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 / VMIC VMIVME-7489-125* at our website: ***Click HERE***

VMIVME-7489
486 PC/AT VMEbus CPU with PC/104

Product Manual

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

500-007489-000 Rev. C

COPYRIGHT AND TRADEMARKS

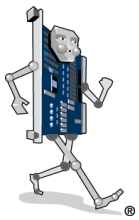
© Copyright December 1999. The information in this document has been carefully checked and is believed to be entirely reliable. While all reasonable efforts to ensure accuracy have been taken in the preparation of this manual, VMIC assumes no responsibility resulting from omissions or errors in this 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, Instant OPC wizard logo, IOMax™, IOWorks Access, IOWorks Foundation, IOWorks man figure, 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*, *VMEmanager*, *VMEmonitor*, VMEnet, VMEnet II, and *VMEprobe* are trademarks. 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)



(Instant OPC wizard logo)



(IOWorks man figure)

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

ActiveX is a trademark and 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.

RECORD OF REVISIONS

REVISION LETTER	DATE	PAGES INVOLVED	CHANGE NUMBER
A	11/01/95	Release	95-0645
B	05/01/96	Cover and page ii	96-0314
C	12/08/99	Reformat and change phone number area code to 256	99-0963

Table of Contents

Table of Contents 5

List of Figures 9

List of Tables 11

Safety 13
VMIC Safety Summary 14
Safety Symbols 16

Chapter 1 - Introduction 19
About this Manual 20
Product Family 21
References 23
PC/AT Features 24
VMEbus Features 27

Chapter 2 - Installation and Setup 29
Unpacking Procedures 30
Hardware Setup 31
Installation 34
Front Panel Connectors 36
PC/104 Expansion Site 37
BIOS Setup 38
Configuring Operating Systems 39
General Rule Regarding Operating Systems 39
Configuration Examples 40
Configuring Windows for the VMIVME-7489 40

Chapter 3 - PC/AT Functions 43
CPU Register Model 44
Physical Memory 45
Memory and Port Maps 46
Memory Map 46
I/O Port Map 46
PC/AT Interrupts 51
Enhanced I/O Ports 56
Serial Ports 56
Parallel Port 56
Video Graphics Adapter 58
PC/104 Expansion Site 59

Chapter 4 - VMEbus Functions 61
VMEbus Access 62
Byte Ordering 64
Byte Swapping 64
The Byte Swapping Problem Defined 65
Byte Swapping and the VMEbus 66
VMIVME-7489 Byte Swapping Hardware 67
Interrupt Handling 69
Nonmaskable Interrupts 69
Soft NMI 69
ACFAIL, SYSFAIL, and BERR NMIs 70
NMI Processing 70
Maskable Interrupts 71
Condition Polling 72
Interrupter 74
Requester 75
System Controller 76
System Registers 77
Register Map 77
Register Details 78
Product ID Register 78
VMEbus Access Control Register 78
VMEbus Access Control Register: Status LED bit 79
VMEbus Access Control Register: ROR bit 79
VMEbus Access Control Register: SysFail bit 79

VMEbus Access Control Register: Big Endian bit 80
VMEbus Access Control Register: VME Enable bit 80
VMEbus Standard Access Register 81
VMEbus Extended/AM Access Register 82
Interrupt Mask Register 83
Clear Condition Register 84
DONE Status-ID Register 85
Group Interrupt Status Register 85
Soft NMI Select Register 86
IRQ Status-ID Register 86
EOI Command Register 86
Interrupt Level Select Register 87
IRQ* Status Register 88
Interrupter Status-ID Register 88

Appendix A - Connector Pinouts 89
Ethernet Connector Pinout 91
Floppy Drive Connector Pinout 92
IDE Hard Drive Connector Pinout 93
Keyboard Connector Pinout 94
PC/104 Connector Pinout 95
Printer Connector Pinout 97
Serial Connector Pinout 98
Video Connector Pinout 99
VMEbus Connector Pinout 100

Maintenance 103
Maintenance Prints 103

List of Figures

Figure 1-1 VMIVME-7489 Board View. 21
Figure 1-2 VMIVME-7489 Partial Block Diagram. 25
Figure 1-3 VMIVME-7489 System VMEbus Functions 27

Figure 2-1 I/O Port and Jumper Locations 31
Figure 2-2 PC/104 Mechanical Connection 37

Figure 3-1 80486 CPU Block Diagram 44
Figure 3-2 Interrupt Logic Diagram. 55

Figure 4-1 VMIVME-7489 VMEbus Interface Block Diagram 62
Figure 4-2 Byte Relationships Using the Little-Endian 80486 65
Figure 4-3 Byte Relationships Using the Big-Endian 68040. 66
Figure 4-4 80486-to-VMEbus Data Byte Lanes 67
Figure 4-5 Interrupt Response Flowchart 72

Figure A-1 VMIVME-7489 Connector Locations 90
Figure A-2 Ethernet Connector Pinout 91
Figure A-3 Floppy Drive Connector Pinout 92
Figure A-4 IDE Hard Drive Connector Pinout 93
Figure A-5 Keyboard Connector Pinout 94
Figure A-6 PC/104 Connector Diagram 95
Figure A-7 Printer Connector Pinout 97
Figure A-8 Serial Connector Pinouts 98
Figure A-9 Video Connector Pinout 99
Figure A-10 VMEbus Connector Diagram 100

List of Tables

Table 1-1 PC/AT I/O Features 26

Table 2-1 VMIVME-7489 Jumper Functions and Settings. 32
Table 2-2 VMEbus Window Addresses 39

Table 3-1 VMIVME-7489 Memory Address Map 46
Table 3-2 VMIVME-7489 I/O Address Map 47
Table 3-3 PC/AT Hardware Interrupts 51
Table 3-4 PC/AT Interrupt Vector Table 52
Table 3-5 Parallel Port Modes 56
Table 3-6 Common Supported Graphics Video Resolutions 58

Table 4-1 VMEbus Byte Assignment to the Data Lines 64
Table 4-2 Byte Swap Modes 68
Table 4-3 VMIVME-7489 VMEbus Register Map 77
Table 4-4 Valid VMEbus AM Codes 82
Table 4-5 IRQ Level IPL Values 87
Table A-6 PC/104 Connector Pinout 95
Table A-7 VMEbus Connector Pinout 100

Safety

Contents

VMIC Safety Summary 14
Safety Symbols 16

VMIC Safety Summary



Note 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. VME Microsystems International Corporation 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 VME Microsystems International Corporation for service and repair to ensure that safety features are maintained.

Dangerous Procedure Warnings

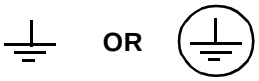
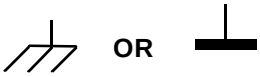
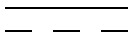
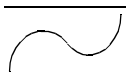
Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

W A R N I N G

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

Safety Symbols

General definitions of safety symbols used in this manual:

Symbol	Description
	Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the system.
	Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts are so marked).
	Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.
	Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. Before operating the equipment, terminal marked with this symbol must be connected to ground in the manner described in the installation (operation) manual.
	Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.
	Alternating current (power line).
	Direct current (power line).
	Alternating or direct current (power line).

Symbol	Description
	The WARNING sign denotes a hazard. It calls attention to a procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in injury or death to personnel.
	The CAUTION sign denotes a hazard. It calls attention to an operating procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the system.
	The Note sign denotes important information. It calls attention to a procedure, a practice, a condition or the like, which is essential to highlight.

CHAPTER

1

Introduction

Contents

About this Manual 20
Product Family 21
References 23
PC / AT Features. 24
VMEbus Features 27

Introduction to the VMIVME-7489

The VMIVME-7489 single-board microcomputer is a complete IBM PC / AT 80486 compatible microcomputer with the additional benefits of Eurocard construction and full compatibility with the VMEbus Specification Rev. C.1. The VMIVME-7489 also has an industry-standard PC / 104 expansion site.

VMIC's VMIVME-7489 has two operating modes: a standard PC / AT compatible mode, and a VMEbus controller mode. Upon powerup it functions as a standard PC / AT. It executes a PC / AT-type power-on self-test, then boots up MS-DOS, Windows, OS / 2, NextStep, XENIX, or any other PC / AT compatible operating system. Its keyboard and video console interaction with the user is typical of a PC / AT. This PC / AT mode of the VMIVME-7489 is discussed in Chapter 3, *PC/AT Functions* on page 43 of this manual.

After booting, the VMIVME-7489 may take on the additional functions of a VMEbus controller and interact with other VMEbus modules. The VMEbus controller functions are available by programming the VMIVME-7489's VMEbus interface registers according to Chapter 4, *VMEbus Functions* on page 61 of this manual.

With VMIC's VMIVME-7420 VMEaccess software, the VMIVME-7489 programmer may quickly and easily control VMEbus slave boards simply by linking to a library of VMIVME-7489 VMEbus interrupt and control functions.

About this Manual

Because this product bridges the traditionally divergent worlds of Intel-based PCs and Motorola-based VMEbus controllers, some confusion over *conventional* notation and terminology may exist. We have made every effort 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 *double-word*.
- 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 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, e.g., $C000, $E000, etc. When using this formula, make sure to round the offset calculation properly!

Product Family

Figure 1-1 shows a simplified view of the VMIVME-7489 board. The VMIVME-7489 is one member of VMIC's line of PC / AT compatible VMEbus controllers, all of which combine a standard PC / AT architecture with the ability to control VMEbus slave boards.

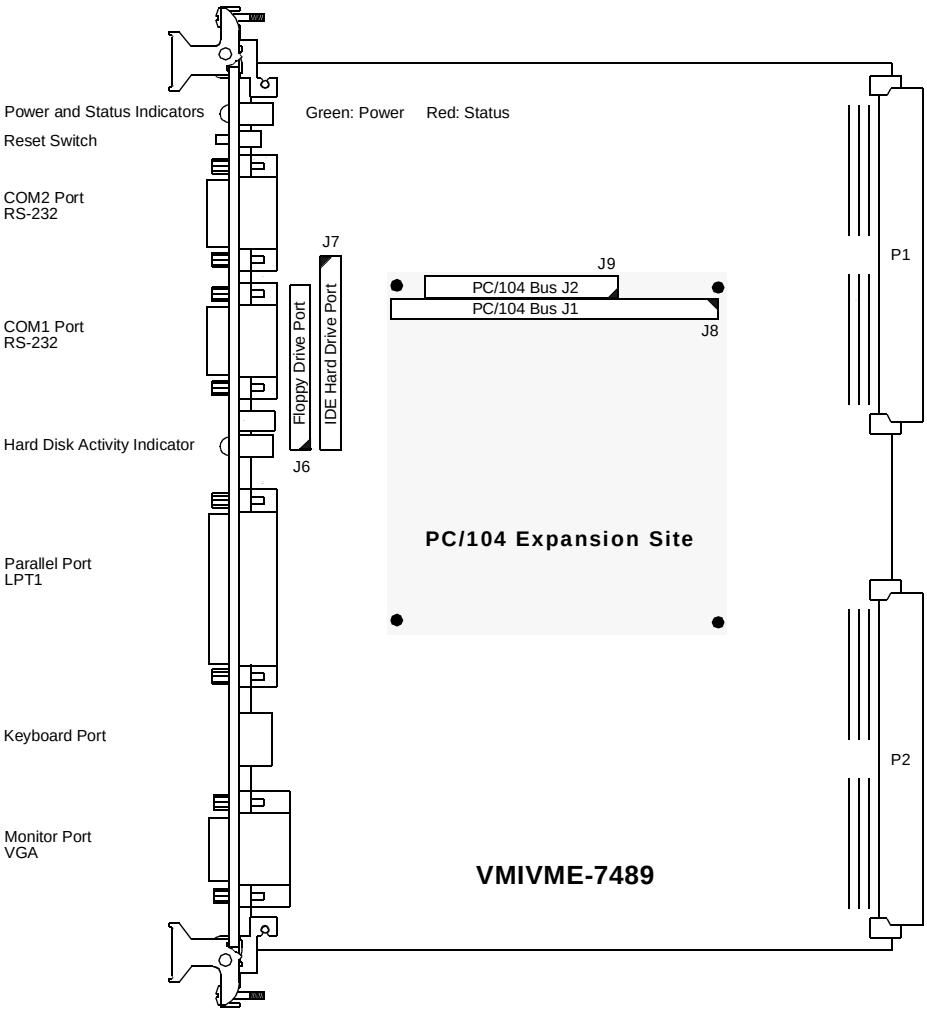


Figure 1-1 VMIVME-7489 Board View

The VMIVME-7486 is VMIC's baseline PC / AT compatible VMEbus controller and includes such features as standard serial and parallel ports, 16 Mbyte RAM capacity, and super VGA video.

The VMIVME-7489 is similar to the VMIVME-7486 PC / AT compatible VMEbus controller, but adds external cache memory, high-speed 16550-compatible serial ports, an enhanced bidirectional parallel port, up to 32 Mbytes of RAM, and a PC / 104 expansion site.

The VMIVME-7487 does not contain the VMIVME-7489's enhanced I / O, but adds full multiprocessing support with mailbox registers, dual-ported memory, and a master / slave VMEbus interface chip.

VMIC also has other support products for the PC / AT compatible VMEbus controller line. The VMIVME-7450 is a dual-slot module which holds one 3.5 inch floppy drive and one 3.5 inch hard drive. The VMIVME-7451 is a single slot module which holds one 3.5 inch floppy drive and up to two 2.5 inch hard drives. The VMIVME-7432 is a PC / 104 module with a VMEbus 6U form factor that allows a half-length PC ISA bus expansion board to be used in a VMEbus chassis along with the VMIVME-7489. The VMIVME-7432 is also compatible with the VMIVME-7450.

VMIC's VMIVME-7420 VMEaccess software includes a linkable library of functions to simplify the integration and development of high-performance VMEbus systems using any VMIC PC / AT compatible VMEbus controller. The library includes functions for such low-level chores as setting up VMEbus data transfers, processing interrupts and conditions, and handling errors. VMEaccess is highly recommended for programmers who plan to use any VMIC PC / AT compatible processor for VMEbus control.

References

For the most up-to-date physical description and specifications for the VMIVME-7489, please refer to VMIC specification number 800-007489-000.

There are many books widely available on the subject of general PC / AT use and programming. Some reference sources which may be particularly helpful in using or programming the VMIVME-7489 are listed below.

i486 Microprocessor Programmer's Reference Manual
and the *Intel 486DX Microprocessor Data Book*
Intel Corporation
Literature Sales Dept.
P.O. Box 58130
Santa Clara, CA 95052-8130

VMEbus Specification Rev. C1
and *The VMEbus Handbook*
VITA - VFEA International Trade Association
10229 N. Scottsdale Road, Suite B
Scottsdale, AZ 85253

IEEE P996.1 Standard for Compact Embedded-PC Modules
PC / 104 Consortium
990 Almanor Avenue
Sunnyvale, CA 94086

BIOS Setup User's Manual
Document Number 520-000085-000
VME Microsystems International Corporation
12090 South Memorial Parkway
Huntsville, AL 35803-3308
PH: (256) 880-0444
(800) 322-3616
FX: (256) 882-0859

PC/AT Features

The VMIVME-7489 performs all the functions of a standard IBM PC / AT motherboard with the following features:

- Single-Slot 6U Size
- High-performance 80486 processor
  Available in speeds from 33 to 100 MHz
  Standard 8 Kbyte internal cache (DX4-16 Kbyte internal cache)
  High-performance math coprocessor on 80486DX versions
- Up to 32 Mbytes of RAM
- Optional 256 Kbyte or 1 Mbyte high-speed external level 2 write-back cache
- Super VGA video
  1 Mbyte of video DRAM
  Scan resolutions up to 1280 x 1024, noninterlaced
  Color resolutions up to 16.5 million (TruColor)
- Battery-backed clock / calendar
- Front panel reset switch
- On-board ports for keyboard, IDE hard drive, floppy drive, high-speed serial, and enhanced parallel I / O
- PC / 104 expansion site for industry-standard modules
- Front panel “vital sign” indicators
  (power, programmable status, and hard drive activity)

The VMIVME-7489 also supports standard PC / AT I / O features such as those listed in Table 1-1 on page 26. Figure 1-2 on page 25 also shows a partial block diagram of the VMIVME-7489, emphasizing the I / O features.

