logic tightly couples the thermal management implementation.

The TAU consists of a thermal sensor, digital-to-analog convertor, comparator, control logic, and the dedicated SPRs described in Figure 2-1 on page 58. The TAU does the following.

- Compares the junction temperature against user-programmable thresholds.
- Generates a thermal management interrupt if the temperature crosses the threshold.
- Enables the user to estimate the junction temperature by way of a software successive approximation routine.

The TAU is controlled through the privileged **mtspr**/**mfspr** instructions to the three SPRs provided for configuring and controlling the sensor control logic, which function as follows.

- THRM1 and THRM2 provide the ability to compare the junction temperature against two user-provided thresholds. Having dual thresholds gives the thermal management software finer control of the junction temperature. In single threshold mode, the thermal sensor output is compared to only one threshold in either THRM1 or THRM2.
- THRM3 is used to enable the TAU and to control the comparator output sample time. The thermal management logic manages the thermal management interrupt generation and time multiplexed comparisons in the dual threshold mode as well as other control functions.

Instruction cache throttling provides the PowerPC 750FX/GX with overall junction temperature control by determining the interval at which instructions are fetched. This feature is accessed through the ICTC register.

Performance Monitor

Facilities are incorporated into the 750FX/GX to monitor its selected behavioral characteristics. These facilities are collectively called the performance monitor. Some motivations for incorporating a performance monitor into the 750FX/GX:

- To increase system performance with efficient software, memory hierarchy behavior must be monitored and studied in order to develop algorithms that schedule tasks (and perhaps partition them) and that structure and distribute data optimally.
- To improve processor architecture, the detailed behavior of the 750FX/GX structure must be known and understood in many software environments. Some environments may not easily be characterized by a benchmark or trace.
- The performance monitor will help system developers to bring up and debug their systems.

The Performance Monitor on the 750FX/GX processor incorporates:

- Event Counting
- Interrupt Generation

BootROM Monitor (PMON)

PMON/2000 is an Open Source, BSD style licenced, firmware system created at Opsycon AB, Sweden. It can be freely redistributed and used under the described conditions.

The first PMON/2000 version was based on PMON by Algorithmics UK, which was based on PMON for the MIPS(R) architecture.

The initial PowerPC implementation was created in 2000 on a sub-contract to RTMX Inc and later the MIPS implementation was created on a sub-contract to PMC-Sierra, Inc.

PMON/2000 On-line Documentation

The PMON/2000 BootROM Monitor (PMON/2000) permits full-featured debugging to be performed on a target board that has a serial port, 512KB of ROM, and 128KB of RAM that can be dedicated to the PROM Monitor software.

A PMON/2000 system consists of two components: A Host and a Target.

For information on configuring the PMON/2000 debugger, please visit the "PMON/2000 On-line" Web Site, http://www.pmon2000.com for reference documentation, additional information and support.

For other related activities / products / services visit.

Opsycon AB, Sweden http://www.opsycon.com

GroupBSD Open Source (RTMX) http://groupbsd.org

OpenBSD Project http://www.openbsd.org

PMON R01.00 Readme.txt

PMON Version R01.00 (PMON2000 base version 2.1), build #107

Copyright 2003-2004, GE Fanuc Embedded Systems

This document contains basic instructions and examples for usage of the PMON boot monitor. For more detailed information on PMON commands, please consult the PMON shell help (accessible by using the 'h' command).

Requirements

To use PMON, connect a NULL-modem serial cable to COM1 of the VMIVME-7050 and to a serial port on the host that provdes terminal emulation facilities. Use the following settings in the terminal emulation application:

Bits per second: 9600

Data bits: 8

Requirements (Continued)

Parity: None

Stop bits: 1

Flow control: None

Flash Device Selection

The VMIVME-7050 PowerPC board features a 512KB socketed flash and a 64MB soldered-down flash. Boards are shipped from the factory with PMON installed on the 512KB socketed flash. Switch S1, pin 1, on the VMIVME-7050 controls which flash part is read during the boot process:

Switch S1, pin 1, OFF = Boot from Socketed 512KB flash (PMON boot monitor firmware. This is the factory default)

Switch S1, pin 1, ON = Boot from soldered-down 64 MB flash

NOTE: GE Embedded Systems recommends against modifications to the contents of the 512KB flash, unless instructed otherwise.

