VMIVME-7589A

Single Board AMD-K6/Pentium Processor-Based VMEbus CPU

Product Manual

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

500-107589-000 Rev. A
25-Jan-1999

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

COPYRIGHT AND TRADEMARKS

(I/O man figure)

(Instant OPC wizard logo)

(IOWorks man figure)

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

Table of Contents

List of Figures

List of Tables

Overview

Introduction

VMIC's VMIVME-7589A is a complete IBM PC / AT-compatible AMD-K6 / Pentium processor-based computer with the additional benefits of single Eurocard construction and full compatibility with the VMEbus Specification Rev. C.1. The VMIVME-7589A with advanced VMEbus interface and RAM that is dual-ported to the VMEbus, is ideal for multiprocessor applications.

The single CPU board functions as a standard PC / AT, executing a PC / AT-type power-on self-test, then boots up MS-DOS, Windows 3.11, Windows 95, Windows NT, or any other PC / AT-compatible operating system. The PC / AT mode of the VMIVME-7589A is discussed in Chapter 3 of this manual.

The VMIVME-7589A also operates as a VMEbus controller and interacts with other VMEbus modules by adding the optional PCI-to-VMEbus bridge mezzanine.

The VMIVME-7589A may be accessed as a VMEbus slave board. The VMEbus functions are available by programming the VMIVME-7589A's PCI-to-VMEbus bridge according to the references defined in this volume and / or in the second volume dedicated to the optional PCI-to-VMEbus interface board titled: *VMIVME-7589A Tundra Universe II™-Based VMEbus Interface Option Product Manual (document No. 500-107589-001 Rev. 1.1).*

The VMIVME-7589A programmer may quickly and easily control all the VMEbus functions simply by linking to a library of VMEbus interrupt and control functions. This library is available with VMIC's VMISFT-9420 IOWorks Access software for Windows NT users.

The VMIVME-7589A also provides capabilities beyond the features of a typical IBM PC / AT compatible CPU including general-purpose timers, a programmable watchdog timer, a bootable Flash Disk system, and nonvolatile, battery-backed SRAM. These features make the unit ideal for embedded applications. These nonstandard PC / AT functions are discussed in Chapter 4 of this manual.

Organization of the Manual

This manual is composed of the following chapters and appendices:

Chapter 1 - VMIVME-7589A Features and Options describes the features of the base unit followed by descriptions of the associated features of the unit in operation on a VMEbus. The interface configuration includes the option to attach a VMEbus interface.

Chapter 2 - Installation and Setup describes unpacking, inspection, hardware jumper settings, connector definitions, installation, system setup, and operation of the VMIVME-7589A.

Chapter 3 - PC/AT Functions describes the unit design in terms of the standard PC memory and I/O maps, along with the standard interrupt architecture.

Chapter 4 - Embedded PC/RTOS Features describes the unit features that are beyond standard PC/AT functions.

Chapter 5 - Maintenance provides information relative to the care and maintenance of the unit.

Appendix A - Connector Pinouts illustrates and defines the connectors included in the unit's I/O ports.

Appendix B - System Drive Software includes detailed instructions for the installation of the drivers during installation of Windows for Workgroups Version 3.11, Windows 95, or Windows NT (Versions 3.5x and 4.0) operating systems.

Appendix C - LANWorks BIOS describes the menus and options associated with the LANWorks BIOS.

Appendix D - Basic Input/Output System describes the menus and options associated with the Award (system) BIOS.

Appendix E - Device Configuration: I/O and Interrupt Control provides the user with the information needed to develop custom applications such as the revision of the current BIOS configuration to a user-specific configuration.

Appendix F - Sample C Software provides a library of sample code the programmers may utilize to build the required application software for their system.

References

For the most up-to-date specifications for the VMIVME-7589A, please refer to:

VMIC specification number 800-107589-000

The following books refer to the Tundra Universe II-based interface option available in the VMIVME-7589A:

VMIVME-7589A, Tundra Universe II™-Based VMEbus Interface Option Product Manual
VMIC Doc. No. 500-107589-001

VMEbus Interface Components Manual
Tundra Semiconductor Corporation
603 March Rd.
Kanata, Ontario
Canada, K2K 2M5
(613) 592-0714 FAX (613) 592-1320
www.tundra.com

PCI bus Interface and Clock Distribution Chips
PLX Technology, Inc.
625 Clyde Avenue
Mountain View, CA 94043
(415) 694-2800 (415) 960-0479
www.plxtech.com

Some reference sources helpful in using or programming the VMIVME-7589A include:

Pentium and Pentium Pro Processors and Related Products
Intel Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641
(800) 548-4752
www.intel.com

Pentium Processor Family Developer's Manual
Order Number 241428
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080
www.intel.com

Pentium Processor at 150 MHz, 166 MHz, and 200 MHz

February 1996, Order Number 242769-003
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080
www.intel.com

AMD-K6 Data Sheet

One AMD Place
P.O. Box 3453
Sunnyvale, CA 94088-3453
(800) 538-8450
www.amd.com

Award BIOS

Award Software International, Inc.
777 East Middle Field Road
Mountain View, CA 94043-4023
(650) 237-6800 FAX: (650) 968-0274 BBS: (650) 968-0249
www.award.com

Intel 430TX PCIset

82441FX PCI and Memory Controller (PMC)
82442FX Data Bus Accelerator (DBX)
May 1996, Order Number 290549-001
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080
www.intel.com

PCI Special Interest Group

P.O. Box 14070
Portland, OR 97214
(800) 433-5177 (U.S.) (503) 797-4207 (International)
FAX (503) 234-6762
www.pcisig.com

The VMEbus interrupt and control software library references included for Windows NT:

VMISFT-9420 IOWorks Access User's Guide

Doc. No. 520-009420-910
VMIC
12090 South Memorial Parkway
Huntsville, AL 35803-3308
(800) 322-3616 FAX: (256) 882-0859
www.vmic.com

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

VMEbus Specification Rev. C. and The VMEbus Handbook

VMEbus International Trade Association (VITA)
7825 East Gelding Drive
Suite No. 104
Scottsdale, AZ 85260
(602) 951-8866 FAX: (602) 951-0720
www.vita.com

Safety Summary

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

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

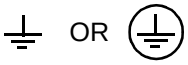
Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

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

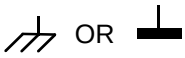
Safety Symbols Used in This Manual

Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 V are so marked).



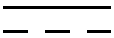
Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.

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).

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

The WARNING sign denotes a hazard. It calls attention to a procedure, a practice, a condition, 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, or a condition, 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.

Notation and Terminology

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

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

Segment:Offset to Linear Address

Linear Address = (Segment × 16) + Offset

Linear Address to Segment:Offset

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

Offset = *remainder* × 65536

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

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

CHAPTER 1

VMIVME-7589A Features and Options

Contents

Introduction

The VMIVME-7589A performs all the functions of a standard IBM PC / AT motherboard with the following features:

- Single-slot VMEbus 6U size
- High-performance AMD-K6 or Intel Pentium processor
- Low-power split voltage-based design
- Up to 128 Mbyte of Synchronous DRAM
- 64-bit PCI SVGA video graphics accelerator
  - 2 Mbyte SGRAM Video Memory
  - Resolutions up to 1280 x 1024 x 256 colors, 1500 x 1200 x 256 colors, 1024 x 768 x 64 K colors, or 800 x 600 x 16 M colors
- Battery-backed clock / calendar
- Front panel reset switch and miniature speaker
- On-board ports for keyboard and mouse, Ultra-IDE hard drive, floppy drive, Ethernet, video, serial, and parallel I / O
- Front panel “vital sign” indicators (power, Ultra-IDE hard drive activity, VMEbus SYSFAIL, and Ethernet status)
- Three general-purpose programmable 16-bit timers
- Software-controlled watchdog timer
- 16 Mbyte of bootable flash on secondary IDE
- 32 Kbyte of battery-backed SRAM

The VMIVME-7589A supports standard PC/AT I/O features such as those listed in Table 1-1. Figure 1-1 on page 25 shows a block diagram of the VMIVME-7589A emphasizing the I/O features, including the optional PCI-to-VMEbus bridge option.

Table 1-1 PC/AT I/O Features

I/O FEATURE	IDENTIFIER	PHYSICAL ACCESS
Two Serial Ports (16550-Compatible RS-232C)	COM1, COM2	Front Panel, Micro-D 9-Pin
One Enhanced Parallel Port, Supports ECP/EPP Modes	Parallel	Front Panel Micro-D 25-Pin
AT-Style Keyboard Controller with PS/2-Style Adapter	KYB	Front Panel PS/2-Style Connector, Mini-DIN Circular (female)
PS/2 Mouse	MSE	Front Panel PS/2-Style Connector, Mini-DIN Circular (female)
Real-Time Clock/Calendar with Battery and Watchdog Timer	Date, Time	
Super VGA Video Controller with 2 Mbyte DRAM	SVGA	Front Panel DB15HD High Density (female)
Ethernet, 10BaseT, 100BaseTx, Novell NE-2000 Compatible	LAN	Front Panel RJ45
Floppy Disk Controller (two drives maximum)	Drives A, B	34-pin Header Flat Cable Type
DMA capable EIDE Fixed Disk Controller (two drives maximum)	Drives C, D	40-pin Header Flat Cable Type
Hardware Reset	RST	Front Panel Push-Button
IBM/PC Sound	Beep	Front Panel Speaker Port
Power Status, Hard Drive Activity, and Ethernet Status	LED Indicators	Front Panel

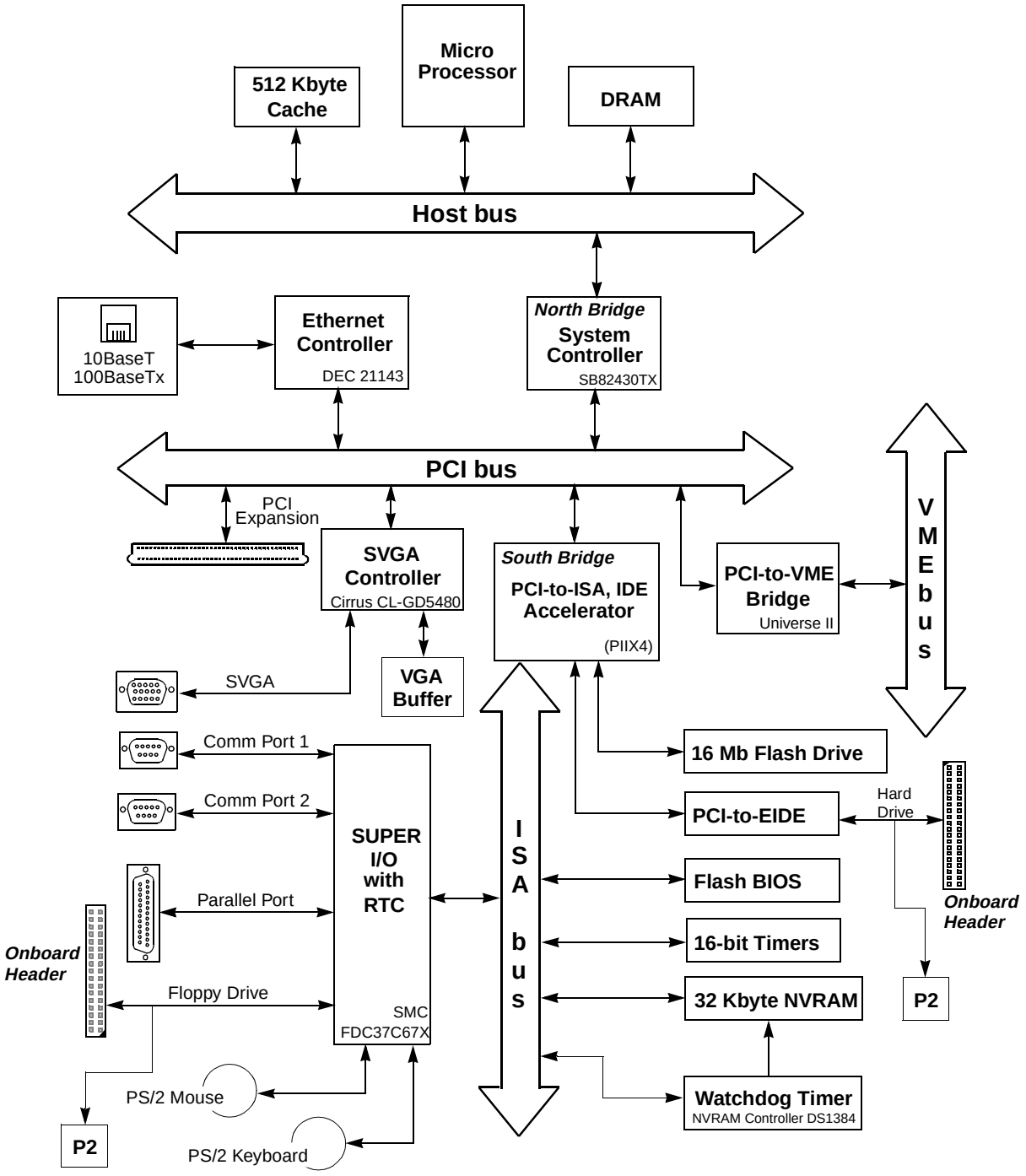


Figure 1-1 VMIVME-7589A Block Diagram

VMEbus Features

In addition to its PC / AT functions, the VMIVME-7589A has the following VMEbus features when purchased with the Universe II-based bridge option:

- Single-slot, 6U height VMEbus board
- Complete six-line Address Modifier (AM-Code) programmability
- VME data interface with separate hardware byte / word swapping for master and slave accesses
- Support for VME64 multiplexed MBLT 64-bit VMEbus block transfers
- User-configured interrupter
- User-configured interrupt handler
- System Controller mode with programmable VMEbus arbiter (PRI, SGL, and RRS modes are supported)
- VMEbus BERR* bus error timer (software programmable)
- Slave access from the VMEbus to local RAM and mailbox registers
- Full-featured programmable VMEbus requester (ROR, RWD, and BCAP modes are supported)
- System Controller autodetection
- Complete VMEbus master access through five separate Protected-mode memory windows

Figure 1-2 illustrates the VMIVME-7589A functions in a typical VMEbus system. The VMIVME-7589A is a versatile single-board solution for VMEbus control with familiar PC / AT operation.