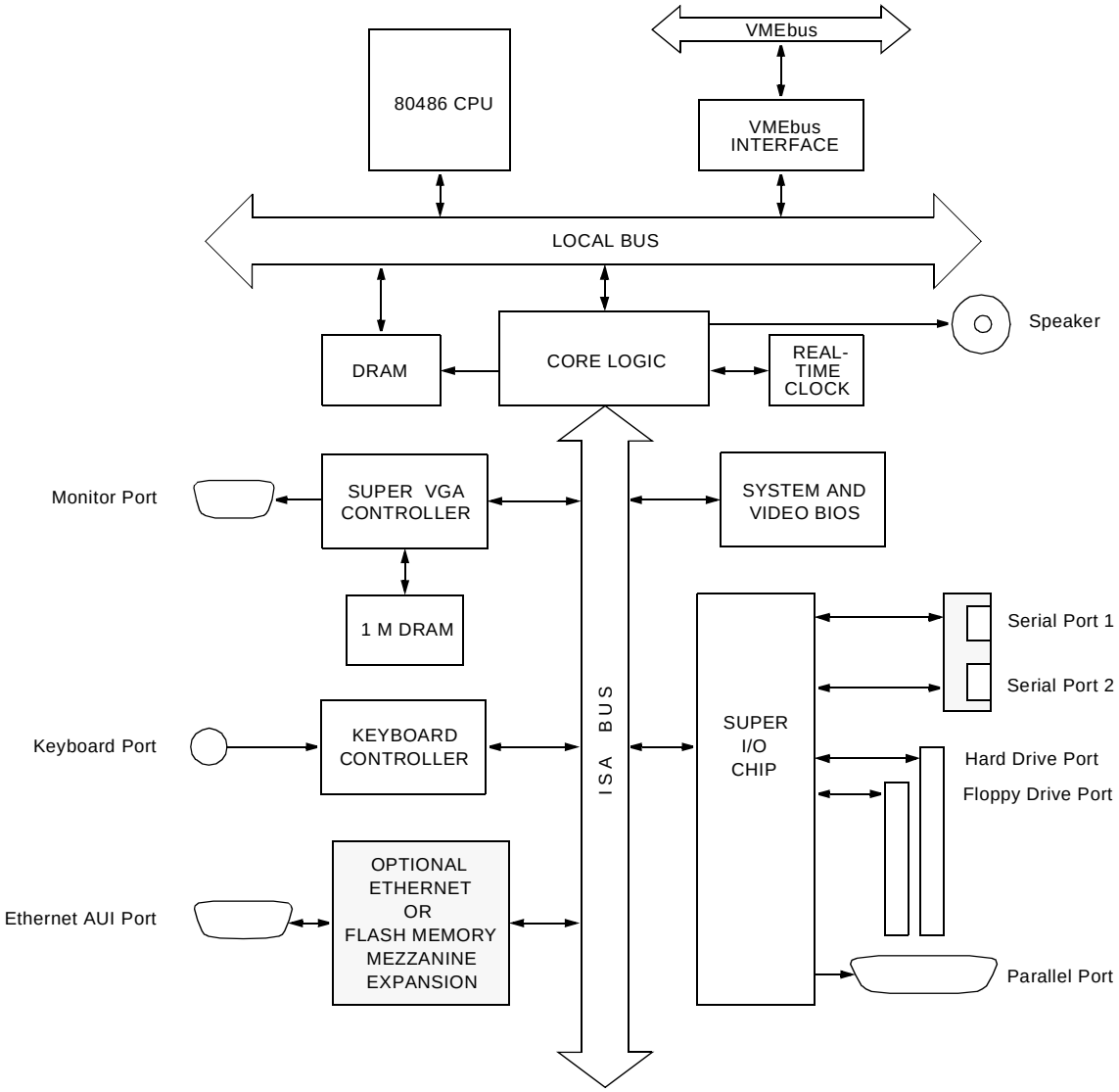


Figure 1-2 VMIVME-7489 Partial Block Diagram

Table 1-1 PC/AT I/O Features

I/O FEATURE	MS-DOS IDENTIFIER	PHYSICAL ACCESS
Two High-Speed Serial Ports (16550-compatible RS-232C)	COM1, COM2	Front Panel DB9P (male) X 2
One Enhanced Bidirectional Parallel Port (IEEE-1284 ECP / EPP compliant)	LPT1 (or LPT2)	Front Panel DB25S (female)
AT-Style Keyboard Controller with PS / 2-Style Adapter	KBD	Front Panel PS / 2-Style Mini-DIN Circular (female)
Real-Time Clock / Calendar with Battery and Watchdog Timer	Date, Time	
Super VGA Video Controller with 1 Megabyte DRAM	Display	Front Panel DB15HD High Density (female)
Floppy Disk Controller (2 drives maximum)	Drives A, B	34-pin Header Flat Cable type
IDE Fixed Disk Controller (2 drives maximum)	Drives C, D	40-pin Header Flat Cable type
Hardware Reset	Cold Boot	Front Panel Pushbutton
IBM / PC Sound	Beep	On-Board Small Speaker
LED Indicators	Power, Status, and Hard Drive Activity	Front Panel

VMEbus Features

In addition to its PC / AT functions, the VMIVME-7489 has the following VMEbus features:

- Single-slot, 6U-height VMEbus board
- Complete 6-line Address Modifier (AM-Code) programmability
- 32-bit data interface with 3-way byte / word swap in hardware
- User-configured interrupter and interrupt handler
- System Controller mode (jumper enabled) as level 3 single-level arbiter
- VMEbus BERR* 10 µs bus error timer (jumper enabled)
- Complete VMEbus access through
  Real-mode memory window, or
  Protected-mode linear addressing

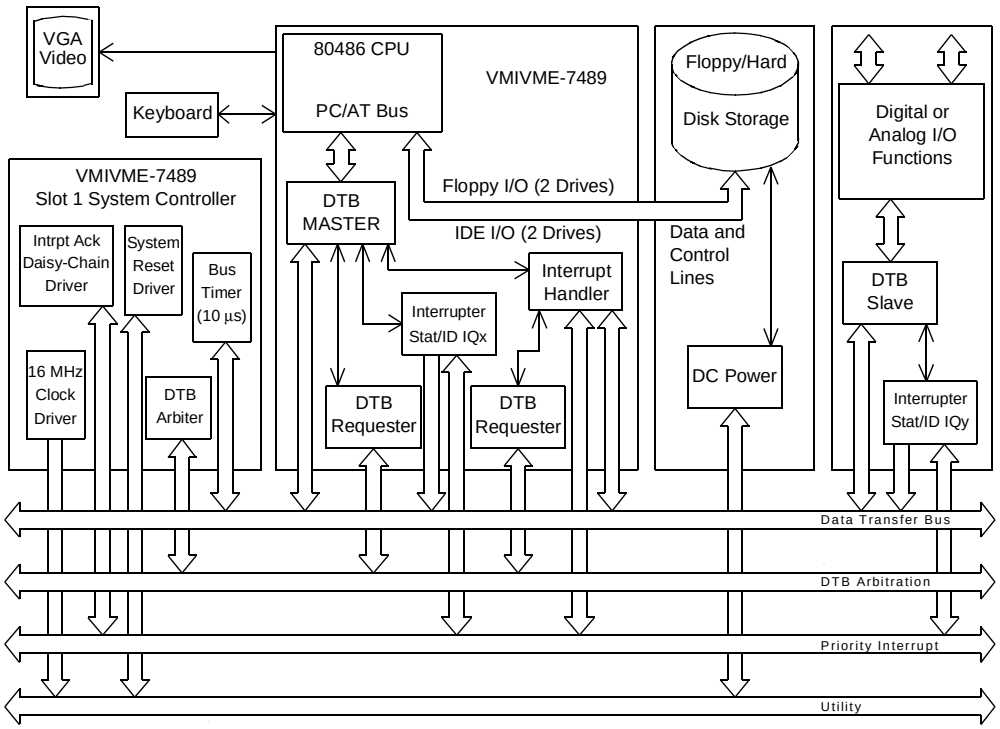


Figure 1-3 VMIVME-7489 System VMEbus Functions

Figure 1-3 shows the VMIVME-7489 functions in a typical VMEbus system. The VMIVME-7489 is a versatile single-board solution for VMEbus control with familiar PC / AT operation.

CHAPTER
2

Installation and Setup

Contents

Introduction 29
Unpacking Procedures 30
Hardware Setup 31
Installation 34
Front Panel Connectors 36
PC/104 Expansion Site 37
BIOS Setup 38
Configuring Operating Systems 39

Introduction

This chapter describes unpacking, inspection, hardware jumper settings, connector definitions, installation, system setup, and operation of the VMIVME-7489.

Unpacking Procedures



CAUTION Some of the components assembled on Vmic's 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.

Upon receipt, 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 VMIC together with a request for advice concerning the disposition of the damaged item(s).

Hardware Setup

The VMIVME-7489 has been tested for system operation and shipped with factory-installed header jumpers. Figure 2-1 illustrates the physical location of the user-configurable jumpers and connectors on the board. Table 2-1 on page 32 lists each jumper designator, its function, and the factory-installed default configuration.

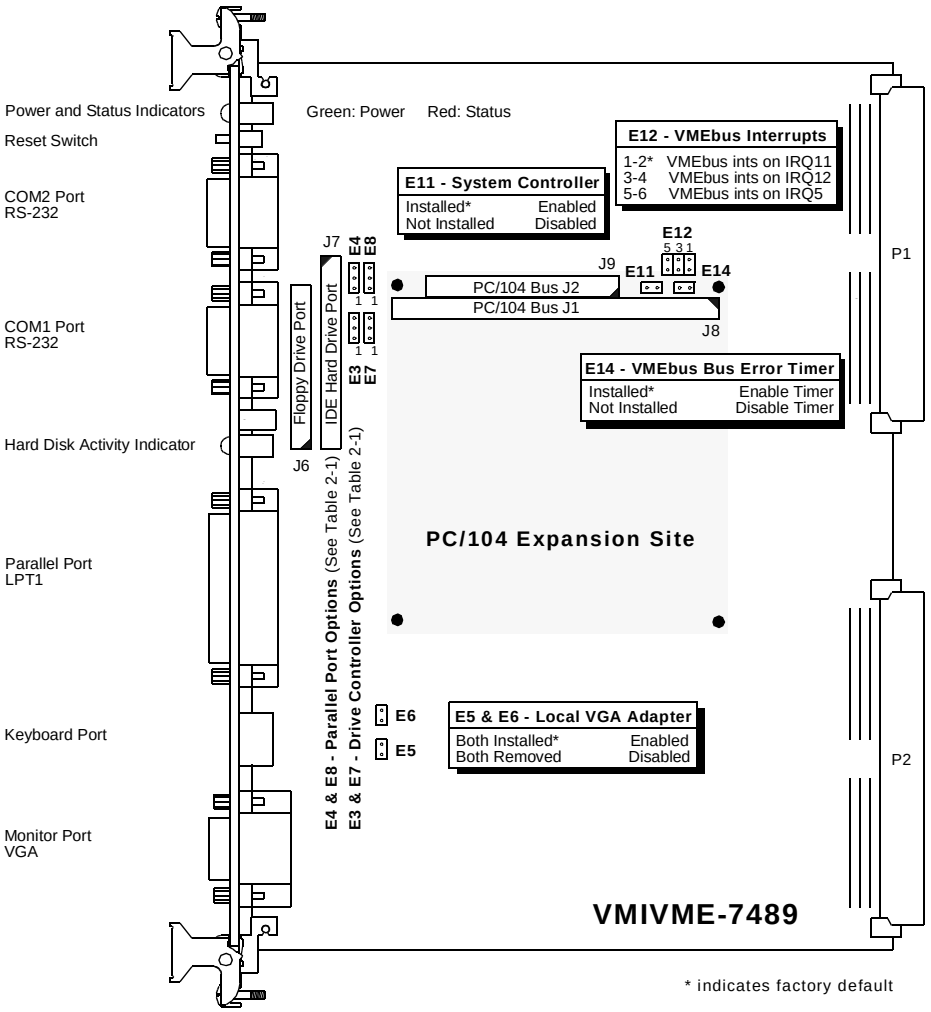


Figure 2-1 I/O Port and Jumper Locations

Table 2-1 VMIVME-7489 Jumper Functions and Settings

JUMPERS	FUNCTIONS	SETTINGS			
E3 & E7	Select Drive Controller Options	Jumper E7	Jumper E3	Floppy Disk Controller	IDE Controller
		2-3	2-3	Disabled	Disabled
		2-3	1-2	I/O Address $3F0-$3F7	Disabled
		1-2	2-3	I/O Address $370-$377	I/O Address $170-$177
		1-2 (Default)	1-2 (Default)	I/O Address $3F0-$3F7	I/O Address $1F0-$1F7
E4 & E8	Select Parallel Port Options	Jumper E4	Jumper E8	Parallel Port Address	Parallel Port Interrupt
		2-3	2-3	Disabled	Disabled
		2-3 (Default)	1-2 (Default)	I/O Address $378-$37F (Standard LPT1)	IRQ7
		1-2	2-3	I/O Address $278-$27F (Standard LPT2)	IRQ5
		1-2	1-2	I/O Address $3BC-$3BF (Alternate LPT1*)	IRQ7
E5 & E6	Enable/Disable Local VGA Adapter			Both Installed = Enabled (Default) Both Removed = Disabled	
E11	Enable/Disable System Controller			Installed = Enabled (Default) Removed = Disabled	
E12	Select VMEbus interrupts through either IRQ5, IRQ11, or IRQ12 of the PC/AT interrupt controller			1-2 = IRQ11 (Default) 3-4 = IRQ12 5-6 = IRQ5	
E14	Enable/Disable VMEbus Bus Error 10 µs Timeout Timer			Installed = Enabled (Default) Removed = Disabled	

Note

Notes

* The Alternate LPT1 address at I/O $3BC cannot be configured as an EPP port.
ANY OTHER JUMPER LOCATIONS ARE RESERVED FOR VMIC USE ONLY AND SHOULD NOT BE ALTERED FROM THE FACTORY DEFAULT SETTINGS.

Installation



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

The VMIVME-7489 conforms to the VMEbus physical specification for a 6U x 4HP dual Eurocard (dual height, single-slot width). It can be plugged directly into any standard chassis accepting this type of board.

Follow these steps to get the VMIVME-7489 up and running:

1. Make sure power to the equipment is off.
2. If a drive module or other expansion module such as the VMIVME-7450 is to be used, connect it to the controller first.
3. Choose a chassis slot. The VMIVME-7489 may be attached to a dual P1/P2 VMEbus backplane or a single P1 backplane. A single P1 connection supplies enough power and allows 16-bit data transfers and 24-bit addressing, but for 32-bit data transfers and 32-bit addressing a dual P1/P2 backplane is required.

   If the VMIVME-7489 is to be the VMEbus system controller, choose the first VMEbus slot and make sure jumper E11 is installed. If some other board is the VMEbus system controller, choose any slot *except* slot one and make sure jumper E11 is removed.
4. Insert the VMIVME-7489 and its attached expansion modules into the chosen VMEbus chassis slot (expansion modules should fill the slots immediately adjacent to the VMIVME-7489). While ensuring that the boards are properly aligned and oriented in the supporting board guides, slide the boards smoothly forward against the mating connector until firmly seated.
5. Connect all needed peripherals to the front panel. Each connector is clearly labelled on the front panel, and detailed pinouts are in Appendix A, *Connector Pinouts* on page 89. Minimally, a keyboard and a monitor are required.

6. Apply power to the system. Several messages are displayed on the screen, including names, versions, and copyright dates for the various BIOS modules on the VMIVME-7489. Among the screen messages should be a VMIC product identifier such as:

```
VME Microsystems International Corp.
VMIVME-7489
```

7. If a drive module was installed, the BIOS Setup program must be run to configure the drive types. The procedure varies according to the BIOS manufacturer. See the procedure for Quick BIOS Setup in the *BIOS Setup User's Manual* to properly configure the system.

8. If a drive module is present, install the operating system according to the manufacturer's instructions. See *Configuring Operating Systems* on page 39 for assistance customizing popular operating systems such as MS-DOS and Windows for the VMIVME-7489.

Front Panel Connectors

Front panel connections for the parallel, video, and serial ports are typical for any PC/AT and are clearly labelled. The keyboard connector is a standard mini-DIN PS/2 style connector; an adapter is supplied to connect a keyboard with a larger connector to the VMIVME-7489. See Appendix A, *Connector Pinouts* on page 89 for connector pinouts and orientation.

PC/104 Expansion Site

Expansion boards that are PC / 104 compatible install directly to the VMIVME-7489's PC / 104 Expansion Site (see Figure 2-1 on page 2-31). Figure 2-2 illustrates the mechanical connection between the VMIVME-7489 and a PC / 104 board. Note that the attachment hardware shown (the nuts, screws, and standoffs) should be included with the PC / 104 board and are not included with the VMIVME-7489.

Configuration information for PC / 104 accessories should come with the board. The most common problems installing PC / AT accessories usually involve either an I / O addressing conflict or an interrupt conflict. See the I / O Port Map and the discussion regarding PC / AT Interrupts in Chapter 3, *PC/AT Functions* on page 43 to determine port and interrupt configurations compatible with the VMIVME-7489.

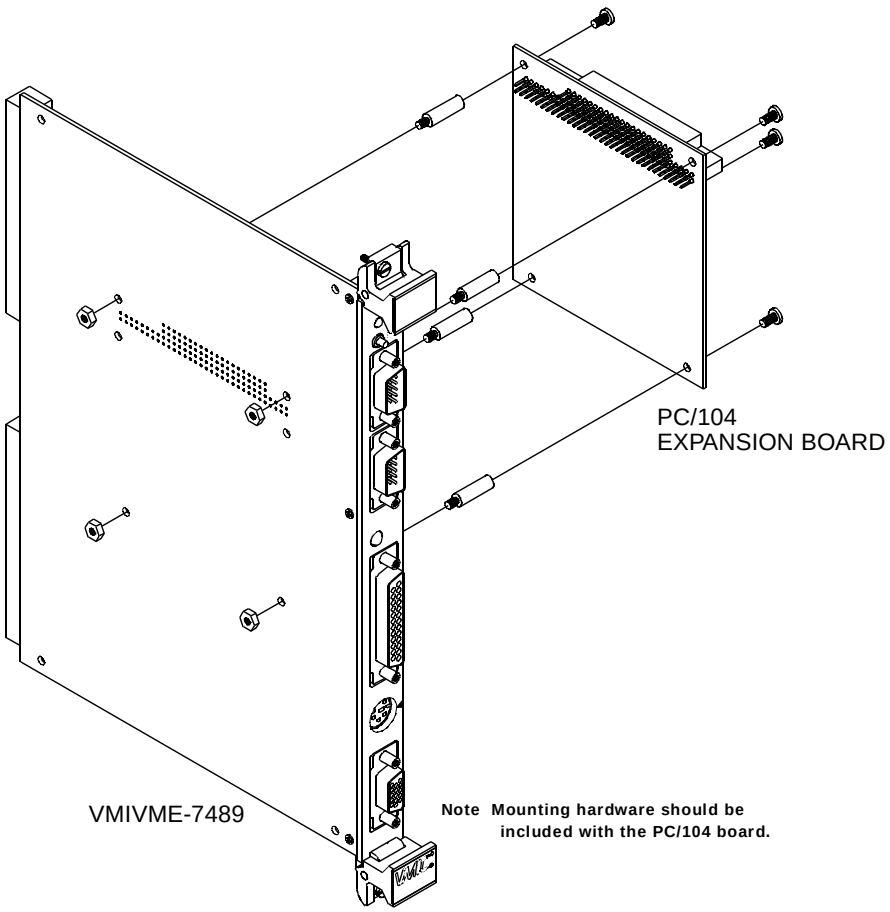


Figure 2-2 PC/104 Mechanical Connection

BIOS Setup

The VMIVME-7489 has an on-board BIOS Setup program that controls many configuration options. These options are saved in a special nonvolatile, battery-backed memory chip and are collectively referred to as the board's "CMOS configuration". The CMOS configuration controls many details concerning the behavior of the hardware from the moment power is applied.