Setting The IP Address For gtx0/NIC0

Make sure switch S1, pin 1, is set to OFF. Run PMON. At the PMON shell prompt, "PMON>", type the following command to set the IP address:

```
set ifconfig "gtx0:127.0.0.1"
```

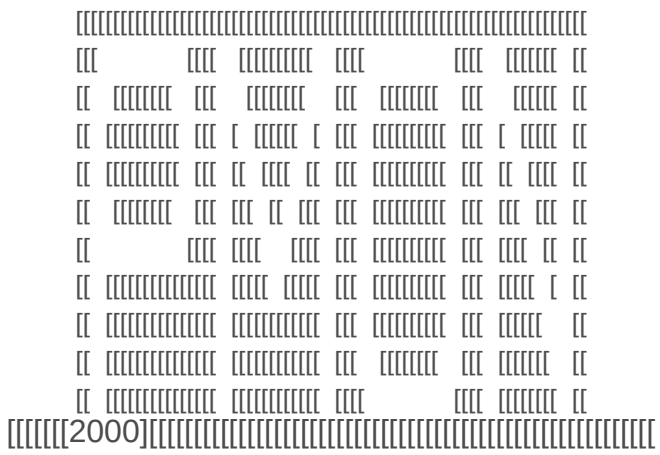
Replace the '127.0.0.1' portion of the above string with the desired IP address. The command 'set' is used for changing the values of environment variables, or for adding new environment variables. The 'ifconfig' command is the environment variable that stores the IP address for NIC 0 (gtx0). PMON automatically stores all environment variables in non-volatile RAM.

During initialization, PMON will load the environment variable table from non-volatile RAM and search for the 'ifconfig' entry. If the entry is found, NIC 0 will be initialized with the specified IP address.

Loading an Image on the Soldered-Down Flash

The 64MB flash soldered part on the VMIVME-7050 may be shipped empty from the factory, or with an operating system installed. This is dependent on ordering options. Before attempting to load an operating system, bootrom or other code on the 64MB flash, please confirm whether you want to overwrite any factory installed operating system. See *Example: Write/Erase Flash Memory* on page 87.

PMON Startup Screens (For Reference Only)



Configuration [DB64360,EB,NET,SCSI,IDE]

Version: PMON2000 2.1 (DB64360) #23: Tue Dec 2 11:43:22 CST 2003.

Supported loaders [srec, elf, bin]

Supported filesystems [net, disk, iso9660, socket, tty, ram]

This software may be redistributed under the BSD copyright.

Copyright (c) 2001-2002, Opsycon AB.

Copyright (c) 2004, VMIC

CPU PowerPC 750FX/GX @ 532.80 MHz / Bus @ 133.20 MHz

Memory size 512 KB.

```
PMON> h
                              Boot and Load
   boot  boot                                 oload  load memory from hostport
   load  load file                           netboot  boot or load from network
                                   Misc
   call  call function                         devls  list devices
  flush  flush caches                         reboot  reboot system
  flash  program flash memory                     tr  transparent mode
     rz  zmodem download
                                 Debugger
      c  continue execution                        t  trace (single step)
     to  trace (step over)                        db  delete break point(s)
      b  set break point(s)                        g  start execution (go)
    sym  define symbol                            ls  list symbols
      r  display/set register                      l  list (disassemble) memory
     bt  stack backtrace
                                   Shell
   more  paginator                                 h  on-line help
     sh  command shell                          vers  print version info
   eval  evaluate and print result              stty  set tty options
```

```
     hi  display command history          date  get/set date and time
  about  about PMON2000
                                Network
 ifaddr  Configure Network Interface      ping  ping remote host
                                Memory
    map  show memory map                     m  modify memory
      d  display memory                compare  compare memory to memory
   copy  copy memory to memory            fill  fill memory
 search  search memory                      mt  simple memory test
  cache  cache control
                                  Pci
 pcicfg  pci config space               pciscan  scan pci bus
                             Environment
    env  display variable                  set  display/set variable
  unset  unset variable(s)                eset  edit variable
PMON> h b
      b  [-s][adr]..                                      set break point(s)
                     -s    command string
PMON> h copy
   copy  from to size                                     copy memory to memory
PMON> vers
PMON: PMON2000 2.1 (DB64360) #23: Tue Dec  2 11:43:22 CST 2003
PMON> map
```