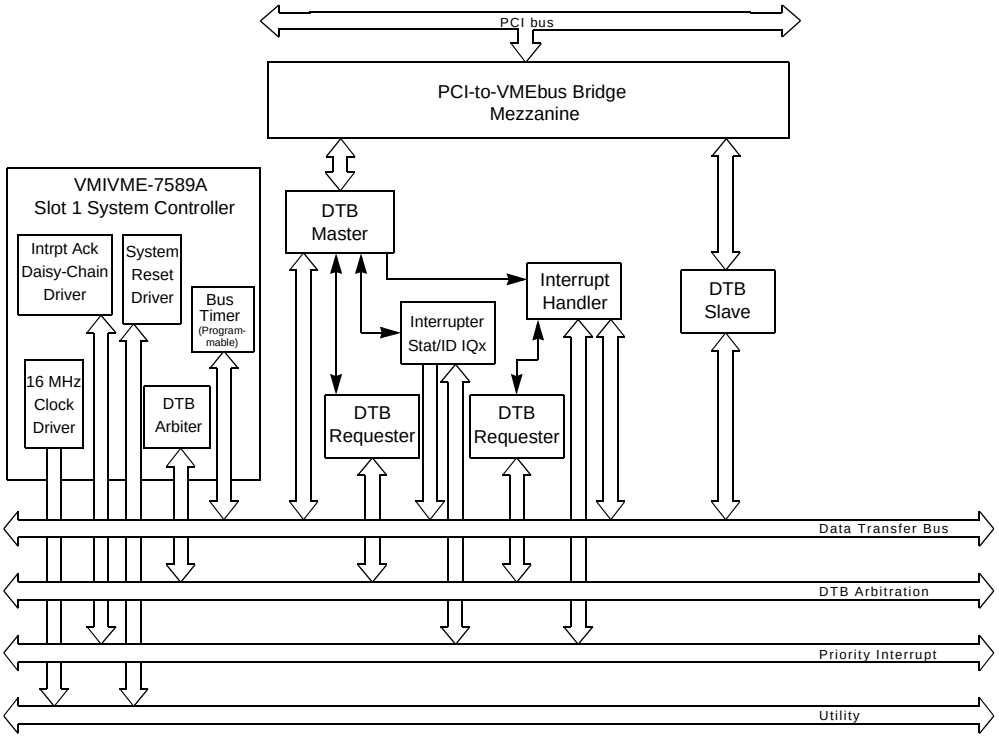


Figure 1-2 VMIVME-7589A VMEbus Functions

VMIVME-7589A Product Options

VMIC's VMIVME-7589A is built around three fundamental hardware configurations. These involve processor performance, RAM memory, and the user interface. *These options are subject to change based on emerging technologies and availability of vendor configurations.*

The options and current details available with the VMIVME-7589A are defined in the device specification sheet available from your VMIC representative.

The VMIVME-7589A VMEbus interface is provided by the PCI-to-VMEbus bridge mezzanine built around the Tundra Semiconductor Corporation Universe II VMEbus interface chip. The Universe II provides a reliable high-performance 64-bit VMEbus-to-PCI interface in one design. The functions and programming of the Universe II-based VMEbus are addressed in detail in a separate associated manual to this product. The manual titled: The *VMIVME-7589A Tundra Universe II Based VMEbus Interface Option Product Manual* is provided with the purchase of the PCI-to-VMEbus bridge option.

CHAPTER 2

Installation and Setup

Contents

Introduction

This chapter describes the hardware jumper settings, connector definitions, installation, system setup, and operation of the VMIVME-7589A. The PCI-to-VMEbus bridge option and the Tundra Universe II-based interface are also included.

Unpacking Procedures

Upon receipt of the VMIVME-7589A, observe any errata sheets or precautions found in the shipping container. All items should be carefully unpacked and thoroughly inspected for damage that might have occurred during shipment.

All claims arising from shipping damage should be filed with the carrier. Contact VMIC Customer Support for a RMA number prior to returning the damaged item(s).

Note

VMIC Customer Service is available at: 1 (256) 880-0444 or toll free 1 (800) 240-7782. Or E-mail us at customer.service@vmic.com

Hardware Setup

The VMIVME-7589A is factory populated with user-specified options as part of the VMIVME-7589A ordering information. The CPU speed and RAM size are not user-upgradable. To change CPU speeds or RAM size, contact customer service to receive a Return Material Authorization (RMA).

VMIC Customer Service is available at: 1-800-240-7782.

The VMIVME-7589A is tested for system operation and shipped with factory-installed header jumpers. The physical location of the jumpers and connectors for the single board CPU are illustrated in Figure 2-1 on page 31. The definitions of the CPU board jumpers and connectors are included in Table 2-1 through Table 2-10. The Tundra Universe II-based PCI-to-VMEbus Bridge jumper configuration is discussed in a following section.

All jumpers are factory configured and should not be modified by the user. There are four exceptions: the Programmable Timer Clock Select (J36), the IDE Routing Option (J12), the Password Clear (J33), and the Watchdog Timer (J30).

Modifying any other jumper will void the warranty and may damage the unit. The default jumper condition of the VMIVME-7589A is expressed in Table 2-1 through Table 2-7 with **bold text** in the table cells.

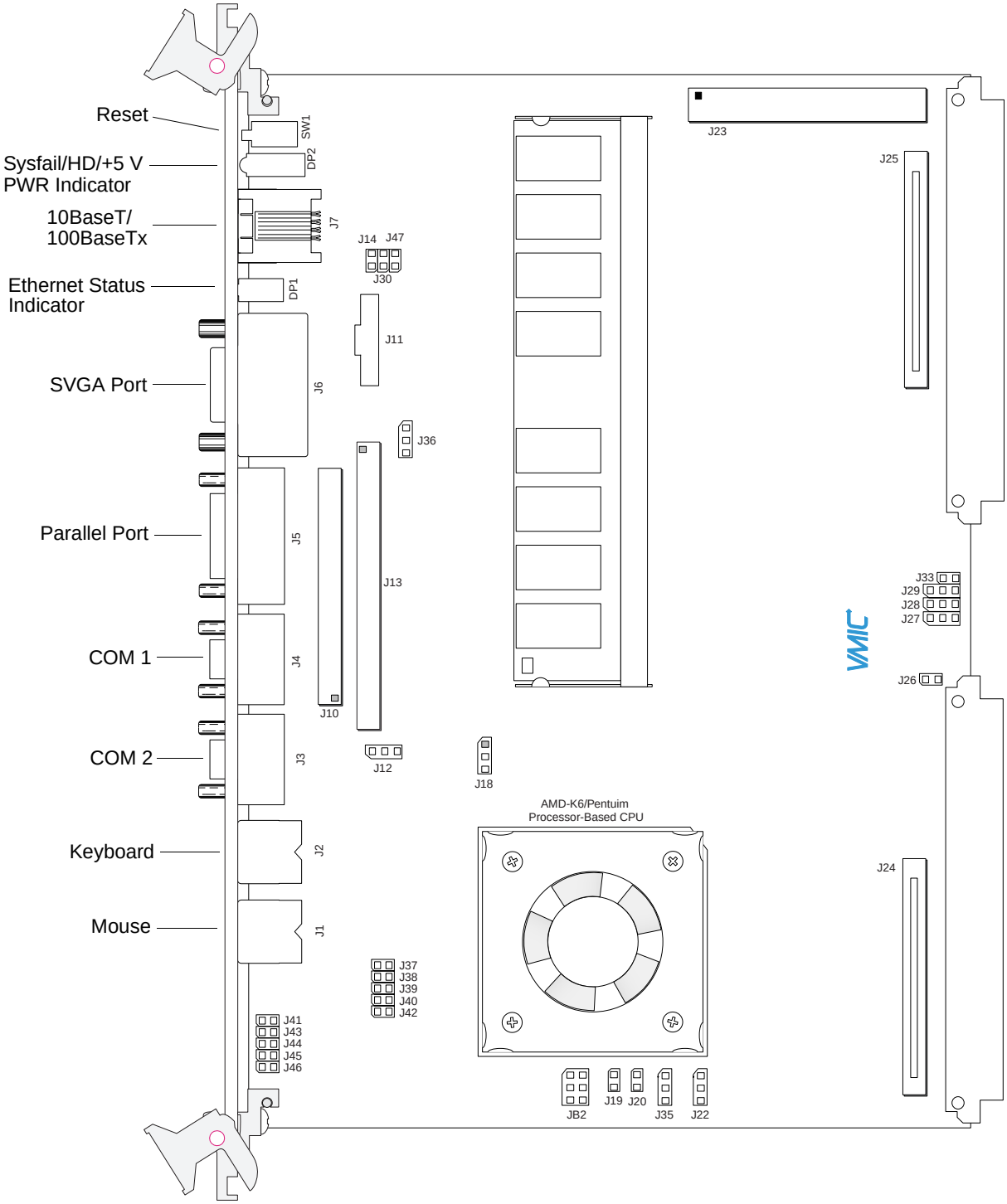


Figure 2-1 VMIVME-7589A CPU Board, I/O Port, and Jumper Locations

Table 2-1 CPU Board Connectors

Connector	Function
J3	COM2
J4	COM1
J5	Parallel Port Connector
J6	Video Connector
J7	Ethernet Connector
J8	PS/2 Keyboard Connector
J9	Mouse Connector
J10	Floppy Interface Connector
J11	Port 80 Test Interface
J13	EIDE Connector
J18	Fan Connector
J23	PCI Expansion Connector
J24, J25	PCI/VME Connectors
P1, P2	VMEbus and Floppy/EIDE Connectors

Table 2-2 Ultra IDE Routing Option (User Configurable) - Jumper (J12)

Select	Jumper Position
Ultra IDE Onboard Header	***1-2***
P2	2-3

Note

The default jumper position of 1-2 on J12 enables the utilization of the onboard headers J10 for the floppy drive interface and J13 for an IDE harddrive interface. If the user is configuring the VMIVME-7589A for use with a VMIVME-7452 with the P2 option, J12 should be placed in the 2-3 position.

Table 2-3 BIOS Mode Option - Jumper (J14)

Select	Jumper Position
Boot Block Programming	In
No Program	***Out***

Table 2-4 Fan Connector Option - Jumper (J18)

Select	Jumper Position
+5 V	***1-2***
+12 V	2-3

Table 2-5 CPU Selection - Jumpers (J19, J20)

Jumper	M1	*Others*
J19	Out	***In***
J20	Out	***In***

Table 2-6 430TX Clock Select - Jumper (J22)

CPU Clock	Jumper Position
66 MHz	1-2
60 MHz	**2-3**

Table 2-7 PCI/VME Mezzanine Option - Jumper (J26)

Select	Jumper Position
Mezzanine Installed	Out
Mezzanine Not Installed	In

Table 2-8 CPU Speed Jumper (J27, J28, J29)

J27	J28	J29	CPU	PCI	48 MHz pin	24 MHz pin
2-3	2-3	2-3	60 MHz	30 MHz	48 MHz	24 MHz
2-3	2-3	1-2	66.6 MHz	33.3 MHz	48 MHz	24 MHz
2-3	1-2	2-3	50 MHz	25 MHz	48 MHz	24 MHz
2-3	1-2	1-2	55 MHz	27.5 MHz	48 MHz	24 MHz
1-2	2-3	2-3	Reserved	Reserved	Reserved	Reserved
1-2	2-3	1-2	83.3 MHz	33.3 MHz	48 MHz	24 MHz
1-2	1-2	2-3	REF/2	REF/4	REF/2	REF/4
1-2	1-2	1-2	Tristate	Tristate	Tristate	Tristate

Table 2-9 Watchdog Timer (User Configurable) - Jumper (J30)

Select	Jumper Position
RTCRESET	In
No RTCRESET	***Out***

Table 2-10 Password Clear (User Configurable) - Jumper (J33)

Select	Jumper Position
Normal	***Out***
Clear NVRAM/CMOS/ Password	In

Note

The VMIVME-7589A's BIOS has the capability (not currently enabled) of password protecting casual access to the unit's CMOS set-up screens. The Password Clear jumper (J33) allows the user to clear the password, as might be necessary to do in the case of a forgotten password.