The VMIVME-7489 is shipped from the factory with no hard or floppy drives configured in CMOS. The BIOS Setup program must be run to configure the specific drives attached. It is recommended that the user follow the procedure for Quick BIOS Setup in the *BIOS Setup User's Manual* to properly configure the system.

Configuring Operating Systems

The VMIVME-7489 is capable of running many popular operating systems, including MS-DOS, PC-DOS, DR-DOS, Microsoft Windows and Windows NT, IBM's OS/2, UNIX versions designed to run on Intel microprocessors, and UNIX variants such as XENIX. In general, any operating system designed to run on an Intel-based PC should work well on the VMIVME-7489.

The operating systems should generally be installed according to the publisher's instructions, preferably with VMEbus access disabled (see Chapter 4, *VMEbus Functions* on page 61). After installation, however, certain modifications to a standard installation of any of these operating systems may be necessary due to the special nature of the VMIVME-7489 as a combination PC and VMEbus controller.

General Rule Regarding Operating Systems

The VMIVME-7489 has two VMEbus Windows that should be off-limits to the operating system: the Real Mode VMEbus Window and the Protected Mode VMEbus Window.

The most important modification that should be made to any operating system running on the VMIVME-7489 is to exclude the two VMEbus Windows from the operating system's memory map.

By excluding the VMEbus Windows, the operating system is prevented from inadvertently scanning or accessing addresses within the VMEbus Windows. This prevents strange activity on the VMEbus, since any access within a VMEbus Window is translated into a VMEbus access once VMEbus access is enabled.

The complete VMIVME-7489 memory map is presented in Chapter 3, *PC/AT Functions* on page 43, but for reference the address ranges to be excluded for the VMEbus Windows are listed in Table 2-2.

Table 2-2 VMEbus Window Addresses

VMEbus WINDOW	ADDRESS RANGE TO EXCLUDE
Real Mode VMEbus Window	$E0000 - $EFFFF
Protected Mode VMEbus Window	$0200 0000 - $FFFD FFFF

Note that although the Protected Mode VMEbus Window is only 16 Mbytes in size, all addresses within the Protected Mode VMEbus Window *and above* should be excluded up to $FFFD FFFF. This is because the addresses above the Protected Mode VMEbus Window up to the beginning of the reserved addresses at $FFFE 0000 are images of the Protected Mode VMEbus Window and are translated into VMEbus accesses just as if they were inside the actual VMEbus Window.

Operating systems vary widely in their implementation of address range exclusion. Indeed, some operating systems without complex memory managers such as plain DOS may not need to have any addresses excluded. Nevertheless, the General Rule should be followed wherever possible to avoid unwanted VMEbus accesses.

Configuration Examples

Configuring MS-DOS for the VMIVME-7489

A plain installation of MS-DOS does not require any modification, since it is strictly a Real Mode operating system and addresses nothing above the lower 640 Kbytes of RAM except for the BIOS.

Especially since the release of MS-DOS Version 5.0, it has become popular to supplement DOS with enhanced memory managers in order to take advantage of extended RAM above the traditional 640 Kbyte limit. Since the VMEbus Windows exist in this space above 640 Kbytes, address range exclusion becomes necessary when using such memory managers.

The most common memory manager used with MS-DOS is the one included with DOS itself, EMM386.EXE. If EMM386.EXE is loaded in your CONFIG.SYS configuration file, edit the line that loads EMM386.EXE and add the parameter `X=E000-EFFF` to the line to exclude the Real Mode VMEbus Window. The result may look similar to this:

```
DEVICE=C:\DOS\EMM386.EXE X=E000-EFFF NOEMS
```

If EMS expanded memory support is desired, it is also useful to specify the 64 Kbyte page frame location, which should be at $D0000 on the VMIVME-7489. Specify the page frame for EMM386.EXE by adding the parameter `FRAME=D000`. The resulting EMM386.EXE line might then look something like this:

```
DEVICE=C:\DOS\EMM386.EXE 1024 FRAME=D000 X=E000-EFFF RAM
```

Notice two things about configuring EMM386.EXE: first, it expects only segment addresses (for example, D000 instead of the full D0000) and second, no exclusion is necessary for the Protected Mode VMEbus Window. The latter is true because EMM386.EXE will not attempt to use memory above that reported by the BIOS, which will always be below the beginning of the Protected Mode VMEbus Window.

The modifications above also work with any operating system that uses EMM386.EXE, including MS-DOS and PC-DOS. Other memory managers, including QEMM, 386Max, and Netroom, have similar ways to specify excluded addresses and expanded memory page frames. Consult the appropriate manual for the exact implementation.

Configuring Windows for the VMIVME-7489

Microsoft Windows will run well on the VMIVME-7489, but the Real Mode VMEbus Window needs to be excluded from its memory manager. Edit the SYSTEM.INI file in the main Windows directory and add the following lines to the [386Enh] section:

EMMExclude=E000-EFFF

ReservedHighArea=E000-EFFF

The Protected Mode VMEbus Window need not be excluded, since Windows will not attempt to use more memory than that reported by the BIOS. Of course, Windows versions up to 3.11 require that DOS be loaded first, so it is important to configure DOS as described previously.

CHAPTER

3

PC/AT Functions

Contents

CPU Register Model 44
Physical Memory 45
Memory and Port Maps 46
PC / AT Interrupts 51
Enhanced I / O Ports 56
Video Graphics Adapter 58
PC / 104 Expansion Site 59

CPU Register Model

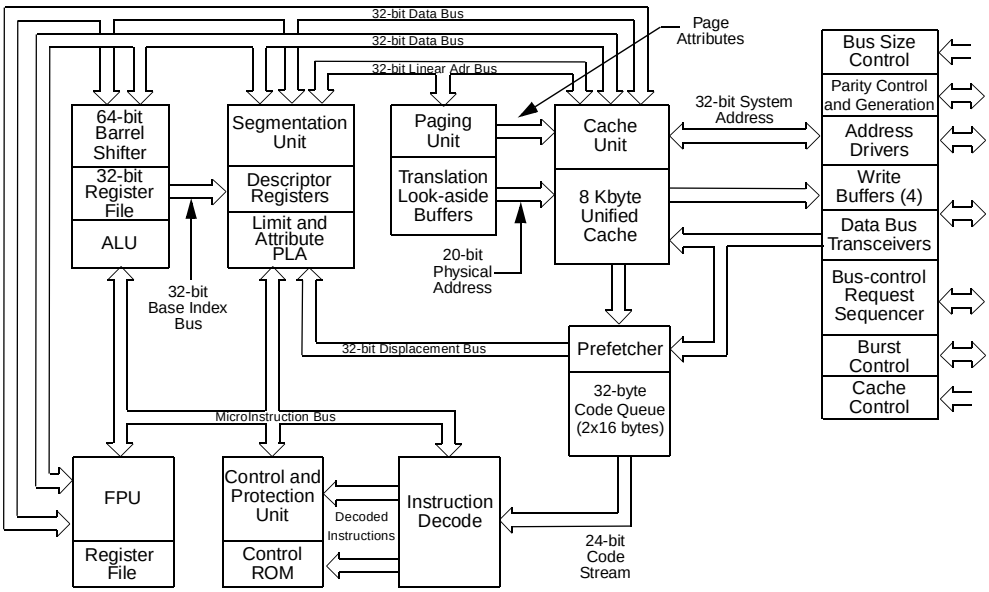


Figure 3-1 80486 CPU Block Diagram

Physical Memory

The VMIVME-7489 has one 72-pin SIMM socket which supports a 1 M x 36, 2 M x 36, 4 M x 36, or 8 M x 36 factory-installed SIMM module for a total of 4, 8, 16, or 32 Mbytes of DRAM. The memory module is rated at 70 ns or faster. Memory can be accessed as bytes, words, or longwords. Note that this memory is not user-upgradable and is not dual-ported to the VMEbus. For applications requiring dual-ported access, ask about the VMIVME-7487 board from VMIC.

The on-board DRAM above the 1 Mbyte boundary can be configured as extended memory (default), managed extended memory (XMS) with a software driver such as HIMEM.SYS, or as LIM EMS expanded memory with the proper driver (like Microsoft's EMM386 or QEMM from Quarterdeck Office Systems). If expanded memory is used, the recommended 64 Kbyte page frame is at $D0000. See Chapter 2, *Installation and Setup* on page 29 for a detailed discussion concerning operating system and memory manager configuration.

The VMIVME-7489 includes both the system and video BIOS in a single 128 K x 8 EPROM. The system portion of this ROM (at $F0000) is automatically shadowed, but for higher performance it is recommended that shadow RAM for the video BIOS be enabled (see the *BIOS Setup User's Manual*)

The VMIVME-7489 may also contain some external cache memory. If present, the cache consists of either 256 Kbytes or 1 Mbyte of direct-mapped level two (L2) fast cache RAM. External cache enhances system performance by reducing memory access time, since cache access is approximately five to ten times faster than DRAM access. Either size external cache uses a high speed write-back design and may be disabled from the BIOS Setup program (see Chapter 2, *Installation and Setup* on page 29).



Note Accesses in the Real Mode VMEbus Window and Protected Mode VMEbus Window address ranges are never cached. This ensures the integrity of any data read from the VMEbus.

Memory and Port Maps

Memory Map

The memory map for the VMIVME-7489 is shown in Table 3-1. All systems share this same memory map, although a VMIVME-7489 with less than the full 32 Mbytes of DRAM does not fill the entire space reserved for On-Board Extended Memory.

Table 3-1 VMIVME-7489 Memory Address Map

MODE	MEMORY ADDRESS RANGE	SIZE	DESCRIPTION
PROTECTED MODE	$FFFF 0000 - $FFFF FFFF	64 Kbytes	ROM BIOS Image
	$FFFE 0000 - $FFFE FFFF	64 Kbytes	Reserved
	$0210 0000 - $FFFD FFFF	4047+ Mbytes	Reserved (Images of the Protected Mode VMEbus Window)
	$0200 0000 - $020F FFFF	16 Mbytes	Protected Mode VMEbus Window
	$0010 0000 - $01FF FFFF	31 Mbytes	Reserved for On-Board Extended Memory (not filled on all systems)
REAL MODE	$F0000 - $FFFFF	64 Kbytes	ROM BIOS
	$E0000 - $EFFFF	64 Kbytes	Real Mode VMEbus Window
	$D0000 - $DFFFF	64 Kbytes	Reserved for Expansion BIOS (alternately may be used as a LIM/EMS page frame)
	$C8000 - $CFFFF	32 Kbytes	Reserved for Expansion BIOS
	$C0000 - $C7FFF	32 Kbytes	Video ROM
	$A0000 - $BFFFF	128 Kbytes	Video RAM
	$00000 - $9FFFF	640 Kbytes	User RAM/DOS RAM

I/O Port Map

The 80486 CPU has special input/output instructions that access I/O peripherals residing in I/O addressing space (which is separate and distinct from memory addressing space). When the CPU decodes and executes an I/O instruction, it produces a 16-bit I/O address on lines A00-A15 and identifies the I/O cycle with the M/IO control line. Thus, the CPU has an independent 64 Kbyte I/O address space which is accessible as bytes, words, or longwords. Locations in I/O address space are often referred to as *ports*.

Standard PC/AT hardware circuitry decodes only 10 of the I/O address lines going out to the expansion bus. This limits the PC/104 address space to 1024 locations ($000-$3FF).

The VMIVME-7489 incorporates all standard I/O peripherals of the PC/AT architecture such as keyboard, DMA, interrupt controllers, timers and real-time clock, as well as parallel and serial I/O ports, video registers, and floppy and hard drive task registers. The BIOS initializes and configures all these registers properly, so beware of adjusting these I/O ports directly. The assigned and user-available I/O addresses are summarized in the I/O Address Map, Table 3-2. Some I/O hardware can be addressed at several different locations determined by a hardware jumper, notably the parallel ports and drive controllers. See Chapter 2, *Installation and Setup* on page 29 for details concerning hardware jumpers.

Table 3-2 VMIVME-7489 I/O Address Map

I/O ADDRESS RANGE	SIZE IN BYTES	HARDWARE DEVICE	PC/AT FUNCTION	ASSOCIATED JUMPERS†
$000 - $00F	16	ACC Micro 2168 Chip	DMA Controller 1 (Intel 8237A Compatible)	
$010 - $01F	16		Reserved	
$020 - $021	2	ACC Micro 2168 Chip	Master Interrupt Controller (Intel 8259A Compatible)	
$022 - $03F	30		Reserved	
$040 - $043	4	ACC Micro 2168 Chip	Programmable Timer (Intel 8254 Compatible)	
$044 - $05F	30		Reserved	
$060 - $064	5	ACC Micro 2168 Chip	Keyboard, Speaker, Eqpt. Config (Intel 8042 Compatible)	
$065 - $06F	11		Reserved	
$070 - $071	2	ACC Micro 2168 Chip	Real-Time Clock, NMI Mask	
$072 - $07F	14		Reserved	
$080 - $08F	16	ACC Micro 2168 Chip	DMA Page Registers	
$090 - $091	2		Reserved	
$092	1	ACC Micro 2168 Chip	Alt. Gate A20 / Fast Reset Register	
$093	1		Reserved	
$094	1	Super VGA Chip	POS102 Access Control Register	E5 & E6
$095 - $09F	11		Reserved	
$0A0 - $0A1	2	ACC Micro 2168 Chip	Slave Interrupt Controller (Intel 8259A Compatible)	

Table 3-2 VMIVME-7489 I/O Address Map (Continued)

I/O ADDRESS RANGE	SIZE IN BYTES	HARDWARE DEVICE	PC/AT FUNCTION	ASSOCIATED JUMPERS†
$0A2 - $0BF	30		Reserved	
$0C0 - $0DF	32	ACC Micro 2168 Chip	DMA Controller 2 (Intel 8237A Compatible)	
$0E0 - $0EF	16		Reserved	
$0F0 - $0FF	16	ACC Micro 2168 Chip	Math Coprocessor / 2168 Config	
$100 - $101	2		Reserved	
$102	1	Super VGA Chip	POS102 Register	E5 & E6
$103 - $13F	61		Reserved	
$140 - $14F	16	**Custom VMIC Hardware**	**VMEbus Interface Registers (see Chapter 4,** *VMEbus Functions* **on page 61 for details)**	E11, E12, and E14
$150 - $16F	32		Reserved	
$170 - $177	8	Super I/O Chip	Secondary Hard Disk Controller	E3 & E7
$178 - $1EF	120		**User I/O**	
$1F0 - $1F7	8	Super I/O Chip	Primary Hard Disk Controller	E3 & E7 (default)
$1F8 - $277	128		**User I/O**	
$278 - $27F	8	Super I/O Chip	LPT2 Enhanced Parallel I/O (EPP/ECP Compatible)	E4 & E8
$280 - $2E7	104		Reserved by VMIC	
$2E8 - $2EE	7	UART*	COM4 Serial I/O*	
$2EF - $2F7	9		**User I/O**	
$2F8 - $2FE	7	Super I/O Chip	COM2 High-Speed Serial I/O (16550A Compatible)	
$2FF - $31F	33		Reserved by VMIC	
$320 - $36F	80		**User I/O**	
$370 - $377	8	Super I/O Chip	Secondary Floppy Disk Controller	E3 & E7
$378 - $37F	8	Super I/O Chip	LPT1 Enhanced Parallel I/O (EPP/ECP Compatible)	E4 & E8 (default)
$380 - $3AF	48		Reserved	

Table 3-2 VMIVME-7489 I/O Address Map (Continued)

I/O ADDRESS RANGE	SIZE IN BYTES	HARDWARE DEVICE	PC/AT FUNCTION	ASSOCIATED JUMPERS†
$3B0 - $3BB	16	Monochrome Adapter* or Super VGA Chip	Monochrome Adapter* or VGA Monochrome Video	E5 & E6‡
$3BC - $3BF	4	Super I/O Chip	Alternate LPT1 Enhanced Parallel I/O (ECP Compatible)	E4 & E8
$3C0 - $3CF	16	Super VGA Chip	VGA Adapter	E5 & E6
$3D0 - $3DF	16	CGA Adapter* or Super VGA Chip	CGA Adapter* or VGA Color Video	E5 & E6‡
$3E0 - $3E7	8		Reserved	
$3E8 - $3EE	7	UART*	COM3 Serial I/O*	
$3F0 - $3F7	8	Super I/O Chip	Primary Floppy Disk Controller	E3 & E7 (default)
$3F8 - $3FE	7	Super I/O Chip	COM1 High-Speed Serial I/O (16550A Compatible)	
$3FF	1		Reserved	
ADDRESSES BEYOND $3FF ARE NOT USABLE BY PC/104 EXPANSION BOARDS				
$400 - $677	632		Reserved	
$678 - $67A	3	Super I/O Chip	LPT2 ECP Registers	E4 & E8
$67B - $777	253		Reserved	
$778 - $77A	3	Super I/O Chip	LPT1 ECP Registers	E4 & E8 (default)
$77B - $7BB	65		Reserved	
$7BC - $7BE	3	Super I/O Chip	Alternate LPT1 ECP Registers	E4 & E8
$7BF - $46E7	16169		Reserved	
$46E8	1	Super VGA Chip	VGA Adapter Sleep Register	E5 & E6
$46E9 - $FFFF	47383		Reserved	

* While these I/O ports are reserved for the listed functions, they are not implemented on the VMIVME-7489. They are listed here to make the user aware of the standard PC/AT usage of these ports.