Table 2-3 Physical Address Map

Address Range	Size	Description
0000 0000 - 8000 0000	*8000 0000	SDRAM
0000 0000 - 0009 550F	0009 5510	PMON2000
0009 5510 - 000F FFFF	0006 AAF0	PMON2000 Heap
7FF8 0000 - 7FFF FFFF	0008 0000	Kernel Data
8000 0000 - F7FF FFFF	77FF FFFF	MV64360 MAP
F800 0000 - FBFF FFFF	0400 0000	Soldered Flash
FE00 0000 - FE03 FFFF	0004 0000	On Chip SRAM
FE04 0000 - FE04 FFFF	0001 0000	Marvel Chipset internal Registers
FE05 0000 - FE05 FFFF	0001 0000	32KB NVRAM / RTC
FE06 0000 - FF7F FFFF	017A 0000	Free / Unused
FE80 0000 - FFFF FFFF	0080 0000	512KB Socketed Flash
NOTE: Top of physical RAM could be 2000 0000, 4000 0000 or 8000 0000 depending on memory size option of 512KB, 1MB or 2MB.		

PMON Commands by Catagory

The PMON / 2000 Monitor (running on the Target) provides a set of powerful assembly-level debugging commands.

Downloading Files / Images	Load File (load), Boot (boot), Network Boot (netboot), Disk Boot (scsiboot), Define a Symbol (sym), List Symbols (ls),
Display / Set Registers, ------- Display / Set Memory	Display / set Registers (r), Modify Memory (m), Display Memory (d), Disassemble Memory (l), Fill Memory (fill), Copy Memory (copy), Search Memory (search), Dump Memory (dump),
Execution Control, ------- Breakpoints	Start Execution (g), Display / Set Breakpoints (b), Delete Breakpoint (db), Single Step (t / to), Back Trace (bt), Continue Execution (c), Execute Subroutine (call),
Miscellaneous and Environment Control	Help (h), About PMON (about), Enter DBX mode (debug), History (hi), Display / set Enviroment Variable (set / eset), Set Terminal Parameters (stty), Date Set / Date Display (date), Write / Erase Flash Memory areas (flash), Transparent Mode (tr), Flush the Caches (flush), Display the version number, (ver) The Command Shell, (sh) Paginator, (more) Reboot PMON. (reboot)
Diagnostics	Memory Test (mt), Network PING (ping)

Example of using PMON for Assembly-level debugging

This page shows an annotated example of using PMON to download, and then debug a sample application.

Load the application program l4003.rec (bubble sort).

```
PMON> load
Downloading from ethernet, ^C to abort
Entry address is 80000200
total = 0x1280 bytes
```

Disassemble the program.

```
PMON> l
      _start        0c0001ba jal     CpuInit                 # 0x800006e8
      _start+0x4    00000000 nop
      _start+0x8    3c01a000 lui     at,0xa000               # 40960
      _start+0xc    02018025 or      s0,s0,at
      _start+0x10   0200f809 jalr    s0
      _start+0x14   24040003 addiu   a0,zero,0x3
      _start+0x18   0200f809 jalr    s0
      _start+0x1c   24040004 addiu   a0,zero,0x4
      _start+0x20   0c0002a4 jal     clrbss                  # 0x80000a90
      _start+0x24   00000000 nop
more... q
```

Go from here to 'main'. Go must only be used once, from then on you must use the 'c' (continue) command.

```
PMON> g . main
        main        27bdffe0 addiu   sp,sp,0xffe0           # -32
```