To clear the CMOS:

1. Turn off power to the unit

2. Install a jumper at J33
3. Power up the unit
4. Turn off the power to the unit and remove the jumper from J33

When power is reapplied to the unit, the CMOS will be cleared.

Table 2-11 CPU Clock/Bus Multiple - Jumper Blocks (JB2 and J35)

Core/Bus Speed Ratio	Intel Based		Other (Non-Intel) Based	
	J35	JB2	J35	JB2
1X	NA	Out	Out	Out
2X	**NA**	**3-5, 2-4**	**Out**	**3-5, 2-4**
5/2X	NA	3-5, 4-6	Out	3-5, 4-6
3X	NA	1-3, 4-6	Out	1-3, 4-6
7/2X	NA	1-3, 2-4	Out	1-3, 2-4
4X	NA	NA	In	3-5, 2-4
9/2X	NA	NA	In	3-5, 4-6
5X	NA	NA	In	1-3, 4-6
11/2X	NA	NA	In	1-3, 2-4

Table 2-12 Programmable Timer Clock Select (User Configurable) - Jumper (J36)

Select	Jumper Position
2 MHz	1-2
1 MHz	***2-3***

Table 2-13 CPU Power Supply Option Jumper (J37, J38, J39, J40, J42) (J41, J43, J44, J45, J46)

Select	J37	J38	J39	J40	J42
VIO=3.3V, VCORE=3.3V	Out	In	In	Out	In
VIO=3.3V, VCORE=2.8V	Out	In	Out	Out	In
	J41	**J43**	**J44**	**J45**	**J46**
VIO=3.3V, VCORE=2.2V	**Out**	**Out**	**Out**	**In**	**Out**
VIO=3.3V, VCORE=1.8V	In	In	Out	In	Out

Table 2-14 Battery Jumper (J47)

Select	Jumper Position
SRAM Battery Backed	In
SRAM Not Battery Backed	Out

VMIVME-7589A, Tundra Universe II™-Based PCI-to-VMEbus Bridge

The optional Tundra Universe II-based PCI-to-VMEbus bridge is tested for system operation and shipped with factory-installed header jumpers. The bridge is shipped with the unit as part of the VMIVME-7589A. Figure 2-2 illustrates the physical location of the user-configurable jumpers and connectors on the bridge option. Table 2-15 lists each jumper designator, its function, and the factory-installed default configuration.

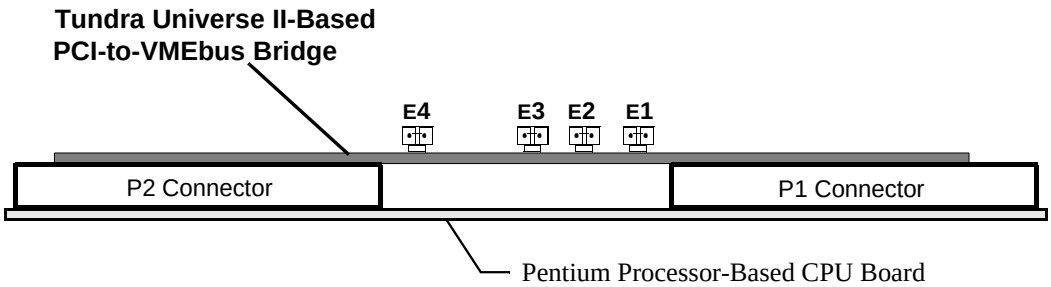


Figure 2-2 Jumper Locations on the Universe II-Based PCI-to-VMEbus Bridge (Rear View)

Table 2-15 Universe II-Based PCI-to-VMEbus Bridge Jumper Functions and Factory Settings

JUMPER	FUNCTION	FACTORY SETTING
E1	Installed - Universe II memory-mapped Removed - Universe II I/O-mapped	Installed **(Should Not be Removed)**
E2	Installed - Receives VMEbus SYSRESET Removed - Does Not Receive	Installed
E3	Installed - Drives VMEbus SYSRESET Removed - Does Not Drive	Installed
E4	Installed - SYSFAIL Not asserted upon reset Removed - SYSFAIL Asserted upon reset	Installed

Note

Any other jumper locations are reserved for VMIC use only and should not be altered from the factory default settings.

Installation

The VMIVME-7589A conforms to the VMEbus physical specification for a 6U dual Eurocard (dual height). It can be plugged directly into any standard chassis accepting this type of board.

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

The following steps describe the VMIC recommended method for VMIVME-7589A installation and powerup:

1. Make sure power to the equipment is off.
2. If a drive module or other expansion module such as VMIC's VMIVME-7452, VMIVME-7453, or VMEbus Floppy / Hard Disk Module is to be used, connect it to the controller prior to board installation. Refer to the disk module manual.
3. Choose chassis slot. The VMIVME-7589A ***must*** be attached to a dual P1 / P2 VMEbus backplane.

   If the VMIVME-7589A with the Universe II-based PCI-to-VMEbus bridge is to be the VMEbus system controller, choose the first VMEbus slot. If some other board is the VMEbus system controller, choose any slot ***except*** slot one. The Universe II-based bridge requires no jumpers for enabling / disabling the system controller function.

The VMIVME-7589A requires forced air cooling. It is advisable to install blank panels over any exposed VMEbus slots. This will allow for better air flow over the VMIVME-7589A board. For 20-slot VME configurations, three 100 CFM fans are recommended.

4. Insert the VMIVME-7589A and its attached expansion modules into the chosen VMEbus chassis slot (expansion modules should fill the slots immediately adjacent to the VMIVME-7589A). 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 labeled on the front panel, and detailed pinouts are in Appendix A. Minimally, a keyboard and a monitor are required if the user has not previously configured the system.
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-7589A.

7. The VMIVME-7589A features a Flash Disk resident on the board. Refer to Chapter 4 for set up details.
8. If an external drive module is installed, the BIOS Setup program must be run to configure the drive types. See Appendix C to properly configure the system.
9. If a drive module is present, install the operating system according to the manufacturer's instructions.

See Appendix B for instructions on installing VMIVME-7589A peripheral driver software during operating system installation.

BIOS Setup

The VMIVME-7589A 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-7589A is shipped from the factory with no hard drives configured in CMOS. The BIOS Setup program must be run to configure the specific drives attached.

Details of the VMIVME-7589A BIOS setup program are included in Appendix C.

Front Panel Connectors

The front panel connections, including connector pinouts and orientation, for the VMIVME-7589A are defined in detail in Appendix A.

PCI Expansion Site Connector

The VMIVME-7589A supplies a PCI expansion site connector for adding a VMIVME-7434 PMC/PC Card expansion board. This expansion capability allows third-party devices to be used with the VMIVME-7589A, as shown in Figure 2-3.

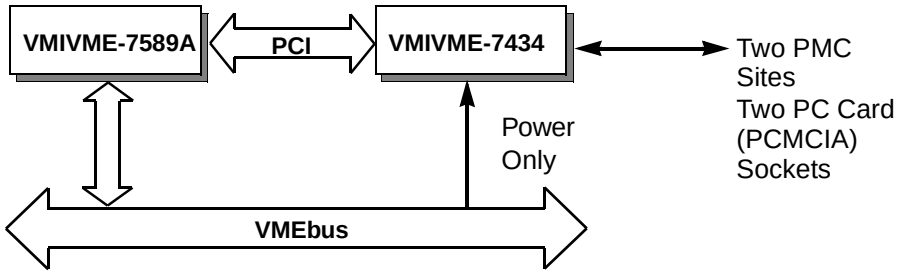


Figure 2-3 PCI Expansion Site

LED Definition

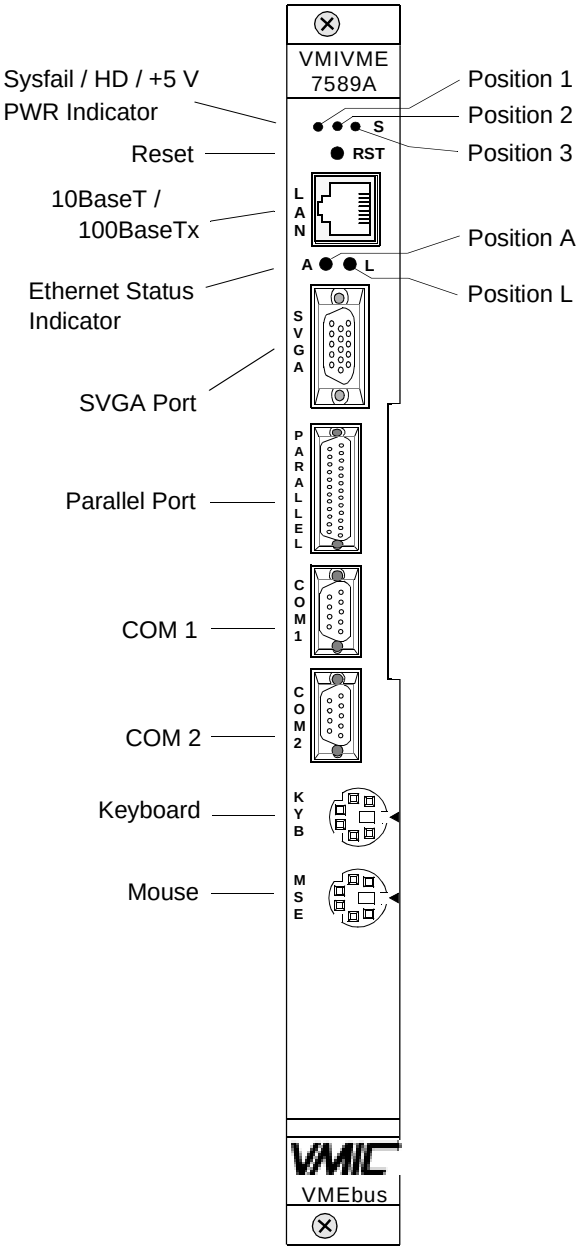


Figure 2-4 LED Position on the Front Panel

Position 1 *Power* - Indicates when power is applied to the board.

Position 2 *Hard Drive Indicator* - Indicates when hard drive activity is occurring.

Position 3 *SYSFAIL* - Indicates when a VMEbus SYSFAIL is asserted.

Position A *Ethernet Active* - Indicates when the Ethernet is transmitting or receiving data.

Position L *Ethernet Link* - *Yellow* indicates when the Ethernet is linked in 10BaseT mode. *Green* indicates when the Ethernet is linked in 100BaseTx mode.

CHAPTER 3

PC/AT Functions

Contents

Introduction

The VMIVME-7589A provides a complete IBM PC / AT-compatible AMD-K6 or Intel Pentium processor-based computer. The design includes a high-speed microprocessor with current technology memory. Reference the VMIC product specifications for available component options.

Because the design is PC / AT compatible, it retains standard PC memory and I / O maps along with standard interrupt architecture. Furthermore, the VMIVME-7589A includes a PCI-compatible video adapter and Ethernet controller.

The following sections describe in detail the PC / AT functions of the VMIVME-7589A.

CPU Socket

The VMIVME-7589A CPU socket is factory populated with a current, high-speed AMD-K6 or Pentium processor. The CPU speed and RAM size are user-specified as part of the VMIVME-7589A ordering information. The CPU speed is not user-upgradable.

To change CPU speeds or RAM size, contact customer service to receive a Return Material Authorization (RMA).

VMIC Customer Service is available at: 1-800-240-7782.

Physical Memory

The VMIVME-7589A provides Synchronous DRAM (SDRAM) as on-board system memory. Memory can be accessed as bytes, words, or longwords.

All RAM on the VMIVME-7589A is dual-ported to the VMEbus through the PCI-to-VME bridge. The memory is addressable by the local processor, as well as the VMEbus slave interface by another VMEbus master. Caution must be used when sharing memory between the local processor and the VMEbus to prevent a VMEbus master from overwriting the local processor's operating system.

When using the Configure utility of I/O Works Access with Windows NT 4.0 to configure RAM, do not request more than 25 percent of the physical RAM. Exceeding the 25 percent limit may result in a known Windows NT bug causing unpredictable behavior during the Windows NT boot sequence and require the use of an emergency repair disk to restore the computer. The bug is present in Windows NT 4.0 service pack level 3. It is recommended that an emergency repair disk be kept up-to-date and easily accessible.