† These jumpers can affect whether or not the listed hardware is addressed at the given location. Hardware jumper settings are discussed in Chapter 2, *Installation and Setup* on page 29.

‡ The BIOS initializes the VGA registers to either the Monochrome or Color I/O space depending on the Primary Display setting in Standard CMOS Setup (see Chapter 2, *Installation and Setup* on page 29 for details concerning BIOS Setup).

PC/AT Interrupts

In addition to an I/O port address, an I/O device often has a separate hardware interrupt line assignment. Assigned to each interrupt line is a corresponding interrupt vector in the 256-vector interrupt table at $00000 to $003FF in memory. The 16 maskable interrupts and the single nonmaskable interrupt (NMI) are listed in Table 3-3 along with their functions. Table 3-4 on page 52 details the vectors in the interrupt vector table.

Table 3-3 PC/AT Hardware Interrupts

IRQ	AT FUNCTION	COMMENTS
NMI	Parity Errors (Must be enabled in BIOS Setup)	Used by VMIVME-7489 VMEbus Interface
0	System Clock/Calendar	Set by BIOS Setup
1	System Clock/Calendar	Set by BIOS Setup
2	Duplexed to IRQ9	
3	COM2 / COM4	
4	COM1 / COM3	
5	LPT2	**Used by VMIVME-7489 VMEbus Interface** (alternate choice, set by jumper E12)
6	Floppy Controller	
7	LPT1	
8	Real-Time Clock	
9	Old IRQ2	VGA or Network I/O
10	Not Assigned	
11	Not Assigned	**Used by** VMIVME-7489 **VMEbus Interface** (primary choice, set by jumper E12)
12	Not Assigned	**Used by** VMIVME-7489 **VMEbus Interface** (alternate choice, set by jumper E12)
13	Math Coprocessor	
14	AT Hard Drive	
15	Not Assigned	

Table 3-4 PC/AT Interrupt Vector Table

INTERRUPT #		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
00	0		Divide Error	Same as Real Mode
01	1		Debug Single Step	Same as Real Mode
02	2	NMI	Memory Parity Error, **VMEbus Interrupts**	Same as Real Mode (Must be enabled in BIOS Setup)
03	3		Debug Breakpoint	Same as Real Mode
04	4		ALU Overflow	Same as Real Mode
05	5		Print Screen	Array Bounds Check
06	6			Invalid OpCode
07	7			Device Not Available
08	8	IRQ0	Timer Tick	Double Exception Detected
09	9	IRQ1	Keyboard Input	Coprocessor Segment Overrun
0A	10	IRQ2	BIOS Reserved	Invalid Task State Segment
0B	11	IRQ3	COM2 Serial I/O	Segment Not Present
0C	12	IRQ4	COM1 Serial I/O	Stack Segment Overrun
0D	13	IRQ5	Fixed Disk Controller or VMEbus Interrupts (alt)	General Protection Violation
0E	14	IRQ6	Floppy Disk Controller	Page Fault
0F	15	IRQ7	LPT1 Parallel I/O	Unassigned
10	16		BIOS Video I/O	Coprocessor Error
11	17		Eqpt Configuration Check	Same as Real Mode
12	18		Memory Size Check	Same as Real Mode
13	19		XT Floppy/Hard Drive	Same as Real Mode
14	20		BIOS Comm I/O	Same as Real Mode
15	21		BIOS Cassette Tape I/O	Same as Real Mode
16	22		BIOS Keyboard I/O	Same as Real Mode
17	23		BIOS Printer I/O	Same as Real Mode
18	24		ROM BASIC Entry Point	Same as Real Mode
19	25		Bootstrap Loader	Same as Real Mode
1A	26	IRQ8	Real-Time Clock	Same as Real Mode

Table 3-4 PC/AT Interrupt Vector Table (Continued)

INTERRUPT #		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
1B	27		Control/Break Handler	Same as Real Mode
1C	28		Timer Control	Same as Real Mode
1D	29		Video Parameter Table Pntr	Same as Real Mode
1E	30		Floppy Parm Table Pntr	Same as Real Mode
1F	31		Video Graphics Table Pntr	Same as Real Mode
20	32		DOS Terminate Program	Same as Real Mode
21	33		DOS Function Entry Point	Same as Real Mode
22	34		DOS Terminate Handler	Same as Real Mode
23	35		DOS Control/Break Handler	Same as Real Mode
24	36		DOS Critical Error Handler	Same as Real Mode
25	37		DOS Absolute Disk Read	Same as Real Mode
26	38		DOS Absolute Disk Write	Same as Real Mode
27	39		DOS Program Terminate, Stay Resident	Same as Real Mode
28	40		DOS Keyboard Idle Loop	Same as Real Mode
29	41		DOS CON Dev. Raw Output	Same as Real Mode
2A	42		DOS 3.x+ Network Comm	Same as Real Mode
2B	43		DOS Internal Use	Same as Real Mode
2C	44		DOS Internal Use	Same as Real Mode
2D	45		DOS Internal Use	Same as Real Mode
2E	46		DOS Internal Use	Same as Real Mode
2F	47		DOS Print Spooler Driver	Same as Real Mode
30-60	48-96		Reserved by DOS	Same as Real Mode
61-66	97-102		User Available	Same as Real Mode
68-71	103-113		Reserved by DOS	Same as Real Mode
72	114	IRQ10		Same as Real Mode

Table 3-4 PC/AT Interrupt Vector Table (Continued)

INTERRUPT #		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
73	115	IRQ11	VMEbus Interrupts (primary)	Same as Real Mode
74	116	IRQ12	VMEbus Interrupts (alternate)	Same as Real Mode
75-7F	117-127		Reserved by DOS	Same as Real Mode
80-F0	128-240		Reserved for BASIC	Same as Real Mode
F1-FF	241-255		Reserved by DOS	Same as Real Mode

The maskable interrupts are prioritized in hardware by the equivalent of two cascaded Intel 8259A Priority Interrupt Controller (PIC) chips. At boot-up time, the BIOS writes to each PIC an 8-bit vector that maps each Interrupt Request line (IRQx) to its corresponding interrupt vector in memory. Also at boot-up time, the correct 32-bit interrupt vector must be loaded at the right place in the interrupt table. Later, when the IRQx line is acknowledged, the CPU reads the 8-bit vector returned by the PIC, multiplies it by 4 to create an index into the interrupt table, then retrieves the 32-bit (segment:offset) pointer from the table. The CPU uses that pointer to branch to the interrupt service routine corresponding to the IRQx line.

The interrupt hardware implementation on the VMIVME-7489 is standard for computers built around the PC / AT architecture, which evolved from the IBM PC / XT. In the IBM PC / XT computers only eight interrupt request lines exist, numbered from IRQ0 to IRQ7 at the PIC. The IBM PC / AT computer added eight more IRQx lines, numbered IRQ8 to IRQ15, by cascading a second slave PIC into the original master PIC. IRQ2 at the master PIC was committed as the cascade input from the slave PIC. IRQ13 at the slave is reserved for the math coprocessor.

To maintain backward compatibility with PC / XT systems, IBM chose to use the new IRQ9 input on the slave PIC to operate as the old IRQ2 interrupt line on the PC / XT Expansion Bus. Thus, in AT systems, the IRQ9 interrupt line connects to the old IRQ2 pin (pin B4) on the AT Expansion Bus (or ISA bus). This is the same implementation used on the VMIVME-7489's PC / 104 expansion port.

Figure 3-2 depicts the VMIVME-7489 interrupt logic pertaining to VMEbus operations. Note that jumper E12 selects either IRQ5, IRQ11, or IRQ12 for receiving VMEbus related interrupts. Also, note that the NMI interrupt generated by the VMIVME-7489 must be enabled in the BIOS Setup (see Chapter 2, *Installation and Setup* on page 29).

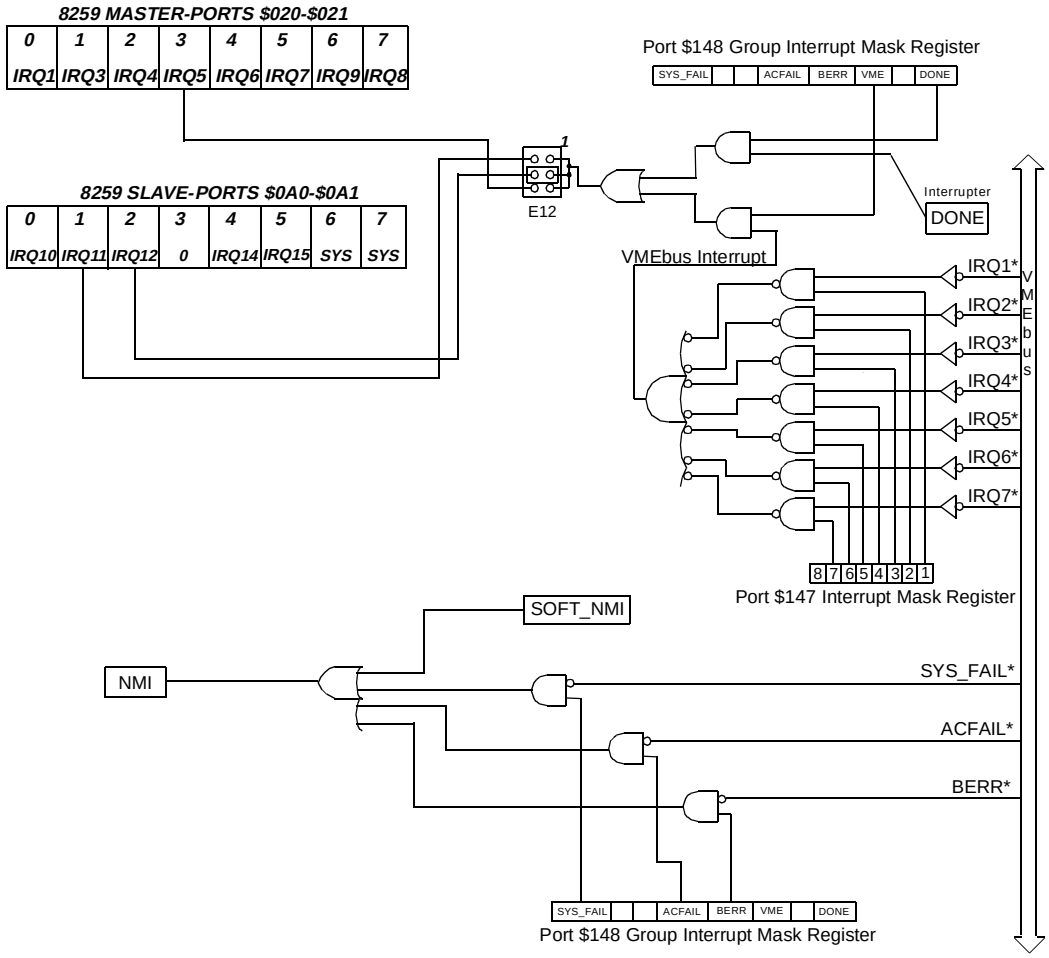


Figure 3-2 Interrupt Logic Diagram

Enhanced I/O Ports

The VMIVME-7489 incorporates the Intel 82091AA Advanced Integrated Peripheral chip, also known as the AIP or Super I/O chip. This chip provides the VMIVME-7489 with a floppy drive controller, IDE hard drive interface, two serial ports, and one parallel port. While the floppy and hard drive interfaces are standard, the AIP has two high-performance advantages over ordinary ISA bus peripherals: high-speed 16550 compatible serial ports, and a fast multifunctional bidirectional parallel port.

Serial Ports

Each of the two serial ports uses a 16550 compatible UART, including 16 byte transmit and receive FIFO buffers. The FIFOs reduce processor overhead usually needed when the ports are active, allowing reliable serial transfer speeds up to 38.4 kilobaud.

Note that in order to maintain compatibility with older, non-FIFO-aware communications programs, the VMIVME-7489's BIOS does not enable the serial port FIFO buffers. Most contemporary communications programs and drivers automatically detect a 16550 compatible port and enable the buffers themselves.

The Microsoft Windows 3.1 communications port driver recognizes 16550 compatible UARTs, but only enables the receive FIFO with a fixed interrupt depth of 10. By not taking advantage of the transmit FIFO, the Windows driver incurs more processor overhead and may not be reliable beyond 9600 baud, especially while multitasking. Fortunately, many third party vendors have custom Windows communications port drivers that address this problem.

Parallel Port

The VMIVME-7489's enhanced parallel port may be configured for any one of four modes as defined by IEEE Standard 1284 shown in Table 3-5.

Table 3-5 Parallel Port Modes

PARALLEL PORT MODE	PARALLEL INTERFACE PROTOCOL
ISA-Compatible Mode	Compatibility, Nibble
PS/2-Compatible Mode	Byte
EPP Mode*	EPP
ECP Mode	ECP

* EPP Mode is unavailable if the parallel port is jumpered for the alternate LPT1 I/O address at $3BC-$3BF. See Chapter 2 for hardware jumper details.

In the ISA compatible and PS/2 compatible modes, software controls the parallel port handshake signals during data transfers. The EPP protocol increases throughput by generating an automatic handshake in hardware. The ECP protocol not only implements a hardware handshake, but adds a 16 byte FIFO with DMA capability to reduce processor overhead.

The VMIVME-7489's AIP chip supports all parallel port modes in Table 3-5 on page 56, except the EPP mode is not supported if the alternate LPT1 I/O address of $3BC-$3BF is used (See Chapter 2, *Installation and Setup* on page 29 for hardware setup details). The BIOS initializes the port to ISA-Compatible mode in order to maximize compatibility with older applications, but newer software will recognize the advanced port and reconfigure it accordingly.

Video Graphics Adapter

The monitor port on the VMIVME-7489 is controlled by a Cirrus Logic chip with 1 Mbyte video DRAM. The video controller chip is hardware and BIOS compatible with the IBM EGA and VGA standards and also supports VESA high-resolution and extended video modes. Table 3-6 shows the popular graphics video modes supported by the Cirrus Logic video chip.

Table 3-6 Common Supported Graphics Video Resolutions

SCREEN RESOLUTION	MAXIMUM COLORS	REFRESH RATES
640 x 480	up to 16.5 million (TruColor)	60 Hz, 72 Hz
800 x 600	up to 65536 (HiColor)	56 Hz, 60 Hz, 72 Hz
1024 x 768 (Interlaced)	up to 256	43.5 Hz (Interlaced to 87 Hz)
1024 x 768	up to 256	60 Hz, 70 Hz, 72 Hz
1280 x 1024 (Interlaced)	up to 16	43.5 Hz (Interlaced to 87 Hz)

Note that not all VGA monitors support resolutions and refresh rates beyond 640 x 480 at 60 Hz. Do not attempt to drive a monitor to a resolution or refresh rate beyond its capabilities.

The VMIVME-7489's processor has 16-bit access to video memory with no wait states. Video I/O registers are accessed at the maximum ISA bus rate. In addition, the Cirrus Logic video controller supports up to a 64 x 64 pixel hardware cursor.

Floppy disks supplied with the VMIVME-7489 contain video drivers for Windows and other popular programs and operating systems. Follow instructions on the floppy disk label to install the drivers using an interactive installation program.

PC/104 Expansion Site

The VMIVME-7489 supports industry-standard expansion modules through PC/104: the IEEE P996.1 Standard for Compact Embedded-PC Modules. PC/104 modules are available from third party vendors for display devices, PCMCIA adapters, Ethernet and SCSI interfaces, and other I/O functions.

VMIC's VMIVME-7432 PC/104 to ISA Adapter is designed specifically to mate with the VMIVME-7489, allowing it to use a standard half-length ISA bus board in the same chassis. This expansion allows low-cost or specialty ISA bus boards to be used in the same chassis with VMEbus equipment.

In order to allow the VMIVME-7450 hard and floppy drive module to be attached along with the VMIVME-7432, the VMIVME-7432 duplicates the hard drive and floppy drive controller functions of the VMIVME-7489. The on-board VMIVME-7489 hard and floppy controller should therefore be disabled when a VMIVME-7432 is installed.

See Chapter 2, *Installation and Setup* on page 29 for hardware configuration details, a pinout table for the PC/104 connectors, and tips on installing PC/104 expansion boards.

CHAPTER

4

VMEbus Functions

Contents

VMEbus Access 62
Byte Ordering 64
Interrupt Handling 69
Interrupter 74
Requester 75
System Controller 76
System Registers 77

VMEbus Access

This section discusses how the VMIVME-7489 interfaces with the VMEbus. At power up or after a system reset (but not after a soft reset, or <CTRL+ALT+DEL> reset), the VMIVME-7489 is "isolated" from the VMEbus. The user must first set the VME Enable bit in the VMEbus Access Control Register before access to the VMEbus is possible so that the VMIVME-7489 board does not interfere with other VMEbus modules or inadvertently write to VMEbus memory during initialization and POST.

Once the operating system is loaded into the VMIVME-7489, the user may proceed with VMEbus activities from a known state and in an orderly manner.

At power up, the 80486 processor enters the Real Mode. MS-DOS is typically loaded and run in this mode. Only 1 Mbyte of memory is available in Real Mode. Windows, OS/2, and XENIX, on the other hand, run in Protected Mode and can utilize all available extended memory.