Disassemble but only show calls.

```
PMON> l -c
        main+0xc    0c00014f jal     _puts                   # 0x8000053c
        main+0x14   0c0000bf jal     init                    # 0x800002fc
        main+0x1c   0c0000f8 jal     sort                    # 0x800003e0
        main+0x24   0c00014f jal     _puts                   # 0x8000053c
```

```
        init+0x14   0c000140 jal     malloc                # 0x80000500
        init+0x74   0c0000e8 jal     insert                # 0x800003a0
        sort+0x58   0c00011e jal     swap                  # 0x80000478
       _puts+0x38   0c00018b jal     sio_putc              # 0x8000062c
       _puts+0x4c   0c00018b jal     sio_putc              # 0x8000062c
    sio_putc+0x14   0c000170 jal     sio_init              # 0x800005c0
more... q
```

Continue until the function init.

```
PMON> c init
        init        27bdffe0 addiu   sp,sp,0xffe0          # -32
```

Continue until the address in register ra,

i.e., until this function returns.

```
PMON> c @ra
        main+0x1c   0c0000f8 jal     sort                  # 0x800003e0
```

Single step into the function sort.

```
PMON> t
        sort        8f8e8028 lw      t6,-32728(gp)         # 0xffff8028
```

Disassemble.

```
PMON> l
        sort        8f8e8028 lw      t6,-32728(gp)         # 0xffff8028
        sort+0x4    27bdffd8 addiu   sp,sp,0xffd8          # -40
        sort+0x8    11c0001f beq     t6,zero,sort+0x88     # 0x80000468
        sort+0xc    afbf0014 sw      ra,20(sp)             # 0x14
        sort+0x10   8f8f8028 lw      t7,-32728(gp)         # 0xffff8028
        sort+0x14   afa00024 sw      zero,36(sp)           # 0x24
        sort+0x18   afaf0020 sw      t7,32(sp)             # 0x20
        sort+0x1c   8fb80020 lw      t8,32(sp)             # 0x20
        sort+0x20   00000000 nop
        sort+0x24   8f190000 lw      t9,0(t8)
more... q
```

Single step.

```
PMON> t
```

```
        sort+0x4    27bdffd8 addiu   sp,sp,0xffd8          # -40
```

Single step by pressing the return key (repeat last command).

```
PMON>
        sort+0x8    11c0001f beq     t6,zero,sort+0x88     # 0x80000468
PMON>
        sort+0x10   8f8f8028 lw      t7,-32728(gp)         # 0xffff8028
PMON>
        sort+0x14   afa00024 sw      zero,36(sp)           # 0x24
PMON>
        sort+0x18   afaf0020 sw      t7,32(sp)             # 0x20
PMON>
        sort+0x1c   8fb80020 lw      t8,32(sp)             # 0x20
PMON>
        sort+0x20   00000000 nop
PMON>
        sort+0x24   8f190000 lw      t9,0(t8)
PMON>
        sort+0x28   00000000 nop
```

Disassemble.

```
PMON> l

        sort+0x28   00000000 nop
        sort+0x2c   13200012 beq     t9,zero,sort+0x78     # 0x80000458
        sort+0x30   afb9001c sw      t9,28(sp)             # 0x1c
        sort+0x34   8fa80020 lw      t0,32(sp)             # 0x20
        sort+0x38   8faa001c lw      t2,28(sp)             # 0x1c
        sort+0x3c   9509000a lhu     t1,10(t0)             # 0xa
        sort+0x40   954b000a lhu     t3,10(t2)             # 0xa
        sort+0x44   00000000 nop
        sort+0x48   0169082b sltu    at,t3,t1
        sort+0x4c   10200008 beq     at,zero,sort+0x70     # 0x80000450
more... q
```

Disassemble calls only.

```
PMON> l -c
        sort+0x58   0c00011e jal     swap                    # 0x80000478
       _puts+0x38   0c00018b jal     sio_putc                # 0x8000062c
       _puts+0x4c   0c00018b jal     sio_putc                # 0x8000062c
    sio_putc+0x14   0c000170 jal     sio_init                # 0x800005c0
     CpuInit+0x4    0c0000a8 jal     getmachtype             # 0x800002a0
     CpuInit+0x38   0100f809 jalr    t0
searching.. ^C-break!
        sort+0x28   00000000 nop
```

Continue until sort+58.

```
PMON> c sort+58
        sort+0x58   0c00011e jal     swap                    # 0x80000478
```

Single step over the function swap.

```
PMON> to
        sort+0x60   240c0001 addiu   t4,zero,0x1
```

Continue until the function swap.

```
PMON> c swap
                      swap        8c8e0004 lw      t6,4(a0)
```

Example: Set Terminal Options

stty

The stty command displays and sets terminal options.