The VMIVME-7589A includes both the system and video BIOS in a single 256 K x 8 FLASH memory device.

The VMIVME-7589A memory includes 32 Kbyte of battery-backed SRAM addressed at $D800E to $DFFF. All but the first 14 bytes are accessible. Bytes $D8000 - $D800D are used by the Watchdog Timer controller. The battery-backed SRAM can be accessed by the CPU at anytime, and can be used to store system data that must not be lost during power-off conditions.

Memory and Port Maps

Memory Map - Tundra Universe II-Based PCI-to-VMEbus Bridge

The memory map for the Tundra Universe II-based interface VMIVME-7589A is shown in Table 3-1. All systems share this same memory map, although a VMIVME-7589A with less than the full 128 Mbyte of DRAM does not fill the entire space reserved for On-Board Extended Memory.

Table 3-1 VMIVME-7589A, Universe II-Based Interface Memory Address Map

MODE	MEMORY ADDRESS RANGE	SIZE	DESCRIPTION
PROTECTED MODE	$FFFF 0000 - $FFFF FFFF	64 Kbyte	ROM BIOS Image
	$0800 0000 - $FFFE FFFF	3.875 Gbyte	Unused *
	$0010 0000 - $07FF FFFF	127 Mbyte	Reserved for ** On-Board Extended Memory (not filled on all systems)
REAL MODE	$E0000 - $FFFFF	128 Kbyte	ROM BIOS
	$D8000 - $D800D	14 bytes	RTC / Watchdog Timer Control Registers
	$D800E - $DFFFF	32 Kbyte	Battery-Backed SRAM and Watchdog Timer
	$D0000 - $D7FFF	32 Kbyte	Unused
	$C8000 - $CFFFF	32 Kbyte	Reserved
	$C0000 - $C7FFF	32 Kbyte	Video ROM
	$A0000 - $BFFFF	128 Kbyte	Video RAM
	$00000 - $9FFFF	640 Kbyte	User RAM / DOS RAM

* This space can be used to set up protected mode PCI-to-VMEbus windows (also referred to as PCI slave images).
** This space can be allocated as shared memory (for example, between the AMD-K6 / Pentium processor-based CPU and VMEbus Master. Note, that if a PMC board is loaded, the expansion BIOS may be placed in this area.

I/O Port Map

The AMD-K6/Pentium processor-based CPU includes special input/output instructions that access I/O peripherals residing in I/O addressing space (separate and distinct from memory addressing space). Locations in I/O address space are referred to as *ports*. When the CPU decodes and executes an I/O instruction, it produces a 16-bit I/O address on lines A00 to A15 and identifies the I/O cycle with the M/IO control line. Thus, the CPU includes an independent 64Kbyte I/O address space which is accessible as bytes, words, or longwords.

Standard PC/AT hardware circuitry reserves only 1,024 byte of I/O addressing space from I/O $000 to $3FF for peripherals. All standard PC I/O peripherals such as serial and parallel ports, hard and floppy drive controllers, video system, real-time clock, system timers, and interrupt controllers are addressed in this region of I/O space. The BIOS initializes and configures all these registers properly; adjusting these I/O ports directly is not normally necessary.

The assigned and user-available I/O addresses are summarized in the I/O Address Map, Table 3-2.

Table 3-2 VMIVME-7589A I/O Address Map

I/O ADDRESS RANGE	SIZE IN BYTES	HW DEVICE	PC/AT FUNCTION
$000 - $00F	16		DMA Controller 1 (Intel 8237A Compatible)
$010 - $01F	16		Reserved
$020 - $021	2		Master Interrupt Controller (Intel 8259A Compatible)
$022 - $03F	30		Reserved
$040 - $043	4		Programmable Timer (Intel 8254 Compatible)
$044 - $05F	30		Reserved
$060 - $064	5		Keyboard, Speaker, Eqpt. Configuration (Intel 8042 Compatible)
$065 - $06F	11		Reserved
$070 - $071	2		Real-Time Clock, NMI Mask
$072 - $07F	14		Reserved
$080 - $08F	16		DMA Page Registers
$090 - $091	2		Reserved
$092	1		Alt. Gate A20/Fast Reset Register
$093	1		Reserved

Table 3-2 VMIVME-7589A I/O Address Map (Continued)

I/O ADDRESS RANGE	SIZE IN BYTES	HW DEVICE	PC/AT FUNCTION
$094	1	Super VGA Chip	POS102 Access Control Register
$095 - $09F	11		Reserved
$0A0 - $0A1	2		Slave Interrupt Controller (Intel 8259A Compatible)
$0A2 - $0BF	30		Reserved
$0C0 - $0DF	32		DMA Controller 2 (Intel 8237A Compatible)
$0E0 - $16F	142		Reserved
$170 - $177	8	PIIX4	Secondary Hard Disk Controller
$178 - $1EF	120		User I/O
$1F0 - $1F7	8	PIIX4	Primary Hard Disk Controller
$1F8 - $277	128		User I/O
$278 - $27F	8	I/O Chip*	LPT2 Parallel I/O*
$280 - $2E7	104		Reserved
$2E8 - $2EE	7	UART*	COM4 Serial I/O*
$2EF - $2F7	9		User I/O
$2F8 - $2FE	7	Super-I/O Chip	COM2 Serial I/O (16550 Compatible)
$2FF - $36F	113		Reserved
$370 - $377	8	Super-I/O Chip	Secondary Floppy Disk Controller
$378 - $37F	8	Super-I/O Chip	LPT1 Parallel I/O
$380 - $3E7	108		Reserved
$3E8 - $3EE	7	UART*	COM3 Serial I/O*
$3F0 - $3F7	8	Super-I/O Chip	Primary Floppy Disk Controller
$3F8 - $3FE	7	Super-I/O Chip	COM1 Serial I/O (16550 Compatible)
$3FF - $4FF			Reserved
$500 - $503	4	82C54 Timer	Programmable Internal Timer
$504 - $CFF			Reserved

* While these I/O ports are reserved for the listed functions, they are not implemented on the VMIVME-7589A. They are listed here to make the user aware of the standard PC/AT usage of these ports.

PC/AT Interrupts

In addition to an I/O port address, an I/O device 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 Non-Maskable Interrupt (NMI) are listed in Table 3-3 along with their functions. Table 3-4 on page 47 details the vectors in the interrupt vector table. The interrupt number in HEX and decimal are also defined for real and protected mode in Table 3-4.

The interrupt hardware implementation on the VMIVME-7589A 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. This architecture is represented in Figure 3-1 on page 52.

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).

Table 3-3 PC/AT Hardware Interrupt Line Assignments

IRQ	AT FUNCTION	COMMENTS
NMI	Parity Errors (Must be enabled in BIOS Setup)	Used by VMIVME-7589A VMEbus Interface
0	System Timer	Set by BIOS Setup
1	Keyboard	Set by BIOS Setup
2	Duplexed to IRQ9	
3	COM2/COM4	
4	COM1/COM3	
5	Timer	Assigned to On-Board Timer
6	Floppy Controller	
7	LPT1	
8	Real-Time Clock	
9	Old IRQ2	Set by BIOS Setup

Table 3-3 PC/AT Hardware Interrupt Line Assignments (Continued)

IRQ	AT FUNCTION	COMMENTS
10	Not Assigned	Determined by BIOS
11	Not Assigned	Determined by BIOS
12	Mouse	
13	Math Co-processor	
14	AT Hard Drive	
15	Flash Drive	

Table 3-4 PC/AT Interrupt Vector Table

INTERRUPT NO.		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	Co-processor 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	Timer	Same as Real Mode
0E	14	IRQ6	Floppy Disk Controller	Page Fault
0F	15	IRQ7	LPT1 Parallel I/O	Unassigned

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

INTERRUPT NO.		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
10	16		BIOS Video I/O	Co-processor 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
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

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

INTERRUPT NO.		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
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
67-71	103-113		Reserved by DOS	Same as Real Mode
72	114	IRQ10	Not Assigned	
73	115	IRQ11	Not Assigned	
74	116	IRQ12	Mouse	
75	117	IRQ13	Math Co-processor	
76	118	IRQ14	AT Hard Drive	
77	119	IRQ15	Flash Drive	
78-7F	120-127		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

PCI Interrupts

Interrupts on Peripheral Component Interconnect (PCI) Local Bus are optional and defined as “level sensitive,” asserted low (negative true), using open drain output drivers. The assertion and de-assertion of an interrupt line, INTx#, is asynchronous to CLK. A device asserts its INTx# line when requesting attention from its device driver. Once the INTx# signal is asserted, it remains asserted until the device driver clears the pending request. When the request is cleared, the device de-asserts its INTx# signal.

PCI defines one interrupt line for a single function device and up to four interrupt lines for a multifunction device or connector. For a single function device, only INTA# may be used while the other three interrupt lines have no meaning. Figure 3-1 on page 52 depicts the VMIVME-7589A interrupt logic pertaining to VMEbus operations and the PCI expansion site.

Any function on a multifunction device can be connected to any of the INTx# lines. The Interrupt Pin register defines which INTx# line the function uses to request an interrupt. If a device implements a single INTx# line, it is called INTA#; if it implements two lines, they are called INTA# and INTB#; and so forth. For a multifunction device, all functions may use the same INTx# line or each may have its own (up to a maximum of four functions) or any combination thereof. A single function can never generate an interrupt request on more than one INTx# line.

The slave PIC accepts the VMEbus interrupts through lines that are defined by the BIOS. The BIOS defines which interrupt line to utilize depending on which system requires the use of the line.

The PCI-to-VME Bridge has the capability of generating a Non-Maskable Interrupt (NMI) via the PCI SERR# line. Table 3-5 describes the register bits that are used by the NMI. The SERR interrupt is routed through certain logic back to the NMI input line on the CPU. The CPU reads the NMI Status Control register to determine the NMI source (bits set to 1). After the NMI interrupt routine processes the interrupt, software clears the NMI status bits by setting the corresponding enable / disable bit to 1. The NMI Enable and Real-Time Clock register can mask the NMI signal and disable / enable all NMI sources.

Table 3-5 NMI Register Bit Descriptions

Status Control Register (I/O Address $061, Read/Write, Read Only)	
Bit 7	SERR# NMI Source Status (Read Only) - This bit is set to 1 if a system board agent detects a system board error. It then asserts the PCI SERR# line. To reset the interrupt, set bit 2 to 0 and then set it to 1. When writing to port $061, bit 7 must be 0.
Bit 2	PCI SERR# Enable (Read / Write) - 1 = Clear and Disable, 0 = Enable
Enable and Real-Time Clock Address Register (I/O Address $070, Write Only)	
Bit 7	NMI Enable - 1 = Disable, 0 = Enable

I/O Ports

The VMIVME-7589A incorporates the SMC Super-I/O chip. The SMC chip provides the VMIVME-7589A with a standard floppy drive controller, two 16550 UART-compatible serial ports, and one standard Centronics parallel port. The Ultra-IDE hard drive interface is provided by the Intel 82371SB (PIIX4) PCI ISA IDE Xcelerator chip. All ports are present in their standard PC/AT locations, using default interrupts.