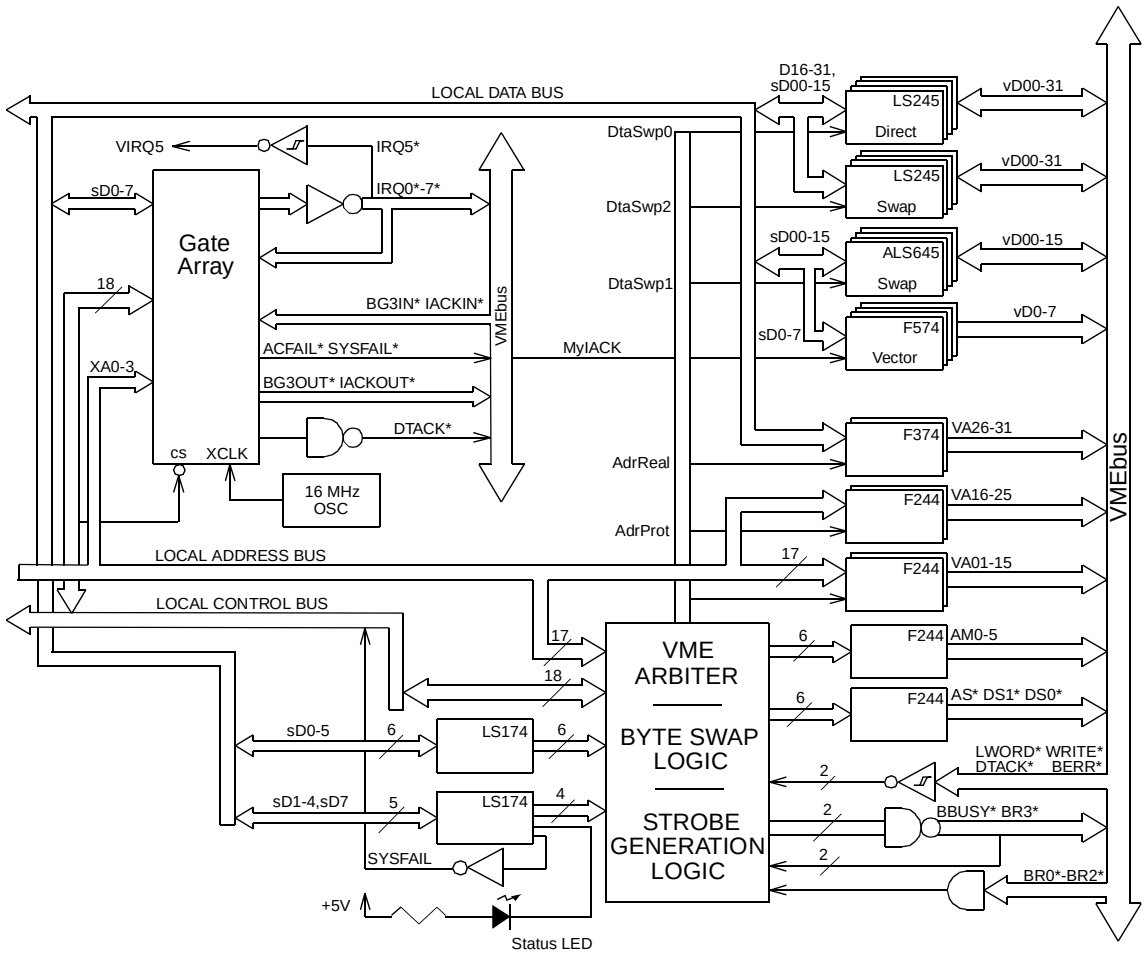


Figure 4-1 VMIVME-7489 VMEbus Interface Block Diagram

The VMIVME-7489 can access the VMEbus in Real Mode or Protected Mode. As a VMEbus interrupt processor the VMIVME-7489 can also perform IACK* cycles. Figure 4-1 shows the functional interconnections between the VMIVME-7489 and the VMEbus.

In Real Mode, the VMIVME-7489 uses a paging scheme to access any part of the 4 Gbytes of VMEbus addressing space, 64 Kbytes at a time, through its Real Mode VMEbus Window at $E0000. The user can set the VMEbus Address Modifier lines (AM5-AM0) as required to define the Data Transfer Cycle Address Mode as Short (16-bit / A16), Standard (24-bit / A24), Extended (32-bit / A32), or User Defined. VMEbus address bits beyond the 16 available within the Real Mode VMEbus Window may be controlled by the VMEbus Standard Access Register (for bits A16-A23) and the VMEbus Extended / AM Access Register (for bits A24-A31 as well as the Address Modifier bits).

In Protected Mode, the VMIVME-7489 can access the VMEbus through the Protected Mode VMEbus Window, which occupies 16 Mbytes beginning at $0200 0000. By supplying the extended VMEbus address bits A24-A31 in the VMEbus Extended / AM Access Register, all 4 Gbytes of VMEbus memory may be addressed in Protected Mode, 16 Mbytes at a time.

Note that due to the way the addresses are decoded, addressing space above the Protected Mode VMEbus Window but below the highest 128 Kbytes will also generate VMEbus addresses. Accesses in this region should therefore be avoided to prevent inadvertent VMEbus activity.



Note Unaligned transfers to or from the VMEbus are not supported. Transfers must be aligned on natural BYTE, WORD, or LONGWORD boundaries.

Byte Ordering

Byte Swapping

For a given addressing mode, the VMEbus Specification defines various types of data transfer cycles to access 1, 2, 3, or 4 byte locations at once. A set of four adjacent byte locations differing only in address bits (A00, A01) is defined as a four-byte group, or a "byte (0-3)" group. Address lines A02-A31 select a four-byte group, then four additional addressing lines (DS0*, DS1*, A01, and LWORD*) select which byte(s) within the group are accessed.

Table 4-1 depicts the Even / Odd Byte assignments to the VMEbus data lines.

Table 4-1 VMEbus Byte Assignment to the Data Lines

DTB CYCLE TYPE	D31-D24	D23-D16	D15–D08 EVEN ADDRESS	D07–D00 ODD ADDRESS
D08(EO) EVEN OR ODD				
Single Odd Byte(3)				Byte(3)
Single Even Byte(2)			Byte(2)	
Single Odd Byte(1)				Byte(1)
Single Even Byte(0)			Byte(0)	
D08(O) ODD ONLY				
Single Odd Byte(3)				Byte(3)
Single Odd Byte(1)				Byte(1)
D16				
Double Byte(2-3)			Byte(2)	Byte(3)
Double Byte(0-1)			Byte(0)	Byte(1)
D32				
Quad Byte(0-3)	Byte(0)	Byte(1)	Byte(2)	Byte(3)

It is important to note the major byte ordering differences between the VMEbus and the Intel 80486 CPU. In addition, communication between Motorola compatible 680X0 VMEbus modules and the VMIVME-7489 requires special attention to avoid byte ordering conflicts.

The Byte Swapping Problem Defined

The byte-ordering issue exists due to the different traditions at the major microprocessor manufacturers, Motorola and Intel. VMEbus equipment is designed around Motorola's 680X0 processors and compatibles, which store multiple-byte values in memory with the *most* significant byte at the lowest byte address. This byte ordering scheme became known as *big-endian* ordering. On the other hand, Intel's 80X86 microprocessors, upon which MS-DOS is based, store multiple-byte values in memory with the *least* significant byte in the lowest byte address, thus earning the name *little-endian* ordering.

The VMIVME-7489 uses an Intel 80486 microprocessor, which uses little-endian byte ordering. Byte arrangement and the byte relationship between data in the processor and transferred data in memory are shown in Figure 4-2.

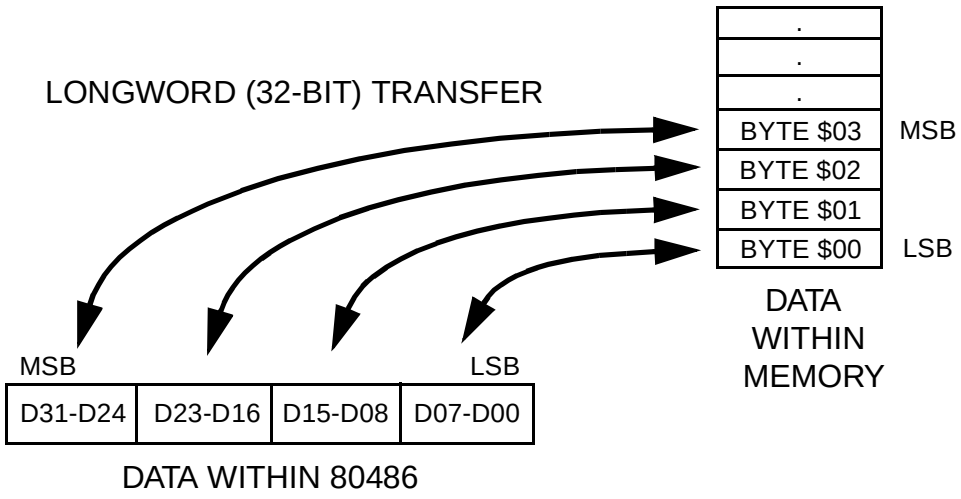


Figure 4-2 Byte Relationships Using the Little-Endian 80486

Note that in little-endian processors like the 80486, the processor's least significant byte is stored in the lowest byte address after a multiple-byte write (such as the longword transfer illustrated), while the processor's most significant byte is stored in the highest byte address after such transfers. Conversely, the processor considers data retrieved from the lowest byte address to be the least significant byte after a multiple-byte read. Data retrieved from the highest byte address is considered to be the most significant byte.

Contrast the behavior of the little-endian 80486 in Figure 4-2 with the same longword transfer using a big-endian processor like the Motorola 68040 in Figure 4-3 on page 66.

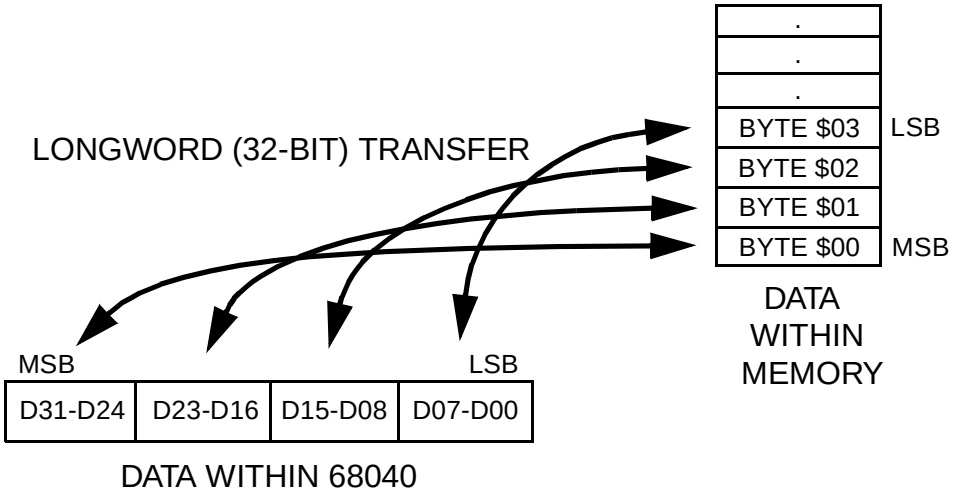


Figure 4-3 Byte Relationships Using the Big-Endian 68040

Note that the big-endian 68040 handles the same longword transfer in a completely different manner than the little-endian 80486. During a multiple-byte transfer like the longword transfer illustrated, a big-endian processor writes its least significant byte in the highest byte address in memory, while its most significant byte is written to the lowest address. The converse is true during read operations: the data in the lowest byte address is considered to be the most significant, while the byte in the highest address is considered to be the least significant.

Byte Swapping and the VMEbus

The VMEbus Specification does not specify which byte of a multiple-byte transfer is most significant. The VMEbus Specification does, however, require certain byte lanes to be associated with certain byte addresses. As shown in Table 4-1 on page 64, byte(0) must be transferred on data lines D31-D24 during a longword transfer while byte(3) must be transferred on lines D7-D0. This byte and address alignment is exactly the same as that for a big-endian processor such as the Motorola 68040 in Figure 4-3.

If a little-endian 80486 were to have its data bus directly connected to the VMEbus (i.e. D31 to D31, D30 to D30, etc.), then the most significant byte data supplied to the VMEbus D31-D24 byte lane during a longword write would be stored by the VMEbus in the lowest of the four destination byte addresses – opposite that expected by the 80486 (see Figure 4-2 on page 4-65). This poses no problem if the 32-bit value written is always read back using a similar longword transfer (i.e. all four bytes at once), since the swapped data gets swapped again and appears to the 80486 exactly as it should. However, if the data written by the 32-bit longword transfer were to be retrieved using any other method – say, using four separate byte transfers – a problem results, since the data at the lowest byte address would be incorrectly assumed to be the least significant, while it is actually the most significant.

The problem cannot be solved by simply connecting the 80486 to the VMEbus with its byte lanes crossed. For example, the 80486 uses D0-D7 to transfer a byte to address $00, while the VMEbus requires D8-D15 be used. For this reason, special hardware has been incorporated into the VMIVME-7489 to facilitate different kinds of byte swapping for varying circumstances.

VMIVME-7489 Byte Swapping Hardware

The VMIVME-7489 employs three-way byte swap buffers to accommodate the byte ordering inconsistencies between the 80486 and the VMEbus, but the programmer must still be aware of the problem and decide when to use the built-in byte swapping function. Figure 4-4 on page 67 diagrams the function of these swap buffers.

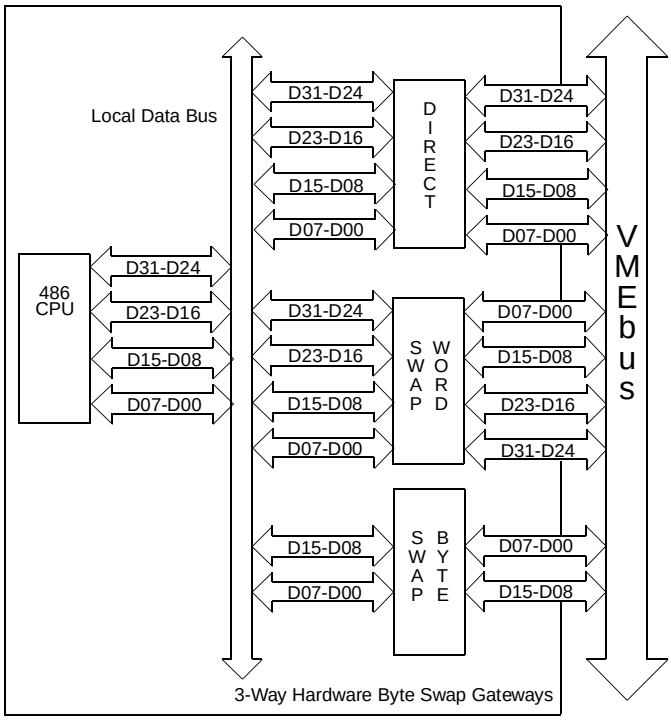


Figure 4-4 80486-to-VMEbus Data Byte Lanes

The byte-swapping hardware is affected by two factors: the Big Endian bit in the VMEbus Access Control Register, and the size of the transfer being carried out. Table 4-2 details this relationship.

Table 4-2 Byte Swap Modes

BIG ENDIAN BIT STATUS	SIZE OF TRANSFER	SWAP MODE
X	single byte	byte-swap
0	word (two bytes)	byte-swap
0	longword (four bytes)	word-swap
1	word (two bytes)	direct
1	longword (four bytes)	direct

Generally speaking, a set Big Endian bit causes the 80486 CPU to access the VMEbus much the same way a big-endian processor would. Two other facts should also be noted from Table 4-2: first, no swapping is ever performed on single-byte transfers, regardless of the state of the Big Endian bit, and second, some form of swapping is *always* used for multiple-byte transfers.

The byte-swapping hardware on the VMIVME-7489 is not a complete solution to the byte ordering problem in itself, since the programmer must still decide when to use each of the available swap schemes. Once decided and configured, however, the hardware does relieve the programmer from tediously swapping the bytes in software.

With this in mind, the following steps should be taken to access the VMEbus in Real Mode through the Real Mode VMEbus Window:

1. After RESET, set the VME Enable and set or clear the Big Endian bit in the VMEbus Access Control Register.
2. Write to the VMEbus Extended / AM Register to select the type of VMEbus cycle and SA31-SA24 lines.
3. Write to the Standard Access Register to select SA23-SA16 lines.
4. Read or write the VMEbus through the Real Mode VMEbus Window at $E0000.

Steps 1-3 may not be required if subsequent accesses are within the same 64 Kbyte range of VMEbus addresses.

Interrupt Handling

The VMIVME-7489 has the ability to recognize and respond to VMEbus interrupt signals IRQ1*-IRQ7*. In addition, the 80486 CPU can be interrupted by other sources associated with the VMEbus interface.

There are six possible VMEbus-related interrupts on the VMIVME-7489: SOFT_NMI, ACFAIL, SYSFAIL, BERR, VMEbus, and DONE. The first four of these (SOFT_NMI, ACFAIL, SYSFAIL, and BERR) interrupt the 80486 using the processor's NMI, allowing normal interrupt processing to be interrupted in the event of a serious error. Specifically, the PC/AT IOCHK signal is activated which in turn produces an NMI. The VMEbus and DONE conditions interrupt the CPU using the INT signal via the PC/AT IRQ 5, IRQ 11, or IRQ 12 signal (jumper selectable – the default is IRQ 11). SOFT_NMI allows the programmer to produce an NMI under software control.