Format

The format for this command is:

```
stty [device][-av] [baud] [sane] [term][ixany|-ixany]
[ixoff|-ixoff]
```

where:

device	is either tty0 or tty1. The default is tty0.
-a	gives a long listing showing all current settings.
-v	displays the possible choices for baud rate and terminal type.
baud	sets the baud rate.
sane	resets terminal settings to the default.
term	sets the terminal emulation type.
ixany	allows any character to restart the output.
-ixany	allows only START to restart the output.
ixoff	enables the tandem mode.
-ixoff	disables the tandem mode.

When invoking the stty command with no parameters, PMON displays the terminal type and baud rate for the tty0 port.

Functional Description

The stty command displays and sets the terminal options, such as terminal emulation type, baud rate, and ioctl settings. First, to display the current terminal type, baud rate, and ioctl settings for tty0, enter:

```
PMON> stty -a
```

To display the same information for tty1, enter:

```
PMON> stty tty1 -a
```

To change the baud rate or terminal type for tty0, simply enter the new setting after stty. Precede the new setting with tty1 to change the settings for tty1. Examples illustrating the use of this command follow.

```
PMON> stty                        Display terminal type and baud rate for tty0.
      term=tvi920 baud=9600

PMON> stty -a                     Display terminal type, baud rate, and ioctl
settings for tty0.
  term=tvi920 baud=9600 canon echo echoe onlcr icrnl istrip ixon
  erase=^H stop= start=^Q eol=^J eol2=^C vintr=^C

PMON> stty 9600                   Set baud rate for tty0 to 9600.

PMON> stty -v                     List available baud rates, terminal types and
current baud.
  Baud rates: 50 75 110 134 200 150 300 600 1200
            1800 2400 4800 9600 19200 38400
  Terminal Types:
          tvi920      vt100
            Baud:     9600
PMON> stty tvi920                 Set terminal type for tty0 to tvi920.

PMON> stty tvi920 9600            Set terminal type and baud rate for tty0 to
tvi920 and 9600 baud.

PMON> stty tty1 sane              Reset ioctl settings for tty1.

PMON> stty tty1 19200             Set tty1 to 19200 baud.
```

Example: Write/Erase Flash Memory

Flash

The flash command provides for programming, erasing and copying flash memory areas.

The soldered-down 64MB flash part on the VMIVME-7050 can be shipped from the factory, with or without an operating system installed depending on ordering options. Before attempting to load an operating system, bootrom, or other code on the 64MB flash, please confirm whether you want to overwrite any factory installed operating system.

Format

The format for the command is:

```
flash [-q] [-e arg] [args ...]
```

where:

<none>	gives a list of available flash areas and types
-q	gives a list of available flash areas and types
-e base_addr	erases the flash area on that base address.
base_addr size from_addr	writes the flash base_addr memory for size bytes using from_addr memory starting point.

Flash the 64 MB part using PMON:

1. Ensure switch S1, pin 1, is in the OFF position.
2. Power up the system. PMON will initialize and present the 'PMON>' shell prompt.
3. Make sure PMON has been configured with a valid IP address.
4. Configure a network host TFTP server. The VMIVME-7050 will load the image from your network host using the TFTP protocol.
5. From the PMON shell, enter the following command:

   ```
   flash
   ```

Each area in the flash device will be identified, and is dependent on the system implementation.