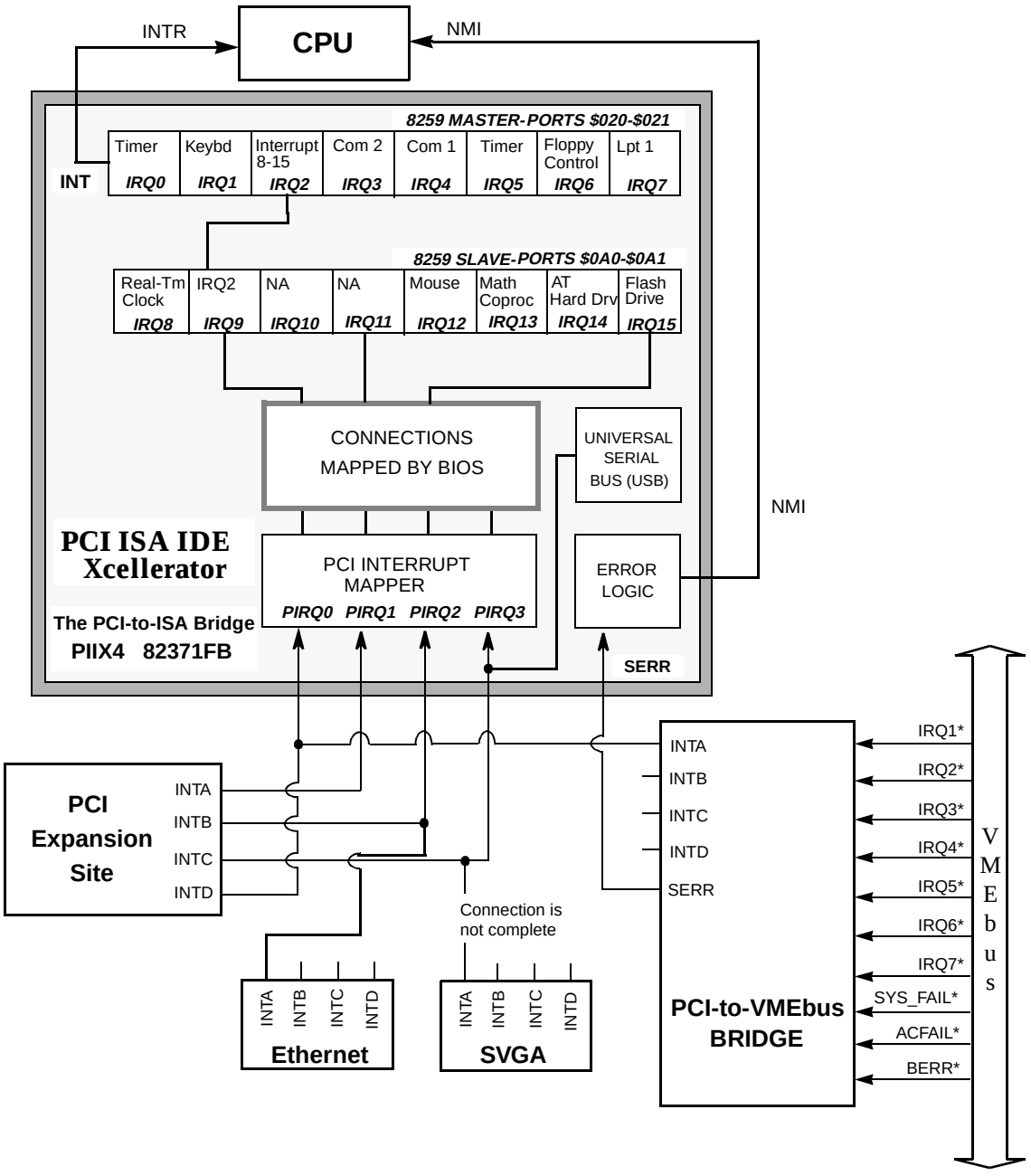


Figure 3-1 Connections for the PC Interrupt Logic Controller

Video Graphics Adapter

The monitor port on the VMIVME-7589A is controlled by a Cirrus Logic CL-GD 5480 chip with 2 Mbyte video DRAM. The video controller chip is hardware and BIOS compatible with the IBM EGA and SVGA standards and also supports VESA high-resolution and extended video modes. Table 3-6 shows the graphics video modes supported by the Cirrus Logic video chip.

Table 3-6 Supported Graphics Video Resolutions

SCREEN RESOLUTION	MAXIMUM COLORS	REFRESH RATES (Hz)
640 x 480	16 M	100
800 x 600	16 M	100
1,024 x 768 (Interlaced)	64 K	100
1,152 x 864	64 K	100
1,280 x 1,024	256	85
1,500 x 1,200	256	70

Not all SVGA 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 capability.

The VMIVME-7589A's processor includes a 32-bit access to video memory with no-wait states. Video I/O registers are accessed using PCI bus. In addition, the Cirrus Logic video controller supports up to a 64 x 64 pixel hardware cursor.

Floppy disks supplied with the VMIVME-7589A also contain video drivers for Windows and other popular programs and operating systems. Appendix B contains instructions on the incorporation of the video drivers during operating system installation.

Ethernet Controller

The network capability is provided by the Digital Semiconductor's 21143. This Ethernet controller is PCI bus based and is software configurable. The VMIVME-7589A supports 10BaseT and 100BaseTx Ethernet.

10BaseT

A network based on the 10BaseT standard uses unshielded twisted-pair cables, providing an economical solution to networking by allowing the use of existing telephone wiring and connectors. The RJ-45 connector is used with the 10BaseT standard. 10BaseT has a maximum length of 100 m from the wiring hub to the terminal node.

100BaseTx

The VMIVME-7589A also supports the 100BaseTx Ethernet. A network based on a 100BaseTx standard uses unshielded twisted-pair cables and a RJ-45 connector. The 100BaseTx has a maximum deployment length of 250 m.

CHAPTER

4

Embedded PC/RTOS Features

Contents

Introduction

VMIC's VMIVME-7589A features additional capabilities beyond those of a typical IBM PC / AT-compatible CPU. The unit provides three software-controlled, general-purpose timers along with a programmable Watchdog timer for synchronizing and controlling multiple events in embedded applications. The VMIVME-7589A also provides a bootable Flash Disk system, and 32 K byte of nonvolatile, battery-backed SRAM. These features make the unit ideal for embedded applications, particularly applications where standard hard drives and floppy disk drives cannot be used.

Timers

General

The VMIVME-7589A provides a user-programmable 82C54 internal timer/counter. The 82C54 provides three independent, 16-bit timers each operating at 1 or 2 MHz clock speed determined by the configuration of jumper J36; reference Table 2-12 on page 35. These timers are completely available to the user, and are not dedicated to any PC/AT function. These timers may be used to generate system interrupts.

Events can be timed by either polling the timers or generating a system interrupt via circuity external to the 82C54. The external circuity consists of logic which generates the interrupt and a Timer Interrupt Status register which indicates which of the three Timers generated an interrupt.

The 82C54 timers are mapped at I/O address $500. The interrupt used by the Timers is IRQ5. The Timer Interrupt Status register is available via the Power Management I/O address space. The access to this space is explained in the Timer Interrupt Status section.

Timer Interrupt Status

A single interrupt, IRQ5, is used by all three Timers. A Timer Interrupt Status register is provided in order to determine which Timer(s) initiated an interrupt. The interrupt status register is a general-purpose input register located, external to the 82C54, at offset 31h from the Power Management Base I/O address. The interrupt status register address can be found by first determining the PCI Configuration Base address for Device ID 7113h and Vendor ID 8086h. The Power Management Base I/O address can be found by reading offset 40h from this PCI Configuration address. The Timer Interrupt Status register bits are located at offset 31h from the Power Management Base I/O address, bits 5, 6, and 7 (refer to Figure 4-2).

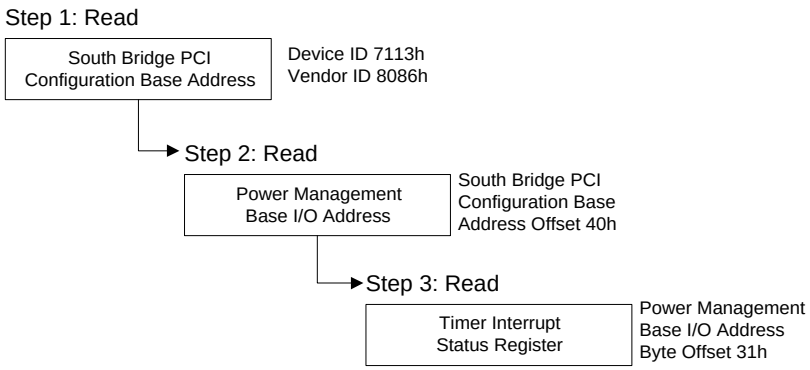


Figure 4-1 Timer Interrupt Status Register Read/Steps

A byte read of Offset 31h from the Power Management Base I/O address is used to obtain these bits. Bits 5, 6, and 7 correspond to Timers 2, 1, and 0, respectively

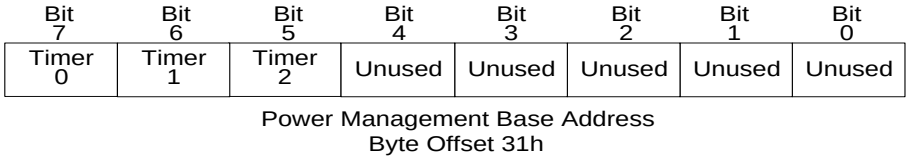


Figure 4-2 Timer Interrupt Status Register

In order to clear the Timer Interrupt Status register, first write zeros (0's) to the general-purpose output register located at offset 37h of the Power Management Base I/O address bits 3, 4, and 6 (Not bits 3, 4 and 5). Then write ones (1's) to these same bits to re-enable the Timer Interrupt Status register. Bits 3, 4, and 6 correspond to Timers 2, 1, and 0, respectively.

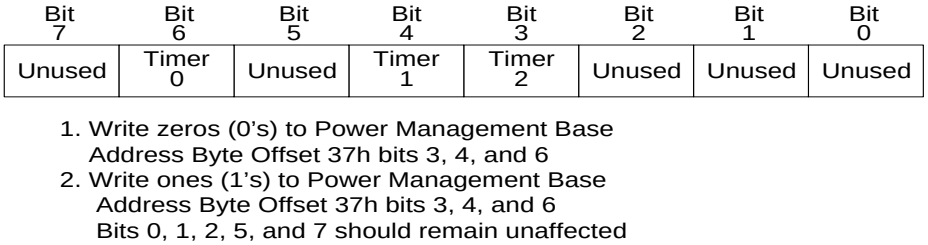


Figure 4-3 Clearing the Timer Interrupt Status Register

Clearing the Interrupt

The Timer Interrupts are cleared using the standard procedure for clearing PC/AT IRQ5. Refer to Appendix D for an example of using the 82C54 timers.

Timer Programming

Architecture

The VMIVME-7589A Timers are mapped in I/O address space starting at $500. See Table 4-1. The Timers, consisting of three 16-bit timers and a Control Word Register (see Figure 4-4) are read from / written to via an 8-bit data bus.

Table 4-1 I/O Address of the Control Word Register and Timers

I/O Address	Select
$500	Timer 0
$501	Timer 1
$502	Timer 2
$503	Control Word Register

Table 4-1 shows the I/O address's of the Control Word Register and Timers.

The Control Word Register is write only. The Timer status information can be obtained from the Read-Back command (see Reading section on page 61).

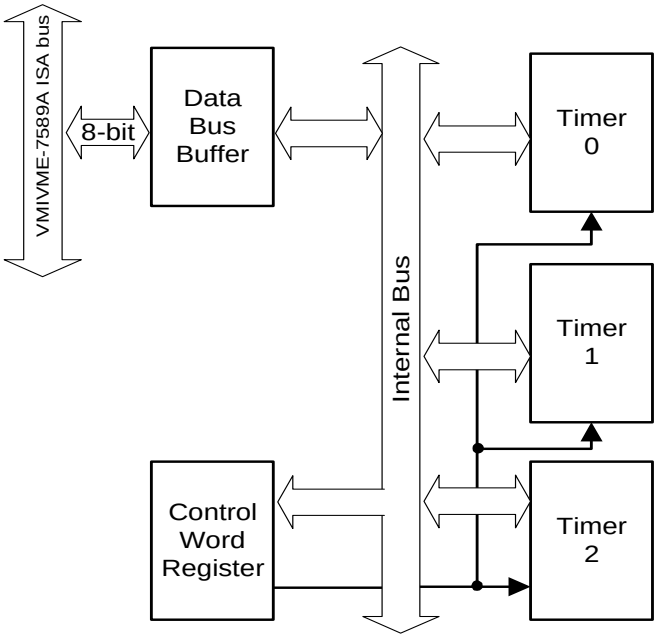


Figure 4-4 82C54 Diagram

The three Timers, Timer 0, 1, and 2, are functionally equivalent. Therefore only a single Timer is described. Figure 4-5 shows a block diagram of the Timers. Each Timer is functionally independent. Although the Control Word Register is shown in the Timer block diagram, it is not a part of the Timer, but its contents directly affect how the Timer functions.

The status register, as shown in Figure 4-5, when latched, contains the present contents of the Control Word Register and the present state of the output and load count flag (The Status Word is available via the Read-Back command, see Reading section on page 61).

The Timer is labelled TE (Timer Element). It is a 16-bit synchronous presettable down counter.

The blocks labelled OL_M and OL_L are 8-bit Output Latches (OL). The subscripts M and L stand for Most Significant byte and Least Significant byte. These latches usually track the TE but when commanded will latch and hold the present count until the CPU reads the count. When the latched count is read, the OL registers will continue to track the TE. When reading the OL registers, two 8-bit accesses must be performed to retrieve the complete 16-bit value of the Timer as only one latch at a time is enabled. The TE cannot be read, the count is read from the OL registers.

There are two 8-bit registers labeled TR_M and TR_L (Timer Register). The subscripts M and L stand for Most Significant byte and Least Significant byte. When a new count is written to the Timer the count is loaded into the TR and later transferred to the TE. The Control logic lets one 8-bit TR register be written to at a time. Two 8-bit writes must be performed to load a complete 16-bit count value. Both TR bytes are transferred to the TE at the same time. The TE cannot be directly written to by the user, the count is written to the TR registers then latched to the TE.