Nonmaskable Interrupts

As stated above, SOFT_NMI, ACFAIL, SYSFAIL, and BERR interrupt the 80486 using the processor's NMI, allowing normal interrupt processing to be interrupted in the event of a serious error. In order for any NMI interrupts to occur at all, however, the Memory Parity Error Check feature must be enabled in the Advanced CMOS Setup menu under BIOS Setup.



Note The Memory Parity Error Check Feature must be enabled in the **Advanced CMOS Setup** screen for any NMI Interrupts to occur.
See the BIOS setup section in Chapter 2, *Installation and Setup* on page 29 for details on changing this setting.

Soft NMI

User software can generate an NMI by writing to the Soft NMI Select Register at I/O $14C. This feature allows testing of NMI-handling code, but can also be useful for performing "pseudo" read-modify-write cycles on global resources without being interrupted, since NMI processing is of the highest priority and cannot be interrupted by any other source.

For example, an attempt to modify the Group Interrupt Mask Register at $148 may be interrupted, potentially causing problems if the outside interrupt occurred between the read and write operations to the register. The solution is to put interrupt-sensitive routines like this inside a Soft NMI handler, guaranteeing that the register update operation cannot be interrupted.

A Soft NMI is generated by writing any value to the Soft NMI Select Register. The user's NMI interrupt service routine may determine whether an NMI was a Soft NMI by reading the Group Interrupt Status Register at I/O $14C and checking the SoftNMI flag therein. If the NMI was caused by a Soft NMI, the flag will be set.

The Soft NMI can be cleared by first setting, then clearing the corresponding SoftNMI bit in the Clear Condition Register.

Note that there is a small amount of delay between the time the user writes to the Soft NMI Select register to cause the NMI and the time the NMI actually occurs at the processor. At most, this delay is the time required for the current CPU instruction plus three CPU clock cycles.

ACFAIL, SYSFAIL, and BERR NMIs

The VMEbus ACFAIL* signal is used to alert the system of an imminent power failure. When ACFail interrupts are enabled (via the Group Interrupt Mask Register), the VMEbus ACFAIL* signal is gated directly to the NMI, not edge-latched. If ACFail condition processing is to continue outside the ISR, then the ACFail bit in the Group Interrupt Mask Register should be cleared before exiting the ISR, since the ACFail interrupt is level sensitive.

The VMEbus SYSFAIL* signal is used to alert the system of an abnormal condition in the VMEbus chassis. When enabled, the VMEbus SYSFAIL* signal is gated directly to the processor's NMI input and is not edge-latched. Like ACFail above, however, the SysFail bit in the Group Interrupt Mask Register must be cleared upon completion of the ISR to enable future SYSFail interrupts. It is recommended that the SysFail interrupt service routine turn on the front panel LED as one of its first operations and turn it off as one of the last operations. Note also that the VMIVME-7489 can cause the SYSFAIL* signal to activate through the VMEbus Access Control Register.

The BERR (Bus Error) NMI alerts the user that the VMEbus BERR* signal was asserted during a VMEbus transfer, although the transfer that caused the bus error was not necessarily initiated by the VMIVME-7489. The BERR NMI is caused by any low-to-high transition of the VMEbus BERR* signal and should be cleared upon completion of the ISR by setting the BERR bit in the Clear Condition Register.

NMI Processing

The processor's NMI vector is at location $008 in memory. Normally, this vector points to the BIOS NMI handler. Since the NMI interrupt on most PC / AT machines is generally used only to detect memory parity errors, the BIOS handler typically responds to NMI interrupts by halting the system and displaying a System Parity Error message on the console.

If the VMIVME-7489 is to handle VMEbus system interrupts using the NMI, however, the default NMI vector needs to be modified to point to the user's own NMI routine. This new ISR should query the Group Interrupt Status Register to determine the cause of the NMI. If the cause is either one of the VMEbus system interrupts described above or the Soft NMI, appropriate response should be taken. The condition should then be cleared and the routine should exit. If the NMI is not caused by a special source, however, the custom ISR should pass control back to the original BIOS ISR to ensure normal PC / AT system error handling.

Maskable Interrupts

The DONE condition or any VMEbus interrupt request can interrupt the processor using IRQ 5, IRQ 11, or IRQ 12, depending upon a system jumper (see Chapter 2, *Installation and Setup* on page 29). Table 4-5 on page 72 shows a typical VMEbus interrupt handler initialization and response using IRQ 12. The corresponding PC / AT interrupt vectors are located in memory at $1B4 (IRQ5), $1CC (IRQ 11), and $1D0 (IRQ 12). Whichever interrupt is used should have its vector modified to point to the user's custom VMEbus / DONE interrupt handler.

The custom handler can determine whether a VMEbus or DONE interrupt occurred by reading the IRQ Status-ID Register at I / O $14D. Reading the IRQ Status-ID Register will initiate a VMEbus IACK cycle if a VMEbus interrupt is pending. The value read will be the 8-bit ID vector supplied by the VMEbus interrupter. It is important to initialize VMEbus interrupter hardware prior to enabling interrupts so that the proper ID vector is fetched during the IACK cycle. VMEbus hardware interrupt initialization varies according to the VMEbus hardware installed – check the VMEbus board's manual for more information.

During the VMEbus IACK cycle, the processor asserts wait states until the cycle is terminated with a VMEbus DTACK. This prevents any interrupts, DMA transfers, and such from occurring during the IACK cycle.

For VMEbus interrupts, reading the IRQ Status-ID register acknowledges the highest priority interrupt pending that is unmasked, from IRQ 7 (highest priority) to IRQ 1 (lowest priority). For example, if both VMEbus IRQ 1 and IRQ 3 are enabled on the VMIVME-7489 through the Interrupt Mask Register and both VMEbus interrupts occur simultaneously, the IRQ Status-ID read acknowledges only VMEbus IRQ 3.

If a DONE interrupt is pending, the value read from the IRQ Status-ID Register will be the data contained in the DONE Status-ID Register. It is therefore important to initialize the DONE Status-ID Register prior to enabling interrupts. If both a VMEbus and DONE interrupt are pending, priority is given to the VMEbus interrupt.

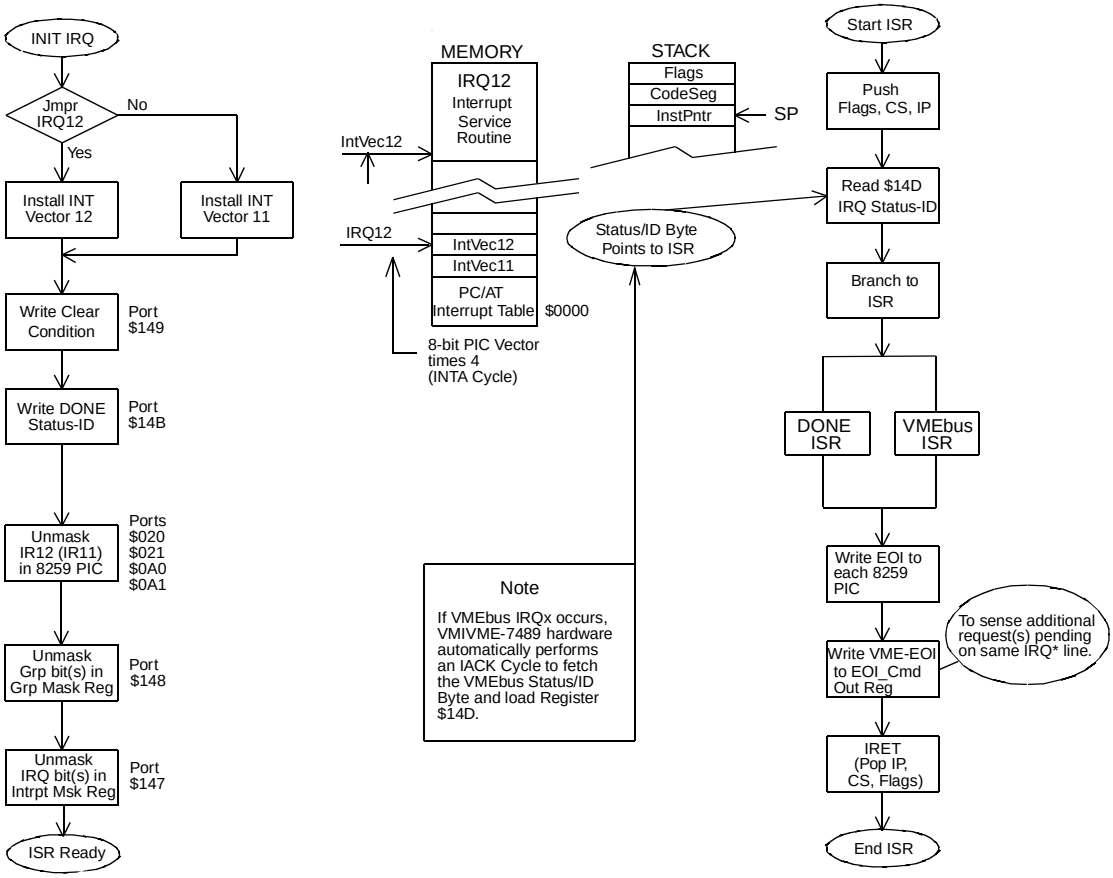


Figure 4-5 Interrupt Response Flowchart

Condition Polling

In some situations, it may be more advantageous for software to poll for various VMEbus conditions rather than wait for interrupts. For example, it may be more efficient to poll for the DONE condition in a VMEbus ISR rather than wait for the DONE interrupt. Another example might be a VMEbus hardware query, wherein the user wants to scan the VMEbus to determine what boards are present. In this case, the scan can perform much faster by polling for the BERR condition after attempting to read an address rather than waiting for the timeout-driven BERR interrupt.

The Clear Condition Register at I/O $149 allows the DONE and BERR conditions to be polled. First, clear the bit(s) to be polled. Afterward, poll for a corresponding DONE or BERR bit to be set in the Group Interrupt Status Register, indicating the condition. Upon detecting the condition, set and then re-clear the corresponding bit in the Clear Condition Register to re-enable the condition. If the Clear Condition bit is set, the condition is masked and will not occur until the bit is cleared.

In some systems, it may also be advantageous to poll VMEbus interrupt signals that are masked off. Thus, a VMIVME-7489 might monitor another VMEbus module's activity without actually performing a VMEbus access. The state of the VMEbus IRQ1*-IRQ7* signals can be read through the IRQ* Status Register. If a VMEbus IRQ* signal is asserted while it is masked off at the Interrupt Mask Register, the condition can only be cleared by accessing the VMEbus interrupter that activated the signal, since reading the IRQ Status-ID Register will *not* clear the condition in this case. Only the VMEbus IRQ* signals enabled to cause VMIVME-7489 interrupts are affected by reading the IRQ Status-ID Register.

Interrupter

The VMIVME-7489 can interrupt another interrupt handler on the VMEbus on one of IRQ7*-IRQ3* lines. The line selection is programmed by writing an IPL2-IPL0 3-bit binary encoded value to the Interrupt Level Select Register. Thereafter, when the CPU writes to the Interrupter Status-ID Register, a VMEbus interrupt is generated on the selected IRQ* line. The interrupt handler acknowledges by reading the Status / ID byte just written.

The interrupter may optionally issue a DONE interrupt to the processor to signal acknowledgment of the VMEbus interrupt. If an interrupt is not desired, the Group Interrupt Status Register can also be polled to determine whether the interrupt was acknowledged.

In summary, to issue a VMEbus Interrupt the following steps should be taken:

1. Select the appropriate level by writing the IPL2-IPL0 value into the Interrupt Level Select Register.
2. Write the 8-bit code into the Interrupter Status-ID Register. The interrupt will be immediately generated.

Requester

Both the Master and Interrupt-Handler functions require the use of the VMEbus. Both request this access through the on-board requester. The requester initiates a bus request then waits for the arbiter to return the BusGrant3* signal on the Bus-Grant daisy chain line.

The ROR bit (bit 4) in the VMEbus Access Control Register defines how the VMIVME-7489 will release the VMEbus. Its power up default is clear, indicating RWD (Release-When-Done) operation. The bus will be released after every transfer. When the ROR bit in the VMEbus Access Control Register is set, the VMIVME-7489 enters Release-On-Request mode and will not release the bus and de-assert BBUSY* until another master has requested the bus. It does this by monitoring all four Bus Request lines BR3*-BR0* to determine if another master is requesting the bus.

The ROR mode is the preferred option when the VMIVME-7489 is the main CPU in the VMEbus. ROR mode decreases arbitration overhead for each transfer, resulting in improved performance.

System Controller

For VMEbus slot-1 applications, the VMIVME-7489 may be jumpered to perform system controller functions including:

- Single Level Arbiter (BR3*)
- 16 MHz System Clock
- SYSRESET* Driver
- IACK Daisy-Chain Driver
- Bus Error Timer

The bus error timer generates a BERR* signal when a nonexistent or nonresponding device is addressed on the VMEbus. The maximum response time is factory set at about 10 µs. The bus error timer function may be disabled by a jumper (see Chapter 2, *Installation and Setup* on page 29for hardware details).

Note that the VMIVME-7489 generates a VMEbus SYSRESET* signal only if it is configured as a system controller, and then only after power up or when the front panel reset switch is pressed. SYSRESET* is *not* generated from the “soft boot” caused by a **<**CTRL+ALT+DEL**>** keyboard reset.

See Chapter 2, *Installation and Setup* on page 29 for information on jumpering the VMIVME-7489 as a system controller.

System Registers

Register Map

There are fifteen active registers located at local I/O addresses $140 through $14F controlling the following functions:

- VMEbus Access Control
- On-board Interrupter
- On-board Interrupt Handler

Table 4-3 details the name, location, access mode, and bit map for each register.

Table 4-3 VMIVME-7489 VMEbus Register Map

REGISTER NAME	ACCESS	I/O ADR	D7	D6	D5	D4	D3	D2	D1	D0
Product ID	R	$140	0	0	0	0	0	0	1	0
VMEbus Access Control	W	$140	Status LED	X	X	ROR	SysFail	Big Endian	VME Enable	X
VMEbus Standard Access	W	$141	A23	A22	A21	A20	A19	A18	A17	A16
VMEbus Extended/AM Access	W	$142	This is a 16-bit register – see below for bit map							
Reserved		$143-$146								
Interrupt Mask	R/W	$147	X	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1
Group Interrupt Mask	R/W	$148	SysFail	X	X	ACFail	BERR	VME	X	DONE
Clear Condition	R/W	$149	X	X	SoftNMI	X	BERR	X	X	DONE
Reserved		$14A								
DONE Status-ID	R/W	$14B	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0
Group Interrupt Status	R	$14C	SysFail	X	SoftNMI	ACFail	BERR	VME	X	DONE
Soft NMI Select	W	$14C	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0
IRQ Status-ID	R	$14D	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0
EOI Command	W	$14D	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0
Interrupt Level Select	R/W	$14E	Flag	X	X	X	X	IPL2	IPL1	IPL0
IRQ* Status Register	R	$14F	X	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1
Interrupter Status-ID	W	$14F	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

REGISTER NAME	ACCESS	I/O ADR	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
VMEbus Extended / AM Access	W	$142	A31	A30	A29	A28	A27	A26	A25	A24	X	X	AM5	AM4	AM3	AM2	AM1	AM0

Note

Note An *X* denotes a reserved location. An *XD* bit can be either set or clear.

All registers are 8 bits wide, except the 16-bit wide VMEbus Extended / AM Access Register. All registers are accessible only from the on-board processor, not from any other VMEbus master.

Registers listed as “Reserved” should not be accessed, since they have valid uses in the other VMIC products. When read, register bits marked “Reserved” will return an indeterminate value. When written, reserved bits should be cleared (that is, write a zero to all reserved bits in a write register).

Register Details

Product ID Register

The Product ID Register is a read-only byte register at I / O address $140. This register always contains the value $02, which uniquely identifies the VMIVME-7489 from other VMIC VMEbus products.

Product ID Register (Read-Only byte at I/O $140)							
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0

VMEbus Access Control Register

The VMEbus Access Control Register is a write-only byte register at I / O address $140. The five active bits in this register control certain global aspects of VMEbus activity. It is recommended that the programmer maintain a soft copy of this register for status, since the register cannot be read. All active bits in this register are cleared on power up or hard reset.

VMEbus Access Control Register (Write-Only byte at I/O $140)							
D7	D6	D5	D4	D3	D2	D1	D0
Status LED	X	X	ROR	SysFail	Big Endian	VME Enable	X

VMEbus Access Control Register: Status LED bit

Bit 7 controls the front panel LED. Note that the default state is on. Thus, the front panel LED will be on after power up or hard reset, and will not turn off until this bit is set.

D7	Status LED Bit Function
0	Front panel Status LED ON (default)
1	Front panel Status LED OFF

VMEbus Access Control Register: ROR bit

When the VMIVME-7489 requests bus mastership, the on-board requester initiates a VMEbus request and eventually gets a Bus Grant. The requester then signals the VMIVME-7489 to take control of the VMEbus.

D4	ROR Bit Function
0	RWD (Release When Done) VMEbus requester (default)
1	ROR (Release On Request) VMEbus requester

When the data transfer task is finished, the requester has two ways to release control of the bus. Bit 4 specifies whether to release control of the bus immediately upon completion (Release When Done, or RWD) or to maintain control of the bus until another master requests control (Release On Request, or ROR). RWD mode is the default for maximum compatibility with other masters, but if the VMIVME-7489 is the only master, or is usually the master, then ROR mode can speed up repeated VMEbus accesses since less overhead is involved every transfer. See *Requester* on page 75 for a more detailed discussion of the VMIVME-7489 requester function.

VMEbus Access Control Register: SysFail bit

Bit 3 controls the VMEbus SYSFAIL* line. The default clear state leaves the SYSFAIL* signal active, according to the VMEbus specification for a system controller.