These areas are useable for storage or for bootable images. The -q (Query) option will list each system dependent flash ROM area. The base address and size is listed, along with each flash area configuration, sector size (if supported) and the actual Flash device Manufacturing ID value (if reported). Example:

```
PMON> flash -q
Available FLASH memory
Start   Size     Width Sectorsize Type
ff000000 04000000 4*8   00020000   AM29LV256M
fff00000 00080000 1*8   00010000   AM29LV040
PMON>
```

The -e (Erase) option code calculates the sector from the flash address you enter, and the number of sectors are the size divided by sectorsize (rounded up).

Release Notes

PMON

Version R01.00 (PMON / 2000 base version 2.1), build #107

May 2004

PMON / 2000 is an open source, BSD style licensed, firmware system supplied by Opsycon AB, Sweden. GE Embedded Systems used the PMON / 2000 project as the basis for this version PMON (R01.00). Not all PMON / 2000 features are supported by the GE Fanuc Embedded Systems' version of PMON. However, an effort was made to support critical and most highly useful features. Additional features may be supported in future releases of PMON.

For detailed information regarding the original PMON / 2000 project, please reference:

http: / / www.opsycon.com

http: / / www.pmon2000.com

http: / / www.openbsd.org

If you have specific questions about PMON, you may contact GE Fanuc Embedded Systems Customer Care:

Customer Care:

Phone: 1-800-240-SRVC (7782)
FAX: (256) 650-7245
Customer Care E-mail: support.embeddedsystems@gefanuc.com

GE Embedded Systems - A GE Fanuc Company

12090 South Memorial Parkway
Huntsville, AL 35803-3308
1-800-322-3616

http: / / www.vmic.com

http: / / www.gefanuc.com

What's New

This is the initial release of PMON for GE Fanuc Embedded Systems' PowerPC boards. PMON supports the following PowerPC board:

VMIVME-7050

For examples of how to use PMON with supported processors, please reference the readme.txt file distributed with PMON, and this product manual.

Notes/ Errata

1. Decoding of ELF format images does not work properly. As a workaround, the gnu ‘objcopy’ utility may be used to convert an ELF image to another format, such as s-record or binary. If you are using a toolset other than gnu, there may be a similar utility available to you. This issue is scheduled for resolution in a future release of PMON.
2. There is an anomaly in the flash sector erase routine within PMON. This does not impact usage of the ‘flash –e’ command. ‘flash –e’ works properly. However, the following commands, and respective option, internally call the imperfect flash sector erase routine:

   - boot, when the ‘-f’ option is specified
   - netboot, as it calls ‘boot –f’ internally
   - load, when the ‘-f’ option is specified

Otherwise, the commands should work as described. The ‘load’ and ‘flash’ commands may be used to load an image into Flash memory.

3. Flash, as opposed to using the ‘load’ command with the ‘-f’ option. This issue is scheduled for resolution in a future release of PMON.
4. POST (power-on self-test) does not exercise a large portion of the board feature set. Enhancements to POST are planned for future releases of PMON. Please refer to the readme.txt file for more information concerning POST.
5. The following additional items are not support in PMON R01.00:

   - Memory sizes other than 512KB
   - VMEbus
   - File system(s) on flash
   - Voltage and temperature monitor
   - I^2C bus
   - COM2
   - Baud rates other than 9600 Kb / s

Please note the current limitations do not preclude the use of these features within operating environments other than PMON.

PMON R01.00 Release Notes - End

Maintenance

If a VMIC product malfunctions, please verify the following:

1. Software resident on the product
2. System configuration
3. Electrical connections
4. Jumper or configuration options
5. Boards are fully inserted into their proper connector location
6. Connector pins are clean and free from contamination
7. No components or adjacent boards were disturbed when inserting or removing the board from the chassis
8. Quality of cables and I/O connections

If products must be returned, contact VMIC for a Return Material Authorization (RMA) Number. **This RMA Number must be obtained prior to any return**.

VMIC Customer Care is available at: 1-800-240-7782.
Or E-mail us at **support.embeddedsystems@gefanuc.com**.

Maintenance Prints

User level repairs are not recommended. The drawings and diagrams in this manual are for reference purposes only.

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

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

ARTISAN® TECHNOLOGY GROUP