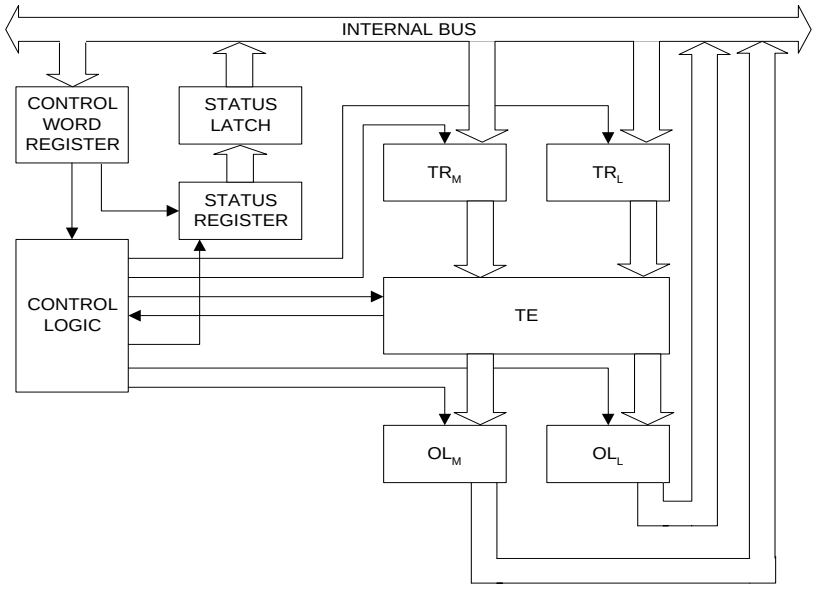


Figure 4-5 Internal Timer Diagram

Writing

The Timers are programmed by first writing a Control Word and then the initial count. The format of the Control Word is shown in Tables 4-2 through 4-6. All Control Words are written into the Control Word Register while the initial counts are written into the individual Timer registers. The format of the initial count is determined by the Control Word.

Table 4-2 Control Word Format

D7	D6	D5	D4	D3	D2	D1	D0
ST1	ST0	RW1	RW0	M2	M1	M0	BCD

Table 4-3 ST - Select Timer

ST1*	ST0*	Description
0	0	Select Timer 0
0	1	Select Timer 1
1	0	Select Timer 2
1	1	Read-Back Command (See Reading section on page 61)

**The ST bits specify which Timer (0, 1, or 2) the Control word refers to or whether this is a Read-Back command*

.

Table 4-4 RW - Read/Write

RW1*	RW0*	Description
0	0	Timer Latch Command (see Reading section)
0	1	Read / Write least significant byte only
1	0	Read / Write most significant byte only
1	1	Read / Write least significant byte first, then most significant

**The RW bits specify whether this is a Timer Latch command or the byte ordering of the Read/Write transaction.*

Table 4-5 M - Mode

M2*	M1*	M0*	Description
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

** Only Mode 2 is supported.*

Table 4-6 BCD

BCD*	Description
0	Binary Timer 16-bits
1	Binary Coded Decimal (BCD) Timer (4 Decades)

** The BCD bit specifies whether the Timer count value is in Binary or BCD.*

When programming the 82C54, only two rules need to be followed.

1. For each Timer, the Control Word must be written first.
2. The initial count must follow the format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte then most significant byte). As long as these rules are adhered to, any programming sequence is acceptable.

Reading

There are two methods for reading the timers: the Timer Latch Command and the Read-Back Command.

Timer Latch Command

The Timer Latch Command allows the reading of a Timer 'on the fly' without affecting the timing in process.

Like a Control Word, the Timer Latch Command is written to the Control Word Register (I/O Address $503, see Table 4-1). The Select Timer bits (ST1, ST0, see Table 4-3) select one of the three timers while the Read/Write bits (RW1, RW0, see Table 4-4) select the Timer Latch Command, RW1 = 0 and RW0 = 0. The selected Timer's count is latched into the 0L registers at the time of the Timer Latch Command. The count is held in the 0L latches until it is read. Multiple Timer Latch Commands can be used to latch more than one Timer. Again, each Timer's count is held latched until it is read.

Read-Back Command

The Read-Back Command allows the user to view the Timer count, the Timer Mode, the current state of the OUT pin, and the Load Flag of the selected Timer. Like a Control Word, the Read-Back Command is written into the Control Word Register and has the format shown in Tables 4-7 and 4-8. The Command applies to the Timer(s) selected by setting the corresponding bits Cnt2, Cnt1, Cnt0 = 1.

Table 4-7 Read-Back Command Format

D7	D6	D5	D4	D3	D2	D1	D0
1	1	$\overline{\text{Count}}$	$\overline{\text{Status}}$	Cnt2	Cnt1	Cnt0	0

Table 4-8 Read-Back Command Description

Bit	Description
D5: $\overline{\text{Count}}$	Latch count of selected Timer(s)
D4: $\overline{\text{Status}}$	Latch status of selected Timer(s)
D3: Cnt2	Select Timer 2
D2: Cnt1	Select Timer 1
D1: Cnt0	Select Timer 0
D0	Reserved, must be 0

The Read-Back Command can be used to latch several Timer counts by setting the $\overline{\text{Count}}$ bit = 0 and selecting the Timers. This is the same as using multiple Timer Latch Commands. Again each Timer's latched count will be held until it is read.

The Read-Back command can also be used to latch the timer status by setting the $\overline{\text{Status}}$ bit = 0 and selecting the Timers. Status of a Timer is accessed by a read from that Timer (see Table 4-1 on page 57). If more than one Timer Status Read-Back command is issued without reading the status, all but the first is ignored.

The format of the Timer Status byte is shown in Tables 4-9 and 4-10.

Table 4-9 Status Byte

D7	D6	D5	D4	D3	D2	D1	D0
OUT	LOAD	RW1	RW0	M2	M1	M0	BCD

Table 4-10 Status Byte Description

Bit	Description
D7: OUT	Current state of Timers OUT pin
D6: LOAD	Count loaded into Timer
D5-D0	Timer Programmed Mode

Bit D7 contains the state of the Timers OUT pin. This allows viewing of the Timer's OUT pin via software.

Bit D6 indicates when the count written to the Timer is actually loaded into the Timer register. The exact time of the loading depends on the Mode the Timer is in and is defined in the Mode Definitions section. The count cannot be read from the Timer until it has been loaded. If a count is read before this time, the value read will not be the new count just written. Refer to Table 4-11.

Bits D5 through D0 contain the Timer's programmed mode exactly, bit for bit, like the Timer Control Words bits D5 through D0. See Table 4-2 on page 59.

Table 4-11 LOAD Bit Operation

Action	Causes
1. Write to the Control Word Register [1]	LOAD bit = 1
2. Write count to Timer [2]	LOAD bit = 1
3. New count loaded into Timer	LOAD bit = 0

[1]Only the Timer specified in the Control Word will have its LOAD bit set to 1. LOAD bits of other Timers are not affected.

[2]If the Timer is programmed for two byte counts (least significant then most significant), the LOAD bit will go to 1 when the second byte is written.

Both the count and status of the specified Timer(s) can be latched at the same time by setting both the $\overline{\text{Count}}$ bit (D5) and $\overline{\text{Status}}$ bit (D4) to zero (0) in the Read-Back command. If this technique is used, the first read operation of the Timer will return the status while the next one or two reads (depending on whether the Timer is programmed for one or two bytes) will return the count. Succeeding reads will return unlatched counts.

Mode Definitions

The VMIVME-7589A utilizes an 82C54 Timer / Counter for its Timers. 82C54 Timer / Counters can be programmed to function in six different modes (numbered Mode 0 through Mode 5). The VMIVME-7589A Timers are hardware configured to operate using Mode 2. Only Mode 2 is defined and supported.

Mode 2 functions as a divide by N counter. Once a Control Word and an initial count are written to the Timer the initial count is loaded on the next Clock cycle. When the count decrements to 1 an interrupt is generated. The Timer then reloads the initial count and the process repeats. This Mode is periodic. For an initial count of N, the sequence repeats every N CLK cycles. An initial count of 1 is illegal.

Writing a new count while the Timer is counting does not affect the current sequence. The new count will be loaded at the end of current sequence.

Flash Disk

The VMIVME-7589A features an on-board Flash mass storage system that allows the host computer to issue commands to read or write blocks to memory in a Flash memory array. This Flash Disk appears to the user as an intelligent ATA (IDE) disk drive with the same functionality and capabilities as a “rotating media” IDE hard drive. The rated life of the Flash disk is 300,000 writes to each sector.

Configuration

The Flash Disk resides on the VMIVME-7589A as the secondary IDE bus master device (the secondary IDE bus slave device is not assignable). The default setting in the Award BIOS ‘STANDARD CMOS SETUP’ screen is the ‘AUTO’ setting. In the Award BIOS ‘PERIPHERAL SETUP’ screen, the secondary PCI IDE interface must be enabled for the Flash Disk to be functional. Refer to Appendix C, Award-Basic Input/Output System for additional details.

Figure 4-6 maps the configuration possibilities for a typical system consisting of the VMIVME-7589A with a resident Flash Disk, a hard drive attached to the Primary IDE interface, and a floppy drive attached to the floppy interface.

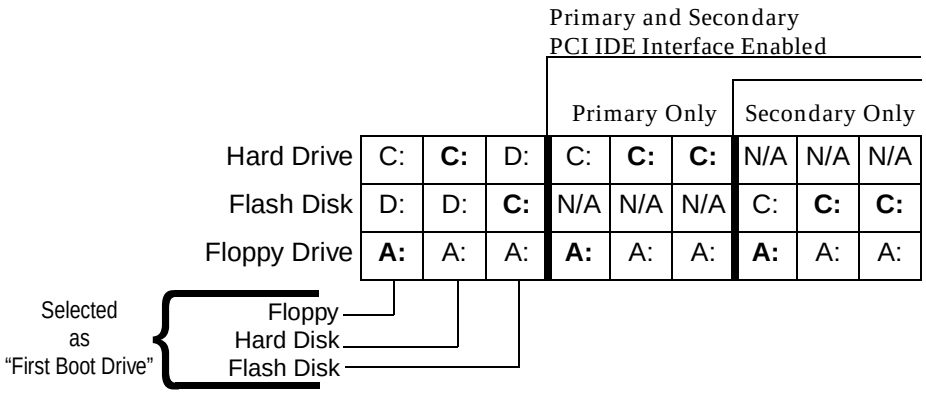


Figure 4-6 Typical System Configuration

The Primary and Secondary PCI IDE Interfaces are controlled (enabled or disabled) in the Peripheral Setup screen of the Award BIOS. The First Boot Device is selected in the Advanced CMOS Setup screen.

Figure 4-6 identifies the drive letter assigned to each physical device, and indicates in bold lettering the device booted from in each configuration, using devices that were bootable. Bootable being a device on which an operating system has been installed, or formatted as a system disk using MS-DOS.

If during the configuration efforts a drive device is identified as a USER setting in the Standard CMOS Setup screen, as occurs after using the Auto-Detect Hard Drives function of the BIOS, then disabled (the device's IDE interface disabled) in the Peripheral Setup screen, the following error may occur:

Primary (or Secondary) Master HHD Error
Run Setup
Press <ESC> to Resume

Press the Escape key as directed to resume the boot procedure. The settings should be changed to prevent the reoccurrence of this error condition

Functionality

The Flash Disk performs identically to a standard IDE hard drive. Reads and writes to the device are performed using the same methods, utilizing DOS command line entries or the file managers resident in the chosen operating system.

Advanced Configuration

The previous discussion is based on using the IDE disk devices formatted as one large partition per device. Some applications may require the use of multiple partitions. The following discussion of partitions includes the special procedures that must be followed to allow the creation of multiple partitions on the VMIVME-7589A IDE disk devices (including the resident Flash Disk).

Partitions may be either a primary partition or an extended partition. An extended partition may be subdivided farther into logical partitions. Each device may have up to four main partitions, one of which may be an extended partition. However, if multiple primary partitions are created, only one partition may be active at one time. Data in the non-active partitions are not accessible.

Following the creation of the partitioning scheme, the partitions can be formatted to contain the desired file system.

As discussed earlier, a typical system consists of the VMIVME-7589A with its resident Flash Disk configured as the Secondary IDE device, a hard drive attached to the Primary IDE interface, and a floppy drive attached to the floppy interface.

Using this configuration, it may be desirable to have a logical device on either IDE device configured as a bootable device, allowing the selection of the first boot device via the Advanced CMOS Setup screen. Using this capability, a user could have a system configured with multiple operating systems (OS's) that would then be selectable by assigning the IDE logical device as the boot device.

The DOS utility FDISK is commonly used to configure the partition structure on a hard drive. Other utility programs are available for performing this task. Partition Magic by PowerQuest is a popular and capable commercially available program. Comments that follow pertain to partitioning efforts using FDISK.

CAUTION: Deleting a partition will erase all the data previously held in that partition.

The Flash Disk will be configured as a single partition device as delivered from the factory. The following sample sequence illustrates a proven method for creating two 8 megabyte partitions, with one as an active primary partition. Take note of the instructions to exit FDISK. This has been shown to be an important step in a successful partitioning effort.