D3	SysFail Bit Function
0	The VMEbus SYSFAIL* line is driven low / active (default)
1	The VMEbus SYSFAIL* line is allowed to be pulled high / inactive

Since other controllers on the VMEbus may not function normally while the SYSFAIL* line is active, VMIC recommends that the SysFail bit be set immediately after power up or reset initialization routines, even if no VMEbus activity is planned. This is especially important if the VMIVME-7489 is not the slot-1 controller, since an active SYSFAIL* line may prevent the controller from performing its normal functions.



Note THE VMIVME-7489 ACTIVATES THE VMEbus SYSFAIL* SIGNAL UPON POWER UP OR RESET. IT IS IMPORTANT TO DEACTIVATE SYSFAIL* AS SOON AS POSSIBLE TO ALLOW NORMAL VMEbus ACTIVITY.

VMEbus Access Control Register: Big Endian bit

The 80486 processor uses Intel's little-endian method for transferring a multiple-byte group between its internal registers and external memory. The VMEbus, however, expects Motorola's big-endian method for transferring a multiple-byte group to the same locations. The VMIVME-7489 has hardware on-board to support either transfer method, and the byte-swapping hardware is controlled by bit 2 of this register.

D2	Big Endian Bit Function
0	Use little-endian multiple-byte transfers (default)
1	Use big-endian multiple-byte transfers

Any byte swapping is transparent both to the on-board CPU and to VMEbus devices. See the discussion on byte swapping in Section for more details.

VMEbus Access Control Register: VME Enable bit

Bit 1 controls VMEbus access. While this bit is clear – the default state after power up or reset – the VMIVME-7489 cannot access the VMEbus. While VMEbus access is disabled, the VMIVME-7489 VMEbus interface registers may still be accessed, but no data transfers to or from the VMEbus will occur. Likewise, no VMEbus signal can cause an interrupt on the VMIVME-7489.

D1	VME Enable Bit Function
0	Disable VMEbus access (default)
1	Enable VMEbus access

The VMIVME-7489 is never completely isolated from the VMEbus, however, regardless of the VME Enable bit. The SysFail bit in this same register always functions. Also, the VMIVME-7489 will still generate a VMEbus SYSRESET* when the front panel reset button is pressed, as long as the VMIVME-7489 is configured as the system controller. Likewise, if the VMIVME-7489 is not the system controller, it will still respond to a VMEbus SYSRESET* regardless of the VME Enable bit status.

VMEbus Standard Access Register

The VMEbus Standard Access Register is a write-only byte register at I/O address $141. The bits in this register control the VMEbus A16-A23 address lines when accessing the VMEbus through the Real Mode VMEbus Window. After power up or reset, the value in this register is indeterminate.

VMEbus Standard Access Register (Write-Only byte at I/O $141)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
A23	A22	A21	A20	A19	A18	A17	A16

This register is only active when accessing the VMEbus through the Real Mode VMEbus Window and even then only if the VMEbus Address Modifier indicates 24-bit or higher VMEbus access (see the next register description for a discussion of VMEbus AM codes and higher address bit programming). If these conditions occur, this register provides the upper eight VMEbus address bits A16-A23.

An easy way to conceptualize the function of this register is to think of it as selecting the 64 Kbyte segment of VMEbus 24-bit addressing space that appears in the Real Mode VMEbus Window. A value of $00 here makes the bottom 64 Kbyte space available through the window, while a value of $FF makes the topmost 64 Kbyte space available.

This register only supplies VMEbus address bits A16-A23. If a VMEbus addressing mode is selected that requires even higher address bits (for example, 32-bit Extended addressing), the higher-order bits are supplied by the VMEbus Extended/AM Register. In this case, the two registers function together to determine which 64 Kbyte segment of up to 4 Gbytes of VMEbus addressing space appears within the Real Mode VMEbus Window.

Note again the restrictions on this register: it only functions when accessing the VMEbus in Real Mode and only if the VMEbus Address Modifier indicates 24-bit or higher addressing. When accessing the VMEbus using Short I/O (16-bit) addressing, all 64 Kbytes of Short I/O space are available through the window, so this register is unnecessary. Likewise, when accessing the VMEbus from the Protected Mode VMEbus Window, all 24 bits of VMEbus Standard addressing space are available at once through the window, so this register again becomes unnecessary.

VMEbus Extended/AM Access Register

The VMEbus Extended / AM Access Register is a write-only word register at I/O address $142. Note that this is the only VMEbus control register that must only be accessed as a 16-bit word rather than as an 8-bit byte. The 14 active bits in this register control the VMEbus A24-A31 address lines and the AM0-AM5 address modifier bits. After power up or reset, the value in this register is indeterminate.

VMEbus Extended/AM Access Register (Write-Only word at I/O $142)															
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
A31	A30	A29	A28	A27	A26	A25	A24	X	X	AM5	AM4	AM3	AM2	AM1	AM0

This register contains both the extended VMEbus address lines A24-A31 as well as the VMEbus AM Code. The Address Modifier bits AM0-AM5 here correspond directly to the VMEbus AM0-AM5 bits. The AM Code determines the type of transfer the VMIVME-7489 will perform: Extended, Standard, Short, or User Defined. The AM code must *always* be initialized to a value corresponding to the desired VMEbus access mode. The VMEbus Specification has a complete list of Address Modifier codes, but Table 4-4 below lists the most common and useful codes.

Table 4-4 Valid VMEbus AM Codes

OPTION CODE	AM CODE	ADDRESSING MODE
A32	$09	Extended Nonprivileged Data Access
A32	$0A	Extended Nonprivileged Program Access
A32	$0D	Extended Supervisory Data Access
A32	$0E	Extended Supervisory Program Access
A24	$39	Standard Nonprivileged Data Access
A24	$3A	Standard Nonprivileged Program Access
A24	$3D	Standard Supervisory Data Access
A24	$3E	Standard Supervisory Program Access
A16	$29	Short Nonprivileged Access (I/O Cycle)
A16	$2D	Short Supervisory Access (I/O Cycle)
Reserved	$19	VMIC Reserved DMA Cycle
Reserved	$1D	VMIC Reserved DMA Cycle (TC)

The MSB of this register directly controls the upper VMEbus address lines A24-A31. Of course, if the selected VMEbus AM code does not indicate Extended 32-bit addressing, the high-order address bits controlled by this register will be ignored by the VMEbus. If the user has selected a 32-bit VMEbus addressing mode, then the MSB will provide the Extended Address lines to the VMEbus.

An easy way to conceptualize the function of the MSB of this register is to think of it as controlling the 16 Mbyte segment of VMEbus 32-bit addressing space that appears in the Protected Mode VMEbus Window. A value of $00 here makes the bottom 16 Mbyte space available through the window, while a value of $FF makes the topmost 16 Mbyte space available. The value in this register also affects A32 access through the Real Mode VMEbus Window, although in that case the upper address bits A16-A31 are controlled jointly between this register and the VMEbus Standard Access Register described previously.

Interrupt Mask Register

The Interrupt Mask Register is a read/write byte register at I/O address $147. The seven active bits in this register can prevent individual VMEbus interrupt lines from interrupting the VMIVME-7489. All active bits in this register are cleared on power up or hard reset.

Interrupt Mask Register (Read/Write byte at I/O $147)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
X	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1

This register is used to individually mask any of the seven VMEbus interrupts to the interrupt handler on the VMIVME-7489. A clear bit indicates that the VMIVME-7489 will not respond to that particular VMEbus interrupt.

Interrupt acknowledgment is prioritized such that if more than one IRQ* line is enabled, the VMIVME-7489 acknowledges the higher priority interrupt first, with IRQ7* having the highest priority and IRQ1* the lowest.

Note that the power up/reset default value for this register renders all VMEbus IRQ* interrupts disabled.

Group Interrupt Mask Register

The Group Interrupt Mask Register is a read / write byte register at I / O address $148. The five active bits in this register disable individual VMEbus system interrupts from interrupting the VMIVME-7489. All active bits in this register are cleared on power up or hard reset.

Group Interrupt Mask Register (Read/Write byte at I/O $148)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
SysFail	X	X	ACFAIL	BERR	VMEbus	X	DONE

When clear, each active bit in this register prevents its associated interrupt source from interrupting the VMIVME-7489 as shown below:

SysFail — When set, this bit enables the VMEbus SYSFAIL* signal to reach the NMI input to the 80486 CPU.

ACFAIL — When set, this bit enables the gating of the VMEbus ACFAIL* line onto the NMI input to the 80486 CPU.

BERR — When set, this bit enables the VMEbus BERR bus error interrupt to reach the NMI input to the 80486 CPU as a result of a BERR* cycle or bus time-out.

VMEbus — When set, this bit enables IRQ1*-IRQ7* interrupts from the VMEbus (although they may also be controlled individually by the Interrupt Mask Register described previously).

DONE — When set, this bit enables the DONE interrupt to reach the slave PIC input.

Clear Condition Register

The Clear Condition Register is a read / write byte register at I / O address $149. The three active bits in this register allow certain interrupt conditions to be cleared in poll mode. All active bits in this register are cleared on power up or hard reset.

Clear Condition Register (Read/Write byte at I/O $149)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
X	X	SoftNMI	X	BERR	X	X	DONE

When clear, the active bits in this register clear their associated condition and also prevent that condition from occurring, effectively acting as another interrupt mask function.

The reason for this register is to allow the SoftNMI, BERR, and DONE interrupts to be handled using a software polling technique rather than with hardware interrupts. The poll mode routine should mask the appropriate interrupt in the Group Interrupt Mask Register, then watch the Group Interrupt Status Register for the interrupt flag. When the action is complete, the interrupt condition should be cleared by setting the corresponding bit in this register.

Since a clear bit here disables that condition, it should be set again immediately after being cleared to re-enable the condition.

DONE Status-ID Register

The DONE Status-ID Register is a read/write byte register at I/O address $14B. All bits in this register are active and together define the status/ID value returned by the DONE interrupt. All bits in this register are of indeterminate value after power up or hard reset.

DONE Status-ID Register (Read/Write byte at I/O $14B)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

This register provides the status-ID byte to the on-board interrupt handler when acknowledging a DONE interrupt. It is used to vector to the appropriate interrupt routine. This is a local register only and no VMEbus access is performed.

Group Interrupt Status Register

The Group Interrupt Status Register is a read-only byte register at I/O address $14C. The six active bits in this register indicate general interrupt conditions.

Group Interrupt Status Register (Read-Only byte at I/O $14C)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
SysFail	X	SoftNMI	ACFAIL	BERR	VMEbus	X	DONE

The Group Interrupt Status Register allows the processor to determine the source of the pending interrupts and also allows the associated VMEbus conditions to be polled. A set bit indicates the corresponding condition is present, although the actual interrupt may be masked off. See Section of this chapter for details concerning interrupt handling and this register.

Soft NMI Select Register

The Soft NMI Select Register is a write-only byte register at I/O address $14C. The bits of this register have no meaning – any data written here will cause a Soft NMI interrupt, if enabled.

Soft NMI Select Register (Write-Only byte at I/O $14C)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
X	X	X	X	X	X	X	X

IRQ Status-ID Register

The IRQ Status-ID Register is a read-only byte register at I/O address $14D. All the bits in this register are active and together constitute a value that is the vector obtained from the VMEbus or DONE interrupter.

IRQ Status-ID Register (Read-Only byte at $14D)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

The VMEbus IRQ and DONE interrupt service routines should read this register to get the vector from either the VMEbus or DONE interrupt sources. Priority is given to VMEbus interrupts. If a VMEbus interrupt is being serviced, an automatic IACK cycle is performed over the VMEbus and the Status/ID byte of the active interrupter is retrieved through this register.

EOI Command Register

The EOI Command Register is a write-only byte register at I/O address $14D. The bits in this register have no real meaning, since any value written here will perform the End-Of-Interrupt function.

EOI Command Register (Write-Only byte at I/O $14D)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
X	X	X	X	X	X	X	X

This register is a local switch register with no specific pattern required on the data lines. It should be written at the end of the interrupt routine to clear the just-serviced interrupt from a local latch and at the same time allow toggling of additional interrupts which may be pending from the VMEbus. This register allows the coupling of edge sensitive PC / AT interrupts with the level sensitive mode of VMEbus interrupts.

This register should not be confused with the EOI commands that must also be sent to the 8259 PICs according to the usual PC / AT architecture.

In case of a RORA Interrupter, the EOI Command should be issued at least 2 µs after reading the Interrupter's Status / ID byte (VMEbus Rule 4.8).

Interrupt Level Select Register

The Interrupt Level Select Register is a read / write byte register at I / O address $14E. The four active bits in this register control the level of the VMEbus interrupts generated by the VMIVME-7489. The value of all bits in this register are indeterminate after power up or hard reset.

Interrupt Level Select Register (Read/Write byte at I/O $14E)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Flag					IPL2	IPL1	IPL0

This register is used by the on-board interrupter to select the IRQ level on which to issue an interrupt on the VMEbus. The IPL bits control the IRQ level according to Table 4-5 below. This register is not altered by the VMEbus IACK* cycle.

Table 4-5 IRQ Level IPL Values

IPL2	IPL1	IPL0	IRQ LEVEL
0	0	0	None
0	0	1	None
0	1	0	None
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

The Flag bit has no specific function, but will retain and read back any value written to it. It is recommended that the Flag be used as a general purpose “busy” semaphore to exclude other tasks from using the interrupter while it is already being used.

IRQ* Status Register

The IRQ* Status Register is a read-only byte register at I/O address $14F. The seven active bits in this register simply indicate the current state of the IRQ* lines on the VMEbus.

IRQ* Status Register (Read-Only byte at I/O $14F)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
X	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1

The processor can read the state of all the VMEbus interrupt lines IRQ1*-IRQ7* through this register. The logic values are also maintained, meaning that a cleared bit indicates the corresponding IRQ* line is low / active.

Interrupter Status-ID Register

The Interrupter Status-ID Register is a write-only byte register at I/O address $14F. All bits in this register are active and together constitute the status-ID value presented to the VMEbus interrupter during a VMEbus IACK cycle.

Interrupter Status-ID Register (Write-Only byte at I/O $14F)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

This register is used by the on-board interrupter and supplies the status-ID byte to a VMEbus interrupt handler during an IACK* cycle. The VMIVME-7489 Interrupter is a ROAK (Release On Acknowledge) interrupter, so the VMIVME-7489 releases the IRQ* line when the handler acknowledges its interrupt by reading this value.

If DONE interrupts are enabled, a DONE interrupt is also issued to the local CPU when the VMEbus interrupt handler reads this register during the IACK* cycle.

APPENDIX

A

Connector Pinouts

Contents

Ethernet Connector Pinout 91
Floppy Drive Connector Pinout 92
IDE Hard Drive Connector Pinout 93
Keyboard Connector Pinout 94
PC / 104 Connector Pinout 95
Printer Connector Pinout 97
Serial Connector Pinout 98
Video Connector Pinout 99
VMEbus Connector Pinout 100

Introduction

The VMIVME-7489 PC / AT Compatible VMEbus Controller has several connectors for its many I / O ports. Figure A-1 on page 90 shows the locations of the connectors. Wherever possible, the VMIVME-7489 uses connectors and pinouts typical for any desktop PC. This ensures maximum compatibility with a minimum of confusion.

Connector diagrams in this appendix are generally shown in a natural orientation with the controller board mounted in a VMEbus chassis. Not all connectors are available on all controller boards.

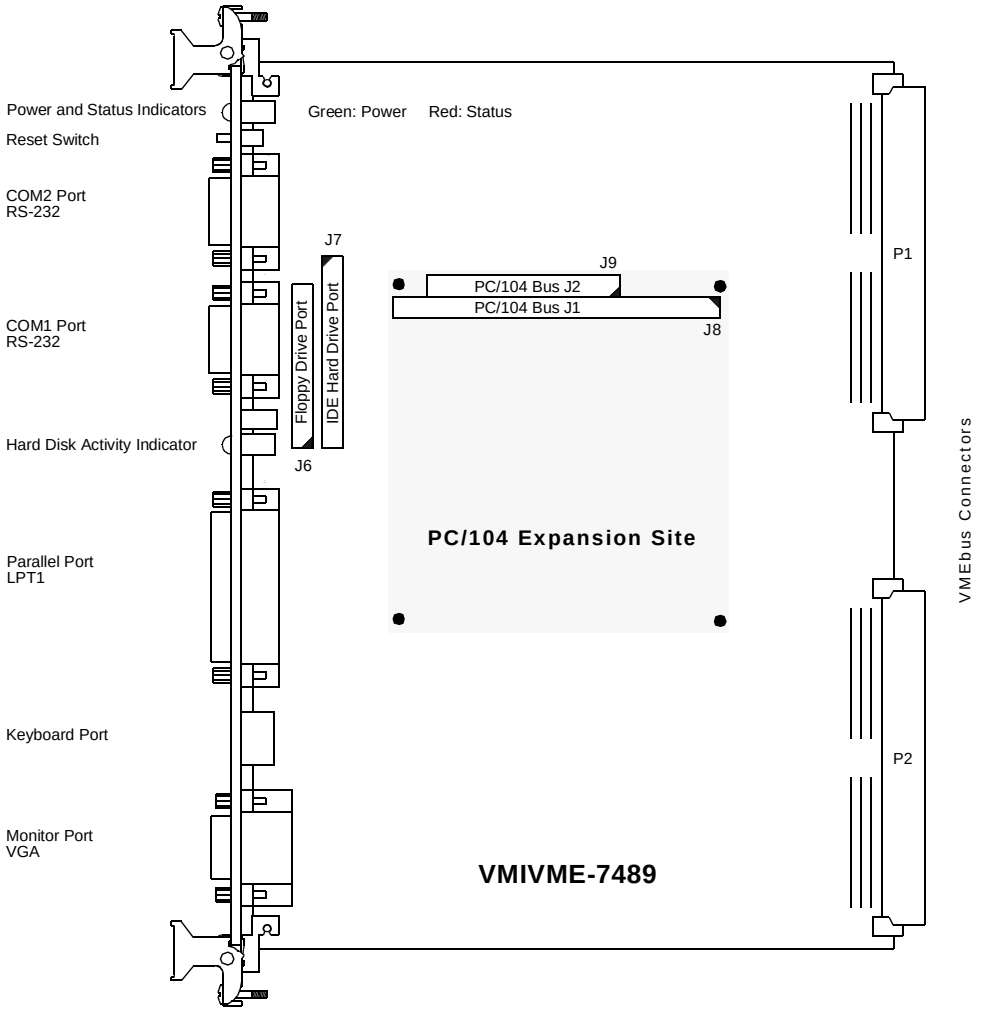
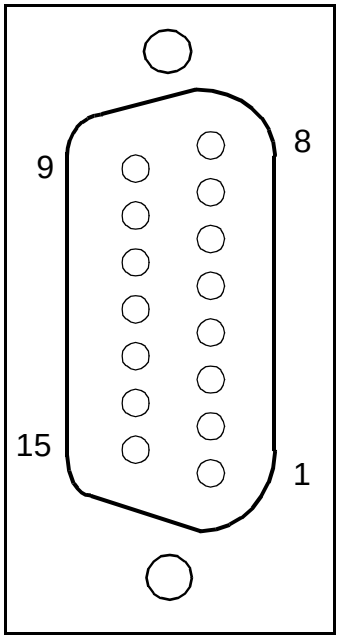


Figure A-1 VMIVME-7489 Connector Locations

Ethernet Connector Pinout

On controller boards with an Ethernet option, a D15 female connector provides the Ethernet AUI interface. The pinout diagram for the Ethernet connector is shown in Figure A-2.