1. Power up the VMIVME-7589A, and enter CMOS Set-up.
2. Set Primary Master to “Not Installed”. Set Secondary Master to “Auto”.
3. Set boot device to floppy.
4. Boot DOS from the floppy, verify that the System Configuration Screen shows only the Flash Disk.
5. Run FDISK.
6. Delete all current partitions (any data currently stored in the partitions will be lost).
7. Exit FDISK, this will cause a reboot, then run FDISK again.
8. Create a 8 Meg primary partition.
9. Create a 8 Meg extended partition, and set-up a logical device for it.

Note

If only one partition is required it will be a 16 megabyte partition

10. Set the Primary partition as an active partition.
11. Exit FDISK.

Note

If an operating system has been installed on the Flash Disk that modifies the Master Boot Record (MBR), then the following step is required to rewrite the MBR for DOS

12. Run FDISK /MBR
13. Run FORMAT C: (use the /s option if you want the Flash Disk as a bootable DOS device.)

14. Format D: (This is only required if two partitions were created).
15. Reset the CPU, and enter the CMOS set-up.
16. Set Primary Master to “AUTO”.
17. Set boot device to desired boot source.

Drive letter assignments for a simple system were illustrated in Figure 4-6. Understanding the order the operating system assigns drive letters is necessary for these multiple partition configurations. The operating system assigns drive letter C: to the active primary partition on the first hard disk (the boot device). Drive D: is assigned to the first recognized primary partition on the next hard disk. The operating system will continue to assign drive letters to the primary partitions in an alternating fashion between the two drives. Next logical partitions will be assigned drive letters starting on the first hard drive lettering each logical device sequentially until they are all named, then doing the same sequential lettering of each logical partition on the second hard disk.

Drive letter changes caused by adding a drive or changing the initial partitioning scheme may cause difficulties with an operating system installed prior to the changes. Plan your configuration prior to installing the operating system to minimize difficulties.

Watchdog Timer

The VMIVME-7589A utilizes a Dallas DS1384 Watchdog Timekeeping Controller as its Watchdog Timer. The device provides a Time of Day feature, a Watchdog Alarm and an Non-Volatile SRAM controller. The Time of Day feature found within the DS1384 device is explained in this section, but is not utilized by the VMIVME-7589A. The actual Time of Day registers used by the VMIVME-7589A are located at the standard PC / AT I / O address. The Time of Day feature in the DS1384 Watchdog Timer is available for use by the user at their discretion.

The Non-Volatile SRAM is explained in the Battery Backed SRAM section of this manual.

The Watchdog Timer provides a Watchdog alarm window and interval timing between 0.01 and 99.99 seconds. If enabled, and if jumper J30 is loaded, the Watchdog alarm will reset the CPU to a known state if not accessed during the alarm window.

Figure 4-7 shows a generalized block diagram of how the Watchdog Timer is used in the VMIVME-7589A. The Watchdog Timer registers are memory-mapped in the bottom fourteen locations of battery-backed SRAM addresses $D8000 through $D800D. Table 4-12 shows the address, content and the range of each Watchdog Register.

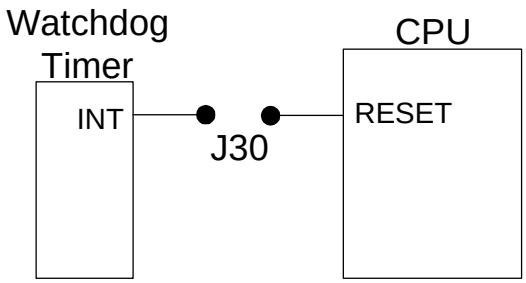


Figure 4-7 Watchdog Alarm Block

Table 4-12 Watchdog Registers

Register	Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Range
0	$D8000	0.1 Seconds (BCD)				0.01 Seconds (BCD)				00 - 99
1	$D8001	10 Seconds (BCD)				Seconds (BCD)				00 - 59
2	$D8002	10 Minutes (BCD)				Minutes (BCD)				00 - 59
3	$D8003	M	10 Minute Alarm (BCD)			Minute Alarm (BCD)				00 - 59
4	$D8004	0	12/24	AM/PM*	10 Hr	Hours (BCD)				
5	$D8005	M	12/24	AM/PM*	10 Hr	Hour Alarm (BCD)				
6	$D8006	0	0	0	0	Days (BCD)				01 - 07
7	$D8007	M	0	0	0	Day Alarm (BCD)				01 - 07
8	$D8008	0	0	10 Date (BCD)		Date (BCD)				01 - 31
9	$D8009	Eosc	1**	0	10 Mo	Months (BCD)				01 - 12
A	$D800A	10 Years (BCD)				Years (BCD)				00 - 99
B	$D800B	Te	Ipsw	Ibh/lo	Pu/lvl	Wam	Tdm	Waf	Tdf	
C	$D800C	0.1 Seconds (BCD)				0.01 Seconds (BCD)				00 - 99
D	$D800D	10 Seconds (BCD)				Seconds (BCD)				00 - 99

** In the 12 hour mode Bit 5 determines AM (0) or PM (1). In the 24 hour mode Bit 5 combines with Bit 4 to represent the 10 hour value.*

*** Bit 6 of Register 9 must be set to a 1. If set to a 0, an unused square wave will be generated in the circuit.*

Registers 0 through A are Clock, Calendar, Time of Day Registers.
Register B is the Command Register.
Registers C and D are Watchdog Alarm Registers.

The Watchdog Timer contains 14 registers which are 8-bits wide. These registers contain all of the Time of Day, Alarm, Watchdog, Control, and Data information. The Clock Calendar, Alarm, and Watchdog Registers have both external (user accessible) and internal memory locations containing copies of the data. The external memory locations are independent of the internal functions except that they are updated periodically by the transfer of the incremented internal values. Registers 0, 1, 2, 4, 6, 8, 9, and A contain Time of Day and Data information in Binary Coded Decimal (BCD). Registers 3, 5, and 7 contain the Time of Day Alarm information in BCD. The Command Register (Register B) contains data in binary. The Watchdog Alarm Registers are Registers C and D and information stored in these registers is in BCD.

Time of Day Registers

Registers 0, 1, 2, 4, 6, 8, 9, and A contain Time of Day data in BCD.

Register 0 contains two Time of Day values. Bits 3 - 0 contain the 0.01 Seconds value with a range of 0 to 9 in BCD while Bits 7 - 4 contain the 0.1 Seconds value with a range of 0 to 9 in BCD. This Register has a total range of 0.00 to 0.99 Seconds.

Register 1 contains two Time of Day values. Bits 3 - 0 contain the 1 Seconds value with a range of 0 to 9 in BCD while Bits 7 - 4 contain the 10 Seconds value with a range of 0 to 5 in BCD. This Register has a total range of 0.0 to 59.0 Seconds. Bit 7 of this register will always be zero regardless of what value is written to it.

Register 2 contains two Time of Day values. Bits 3 - 0 contain the 0.01 Seconds value with a range of 0 to 9 in BCD while Bits 7 - 4 contain the 0.1 Seconds value with a range of 0 to 9 in BCD. This Register has a total range of 0.00 to 0.99 Seconds. Bit 7 of this register will always be zero regardless of what value is written to it.

Register 4 contains the Hours value of the Time of Day. The Hours can be represented in either 12 or 24 hour format depending on the state of Bit 6. When Bit 6 is set to a one (1) the format is 12 hour. When Bit 6 is set to a zero (0) the format is 24 hour. For the 12 hour format Bits 3 - 0 contain the 1 Hour value with a range of 0 to 9 in BCD and Bit 4 contains the 10 Hour value with a range of 0 to 1. In the 12 hour format Bit 5 is used as the AM / PM bit. When AM Bit 5 is a zero (0) and when PM bit 5 is a one (1). The total range of this register in the 12 hour format is 01 AM to 12 AM and 01 PM to 12 PM.

When Register 4 is in 24 hour format (Bit 6 is set to a zero) Bits 3 - 0 contain the 1 Hour value with a range of 0 to 9 in BCD, Bit 5 combines with 4 to represent the 10 Hour value. The 10 Hour range is from 0 to 2. The total range of register 4 in the 24 hour format is 00 to 23 hours. Bit 7 of register 4 will always be zero regardless of what value is written to it and regardless of format (12 or 24 hour).

Register 6 contains the Days value of the Time of Day. Bits 2 - 0 contain the Days value with a range of 1 to 7 in BCD.

Register 8 contains two Time of Day values. Bits 3 - 0 contain the Date value with a range of 0 to 9 in BCD while Bits 5 - 4 contain the 10 Date value with a range of 0 to 3. This Register has a total range of 01 to 31. Bits 7 - 6 of this register will always be zero regardless of what value is written to it.

Register 9 contains two Time of Day values. Bits 3 - 0 contain the Months value with a range of 0 to 9 in BCD while Bits 4 contain the 10 Date value with a range of 0 to 1. This Register has a total range of 01 to 12. Bit 5 will always be zero regardless of what value is written to it. Bit 6 is unused but must be set to a 1. Bit 7, $\overline{\text{Eosc}}$, is the clock oscillator enable bit. When this bit is set to a zero (0) the oscillator is internally enabled. When set to a one (1) the oscillator is internally disabled. The oscillator via this bit is usually turned on once during system initialization but can be toggled on and off at the users discretion.

There are two techniques for reading the Time of Day from the Watchdog Timer. The first is to halt the external Time of Day registers from tracking the internal Time of Day registers by setting the Te bit (Bit 7 of the Command Register) to a logic zero (0) then reading the contents of the Time of Day registers. Using this technique eliminates the chance of the Time of Day changing while the read is taking place. At the end of the read, the Te bit is set to a logic one (1) allowing the external Time of Day registers to resume tracking the internal Time of Day Registers. No time is lost as the internal Time of Day Registers continue to keep time while the external Time of Day registers are halted. This is the recommended method.

The second technique for reading the Time of Day from the Watchdog Timer is to read the external Time of Day registers without halting the tracking of the internal Registers. This is not recommended as the registers may be updated while the reading is taking place, resulting in erroneous data being read.

Time of Day Alarm Registers

Registers 3, 5, and 7 are the Time of Day Alarm registers and are reformatted similar to Register 2, 4, and 6 respectively. Bit 7 of registers 3, 5, and 7 is a mask bit. The mask bits, when active (logic one), disable the use of the particular Time of Day Alarm register in the determination of the Time of Day Alarm (see Table 4-13). When all the mask bits are low (0) an alarm will occur when Registers 2, 4, and 6 match the values found in registers 3, 5, and 7. When Register 7's mask bit is set to a logic one (1) register 6 will be disregarded in the determination of the Time of Day alarm and an alarm will occur everyday. When registers 7 and 5's mask bit is set to a logic one (1), Register 6 and 4 will be disregarded in the determination of the Time of Day alarm and an alarm will occur every hour. When Registers 7, 5 and 3's mask bit is set to a logic one (1), Register 6, 4, and 2 will be disregarded in the determination of the Time of Day alarm and an alarm will occur every minute (when register 1's seconds step from 59 to 00).

Table 4-13 Time of Day Alarm Registers

Register			Comment
Minutes	Hours	Days	
1	1	1	Alarm once per minute
0	1	1	Alarm when minutes match
0	0	1	Alarm when hours and minutes match
0	0	0	Alarm when hours, minutes, and days match

The Time of Day Alarm registers are read and written to in the same format as the Time of Day registers. The Time of Day alarm flag and interrupt are cleared when the alarm registers are read or written.

Watchdog Alarm Registers

Register C contains two Watchdog alarm values. Bits 3 - 0 contain the 0.01 Seconds value with a range of 0 to 9 in BCD while Bits 7 - 4 contain the 0.1 Seconds value with a range of 0 to 9 in BCD. This Register has a total range of 0.00 to 0.99 Seconds.

Register D contains two Watchdog Alarm values. Bits 3 - 0 contain the 1 Second value with a range of 0 to 9 in BCD while Bits 7 - 4 contain the 10 Seconds value with a range of 0 to 9 in BCD. This Register has a total range of 00.0 to 99.0 Seconds.

The Watchdog Alarm Registers can be read or written in any order. When a new value is entered or the Watchdog registers are read, the Watchdog Timer will start counting down from the entered value. When zero is reached the Watchdog Interrupt Output will go active. If jumper J30 is loaded, the CPU will reset to a known state, refer to Figure 4-7. The Watchdog Timer count is reinitialized back to the entered value, the Watchdog flag bit is cleared, and the Watchdog interrupt output is cleared every time either of the registers are accessed. Periodic accesses to the Watchdog Timer will prevent the Watchdog Alarm from occurring. If access does not occur, the alarm will be repetitive. The Watchdog Alarm Register always reads the entered value. The actual countdown value is internal and not accessible to the user. Writing zero's to Registers C and D will disable the Watchdog Alarm feature.