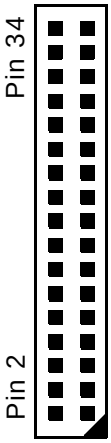


ETHERNET CONNECTOR		
PIN	**DIR**	**FUNCTION**
1		GND
2	In	CD+
3	Out	TX+
4		GND
5	In	RX+
6		GND
7		Reserved
8		GND
9	In	CD-
10	Out	TX-
11		GND
12	In	RX-
13		+12 VDC
14		GND
15		Reserved

Figure A-2 Ethernet Connector Pinout

Floppy Drive Connector Pinout

The floppy drive connector is a dual-row 34-pin header connector. Pin 1 marks the beginning of the odd-numbered row. Most standard PC compatible floppy disk drives use this pinout, which is shown in Figure A-3.

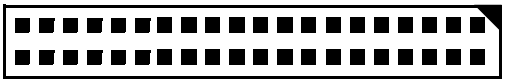


FLOPPY DRIVE CONNECTOR					
PIN	**DIRECTION**	**FUNCTION**	**PIN**	**DIRECTION**	**FUNCTION**
1		Ground	18	From Controller	Step Motor Direction
2	From Controller	Drive Density 0 (not connected on some controllers)	19		Ground
3		Ground	20	From Controller	Step Pulse
4		Reserved	21		Ground
5		Ground	22	From Controller	Write Data
6	From Controller	Drive Density 1 (not connected on some controllers)	23		Ground
7		Ground	24	From Controller	Write Enable
8	From Drive	Index	25		Ground
9		Ground	26	From Drive	Track 0
10	From Controller	Motor Enable A	27		Ground
11		Ground	28	From Drive	Write Protect
12	From Controller	Drive Select B	29		Ground
13		Ground	30	From Drive	Read Data
14	From Controller	Drive Select A	31		Ground
15		Ground	32	From Controller	Select Head 1
16	From Controller	Motor Enable B	33		Ground
17		Ground	34	From Drive	Disk Change

Figure A-3 Floppy Drive Connector Pinout

IDE Hard Drive Connector Pinout

Figure A-4 describes the pin assignment for the 40-pin IDE / ATA hard drive header connector. Like the floppy header connector it has an odd-numbered row and an even-numbered row.

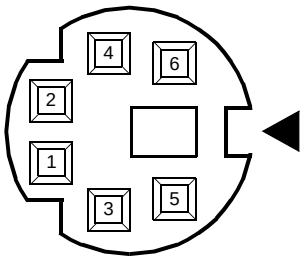


PIN	DIRECTION	DESCRIPTION	PIN	DIRECTION	DESCRIPTION
1	Out	Reset Drive	2	Out	Signal Ground
3	In / Out	Bidirectional Data 07	4	In / Out	Bidirectional Data 08
5	In / Out	Bidirectional Data 06	6	In / Out	Bidirectional Data 09
7	In / Out	Bidirectional Data 05	8	In / Out	Bidirectional Data 10
9	In / Out	Bidirectional Data 04	10	In / Out	Bidirectional Data 11
11	In / Out	Bidirectional Data 03	12	In / Out	Bidirectional Data 12
13	In / Out	Bidirectional Data 02	14	In / Out	Bidirectional Data 13
15	In / Out	Bidirectional Data 01	16	In / Out	Bidirectional Data 14
17	In / Out	Bidirectional Data 00	18	In / Out	Bidirectional Data 15
19	Out	Signal Ground	20	None	Unused, Keying Position
21		Reserved	22	Out	Signal Ground
23	Out	Write Strobe	24	Out	Signal Ground
25	Out	Read Strobe	26	Out	Signal Ground
27		Reserved	28	Out	Address Latch Enable
29		Reserved	30	Out	Signal Ground
31	In	Interrupt Request #14	32	In	16-bit Data Word Size
33	Out	Address Line #1	34	In	Diagnostic Test Passed
35	Out	Address Line #0	36	Out	Address Line #2
37	Out	Chip Select #0	38	Out	Chip Select #1
39	In	Slave / Activity Status	40	Out	Signal Ground

Figure A-4 IDE Hard Drive Connector Pinout

Keyboard Connector Pinout

The keyboard connector is a standard 6-pin female mini-DIN PS/2 style connector shown in Figure A-5; an adapter is supplied to connect a keyboard with a larger PC/AT-style connector to the VMIVME-7489.



KEYBOARD CONNECTOR		
PIN	**DIR**	**FUNCTION**
1	In/Out	Data
2		Reserved
3		Ground
4		+5 V
5	Out	Clock
6		Reserved
Shield		Chassis Ground

Figure A-5 Keyboard Connector Pinout

PC/104 Connector Pinout

Controllers supporting PC/104 expansion sites have a pair of header connectors which function as standard PC/104 connectors J1 and J2. The PC/104 specification designates pins by rows lettered A-D and numbers. The pin numbering is rather unusual; please refer to Figure A-6 to determine the physical location of the pins in Table A-6.

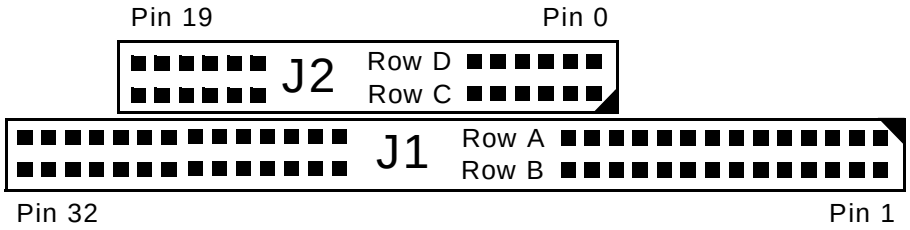


Figure A-6 PC/104 Connector Diagram

Table A-6 PC/104 Connector Pinout

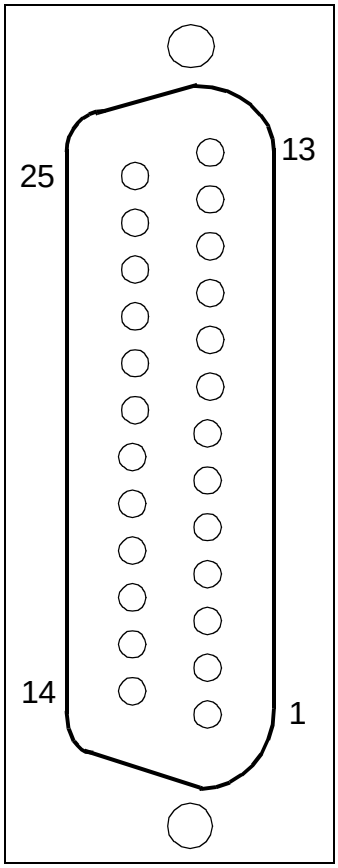
PIN	J1 ROW A	J1 ROW B	J2 ROW C	J2 ROW D
0			0 V	0 V
1	IOCHCHK	0 V	SBHE	MEMCS16
2	SD7	RESETDRV	LA23	IOCS16
3	SD6	+5 V	LA22	IRQ10
4	SD5	IRQ9	LA21	IRQ11
5	SD4	-5 V	LA20	IRQ12
6	SD3	DRQ2	LA19	IRQ15
7	SD2	-12 V	LA18	IRQ14
8	SD1	ENDXFR	LA17	DACK0
9	SD0	+12 V	MEMR	DRQ0
10	IOCHRDY	(key)	MEMW	DACK5
11	AEN	SMEMW	SD8	DRQ5
12	SA19	SMEMR	SD9	DACK6
13	SA18	IOW	SD10	DRQ6
14	SA17	IOR	SD11	DACK7
15	SA16	DACK3	SD12	DRQ7
16	SA15	DRQ3	SD13	+5 V
17	SA14	DACK1	SD14	MASTER
18	SA13	DRQ1	SD15	0 V
19	SA12	REFRESH	(key)	0 V

Table A-6 PC/104 Connector Pinout (Continued)

PIN	J1 ROW A	J1 ROW B	J2 ROW C	J2 ROW D
20	SA11	SYSCLK		
21	SA10	IRQ7		
22	SA9	IRQ6		
23	SA8	IRQ5		
24	SA7	IRQ4		
25	SA6	IRQ3		
26	SA5	DACK2		
27	SA4	TC		
28	SA3	BALE		
29	SA2	+5 V		
30	SA1	OSC		
31	SA0	0 V		
32	0 V	0 V		

Printer Connector Pinout

The printer port shown in Figure A-7 uses a D25 female connector typical of any PC / AT system. Note that software keys (or "dongles") that attach to the printer port may require the supplied adapter for use on the VMIVME-7489. If so, simply attach the supplied adapter to the printer port first, then plug the software key into the adapter.

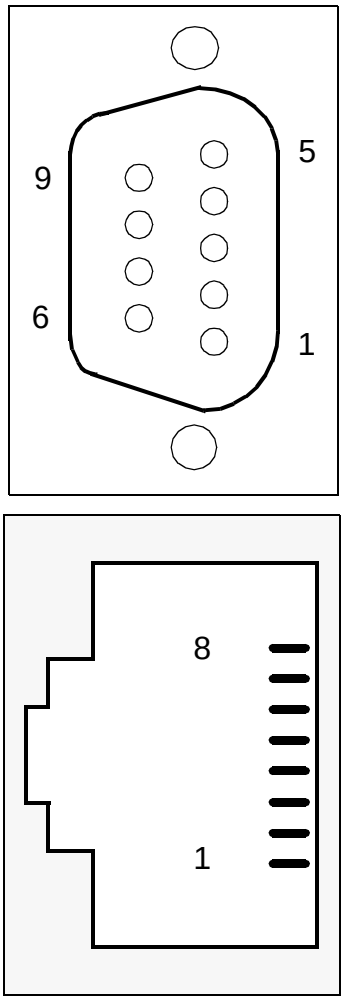


PRINTER CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1	In / Out	Data Strobe
2	In / Out	Bidirectional Data D0
3	In / Out	Bidirectional Data D1
4	In / Out	Bidirectional Data D2
5	In / Out	Bidirectional Data D3
6	In / Out	Bidirectional Data D4
7	In / Out	Bidirectional Data D5
8	In / Out	Bidirectional Data D6
9	In / Out	Bidirectional Data D7
10	In	Acknowledge
11	In	Device Busy
12	In	Out of Paper
13	In	Device Selected
14	Out	Auto Feed
15	In	Error
16	Out	Initialize Device
17	In	Device Ready for Input
18		Signal Ground
19		Signal Ground
20		Signal Ground
21		Signal Ground
22		Signal Ground
23		Signal Ground
24		Signal Ground
25		Signal Ground
Shield		Chassis Ground

Figure A-7 Printer Connector Pinout

Serial Connector Pinout

Each standard RS-232 serial port connectors is either a D9 male as shown in the upper drawing in Figure A-8 or an RJ45 jack like the lower drawing in Figure A-8. The controller boards with RJ45 jacks are supplied with adapters to connect standard D9 serial peripherals. Because only eight of the nine RS-232 signals are brought out to the RJ45 connectors, the signal at pin 8 (pin 1 on the D9 adapter) is assigned to either the DCD signal or the RI signal, depending upon a jumper selection (see Chapter 2, *Installation and Setup* on page 29 for complete board jumper information).



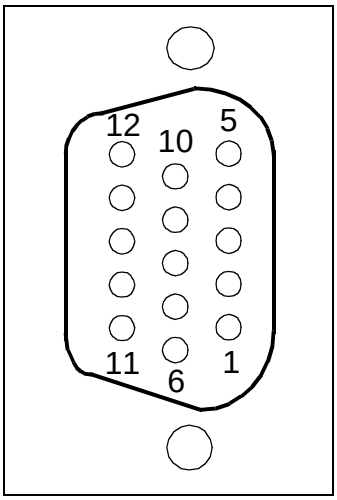
SERIAL COM PORT CONNECTORS				
D9 PIN	**RJ45 PIN**	**DIR**	**RS-232 SIGNAL**	**FUNCTION**
1*	8*	In	DCD	Data Carrier Detect
2	2	In	RX	Receive Data
3	3	Out	TX	Transmit Data
4	4	Out	DTR	Data Terminal Ready
5	1		GND	Signal Ground
6	5	In	DSR	Data Set Ready
7	6	Out	RTS	Request to Send
8	7	In	CTS	Clear to Send
9*	8*	In	RI	Ring Indicator
Shield	Shield			Chassis Ground

* RJ45 pin 8 is either assigned to the DCD or the RI signal, depending upon a jumper on the controller board. If the supplied RJ45-to-D9 adapter is being used, D9 pin 9 is not connected and D9 pin 1 is the DCD or RI signal from RJ45 pin 8.

Figure A-8 Serial Connector Pinouts

Video Connector Pinout

The video port uses a standard high-density D15 VGA connector. Figure A-9 shows the pinout.



VIDEO CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1	Out	Red
2	Out	Green
3	Out	Blue
4		Reserved
5		Ground
6		Ground
7		Ground
8		Ground
9		Reserved
10		Ground
11		Reserved
12		Reserved
13	Out	Horizontal Sync
14	Out	Vertical Sync
15		Reserved
Shield		Chassis Ground

Figure A-9 Video Connector Pinout

VMEbus Connector Pinout

Figure A-10 shows the location of the VMEbus P1 and P2 connectors and their orientation. Table A-7 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.

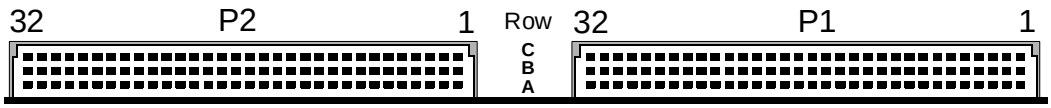


Figure A-10 VMEbus Connector Diagram

Table A-7 VMEbus Connector Pinout

PIN NUMBER	P1 ROW A SIGNAL	P1 ROW B SIGNAL	P1 ROW C SIGNAL	P2 ROW B SIGNAL
1	D00	BBSY	D08	+5 V
2	D01	BCLR	D09	GND
3	D02	ACFAIL	D10	Reserved
4	D03	BG0IN	D11	A24
5	D04	BG0OUT	D12	A25
6	D05	BG1IN	D13	A26
7	D06	BG1OUT	D14	A27
8	D07	BG2IN	D15	A28
9	GND	BG2OUT	GND	A29
10	SYSCLK	BG3IN	SYSFAIL	A30
11	GND	BG3OUT	BERR	A31
12	DS1	BR0	SYSRESET	GND
13	DS0	BR1	LWORD	+5 V
14	WRITE	BR2	AM5	D16
15	GND	BR3	A23	D17
16	DTACK	AM0	A22	D18
17	GND	AM1	A21	D19
18	AS	AM2	A20	D20
19	GND	AM3	A19	D21

Table A-7 VMEbus Connector Pinout (Continued)

PIN NUMBER	P1 ROW A SIGNAL	P1 ROW B SIGNAL	P1 ROW C SIGNAL	P2 ROW B SIGNAL
20	IACK	GND	A18	D22
21	IACKIN	SERCLK	A17	D23
22	IACKOUT	SERDAT	A16	GND
23	AM4	GND	A15	D24
24	A07	IRQ7	A14	D25
25	A06	IRQ6	A13	D26
26	A05	IRQ5	A12	D27
27	A04	IRQ4	A11	D28
28	A03	IRQ3	A10	D29
29	A02	IRQ2	A09	D30
30	A01	IRQ1	A08	D31
31	-12 V	+5 V STDBY	+12 V	GND
32	+5 V	+5 V	+5 V	+5 V

Maintenance

Maintenance

This section provides information relative to the care and maintenance of VMIC's products. If the products malfunction, verify the following:

- Software
- System configuration
- Electrical connections
- Jumper or configuration options
- Boards are fully inserted into their proper connector location
- Connector pins are clean and free from contamination
- No components of adjacent boards are disturbed when inserting or removing the board from the chassis
- 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*.

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