Command Register

Register B is the Command Register. Within this register are mask bits, control bits, and flag bits. The following paragraphs describe each bit.

Te - Bit 7 Transfer Enable - This bit enables and disables the tracking of data between the internal and external registers. When set to a logic zero (0), tracking is disabled (data in the external register is frozen). When set to a logic one (1), tracking is enabled. This bit must be set to a logic one (1) to allow the external register to be updated.

Ipsw - Bit 6 Interrupt Switch - This bit toggles the Interrupt Output between the Time of Day Alarm and the Watchdog Alarm. When set to a logic zero (0), the Interrupt Output is from the Watchdog Alarm. When set to a logic one (1), the Interrupt Output is from the Time of Day Alarm.

Ibh/lo - Bit 5 Reserved - This bit should be set to a logic low (0).

Pu/lvl - Bit 4 Interrupt Pulse Mode or Level Mode - This bit determines whether the Interrupt Output will output as a pulse or a level. When set to a logic zero (0), Interrupt Output will be a level. When set to a logic one (1), Interrupt Output will be a pulse. In pulse mode the Interrupt Output will sink current for a minimum of 3 ms. This bit should be set to a logic one (1).

Wam - Bit 3 Watchdog Alarm Mask - Enables / Disables the Watchdog Alarm to Interrupt Output when Ipsw (Bit 6, Interrupt Switch) is set to logic zero (0). When set to a logic zero (0), Watchdog Alarm Interrupt Output will be enabled. When set to a logic one (1), Watchdog Alarm Interrupt Output will be disabled.

Tdm - Bit 2 Time of Day Alarm Mask - Enables / Disables the Time of Day Alarm to Interrupt Output when Ipsw (see Bit 6, Interrupt Switch) is set to logic one (1). When set to a logic zero (0), Time of Day Alarm Interrupt Output will be enabled. When set to a logic one (1), Time of Day Alarm Interrupt Output will be disabled.

Waf - Bit 1 Watchdog Alarm Flag - This is a read-only bit set to a logic one (1) when a Watchdog Alarm Interrupt occurs. This bit is reset when any of the Watchdog Alarm registers are accessed. When the Interrupt Output is set to Pulse Mode (see Bit 4, Interrupt Pulse Mode or Level Mode), the flag will be set to a logic one (1) only when the Interrupt Output is active.

Tdf - Bit 0 Time of Day Alarm Flag - This is a read-only bit set to a logic one (1) when a Time of Day Alarm Interrupt occurs. This bit is reset when any of the Time of Day Alarm registers are accessed. When the Interrupt Output is set to Pulse Mode (see Bit 4, Interrupt Pulse Mode or Level Mode), the flag will be set to a logic one (1) only when the Interrupt Output is active.

Battery Backed SRAM

The VMIVME-7589A includes 32 K byte of battery-backed SRAM addressed at $D8000 to $DFFFF (the lower 14 bytes, $D8000 to $D800D, are dedicated to Watchdog Timer registers and are unavailable for general use. See the Watchdog Timer section). The battery-backed SRAM can be accessed by the CPU at anytime, and can be used to store system data that must not be lost during power-off conditions.

CHAPTER

5

Maintenance

If a VMIC product malfunctions, please verify the following:

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

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

VMIC Customer Service is available at: 1-800-240-7782.
Or E-mail us at customer.service@vmic.com

Maintenance Prints

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

Appendix

A

Connector Pinouts

Contents

Introduction

The VMIVME-7589A PC / AT-Compatible VMEbus Controller has several connectors for its I / O ports. Figure A-1 shows the locations of the connectors on the VMIVME-7589A. Wherever possible, the VMIVME-7589A uses connectors and pinouts typical for any desktop PC. This ensures maximum compatibility with a varity of systems.

Connector diagrams in this appendix are generally shown in a natural orientation with the controller board mounted in a VMEbus chassis.

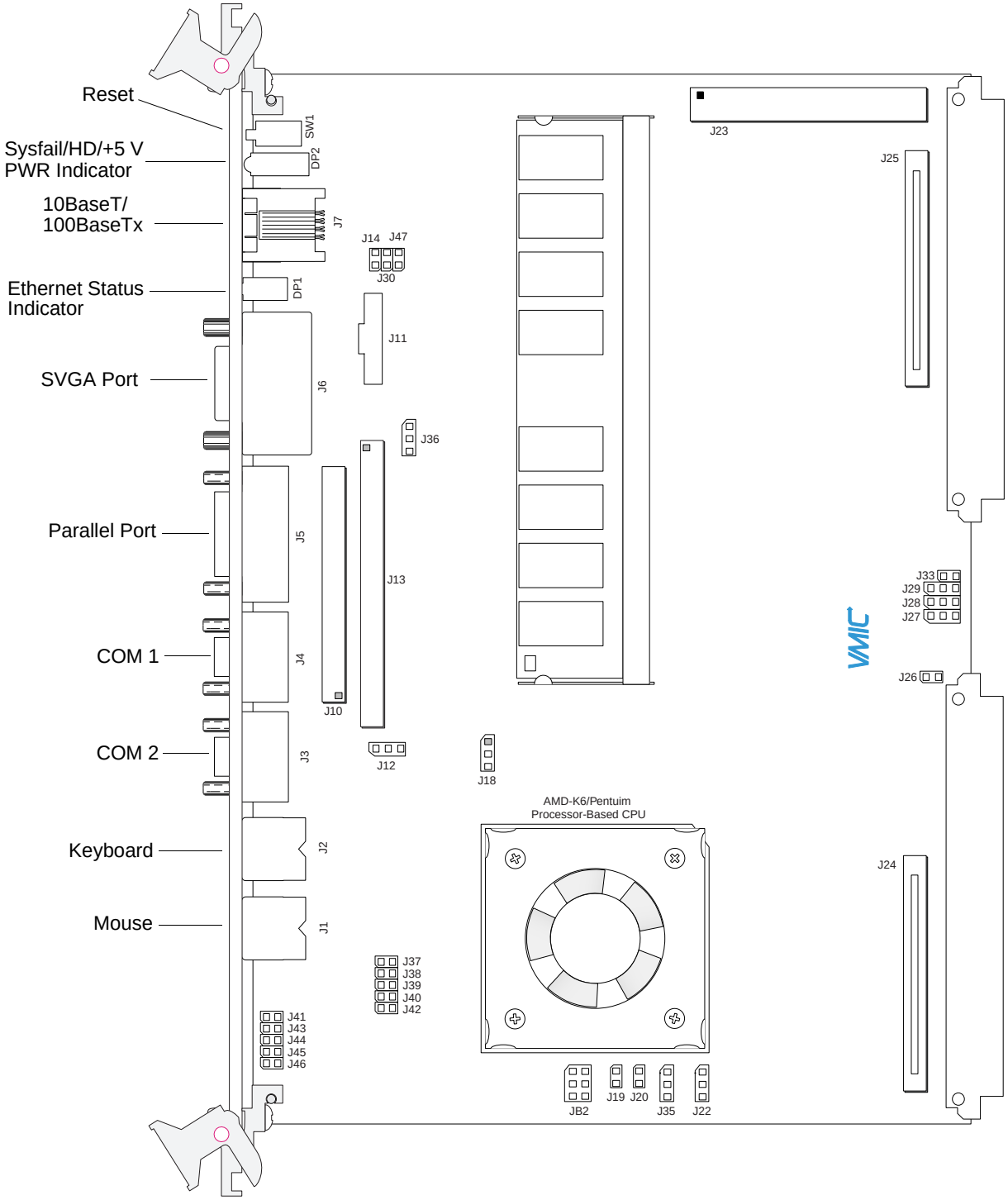
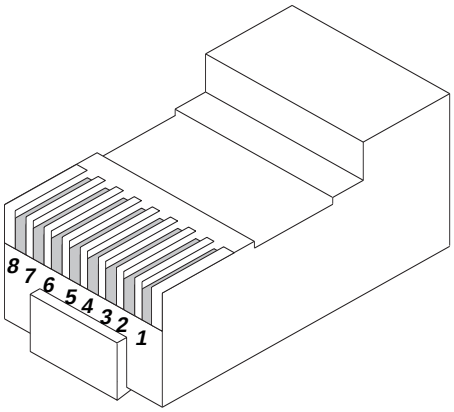


Figure A-1 VMIVME-7589A Connector and Jumper Locations

Ethernet Connector Pinout

The pinout diagram for the Ethernet 10BaseT and 100BaseTx connectors are shown in Figure A-2.



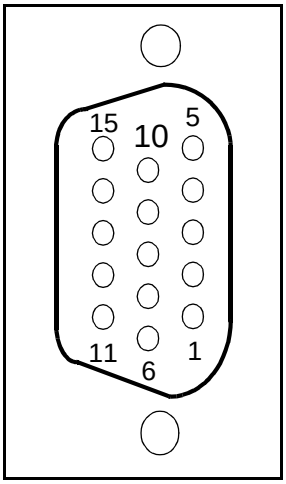
10BaseT, 100BaseTx

ETHERNET CONNECTOR (10BaseT, 100BaseTx)		
PIN	**Signal Name**	
1	TD+	Transmit Data
2	TD-	Transmit Data
3	RD+	Receive Data
4	NC	No Connection
5	NC	No Connection
6	RD-	Receive Data
7	NC	No Connection
8	NC	No Connection

Figure A-2 Ethernet Connector Pinout

Video Connector Pinout

The video port uses a standard high-density D15 SVGA connector. Figure A-3 illustrates the pinout.

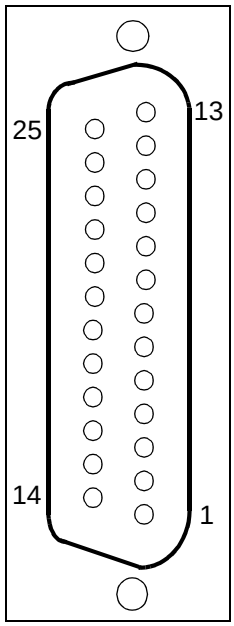


VIDEO CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1	Out	Red
2	Out	Green
3	Out	Blue
4		Reserved
5		Ground
6		Ground
7		Ground
8		Ground
9		Reserved
10		Ground
11		Reserved
12		Reserved
13	Out	Horizontal Sync
14	Out	Vertical Sync
15		Reserved
Shield		Chassis Ground

Figure A-3 Video Connector Pinout

Parallel Port Connector Pinout

The printer port shown in Figure A-4 uses a Microminiature D25 female connector typical of any PC / AT system.

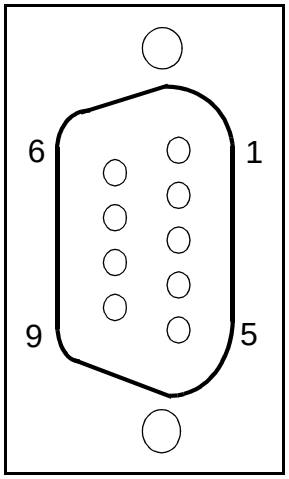


PARALLEL PORT CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1	In / Out	Data Strobe
2	In / Out	Bidirectional Data D0
3	In / Out	Bidirectional Data D1
4	In / Out	Bidirectional Data D2
5	In / Out	Bidirectional Data D3
6	In / Out	Bidirectional Data D4
7	In / Out	Bidirectional Data D5
8	In / Out	Bidirectional 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-4 Parallel Port Connector Pinout

Serial Connector Pinout

Each standard RS-232 serial port connector is a Microminiature D9 male as shown in the upper drawing in Figure A-5. The boards are supplied with adapters to connect standard D9 serial peripherals.



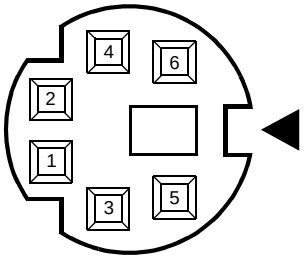
SERIAL COM PORT CONNECTORS			
D9 PIN	**DIR**	**RS-232 SIGNAL**	**FUNCTION**
1*	In	DCD	Data Carrier Detect
2	In	RX	Receive Data
3	Out	TX	Transmit Data
4	Out	DTR	Data Terminal Ready
5		GND	Signal Ground
6	In	DSR	Data Set Ready
7	Out	RTS	Request to Send
8	In	CTS	Clear to Send
9*	In	RI	Ring Indicator
Shield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

Keyboard Connector Pinout

The keyboard connector is a standard 6-pin female mini-DIN PS/2 connector as shown in Figure A-6.

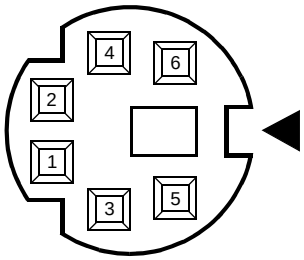


KEYBOARD CONNECTOR		
PIN	**DIR**	**FUNCTION**
1	In/Out	Data
2		Reserved
3		Ground
4		+5 V
5	Out	Clock
6		Reserved
Shield		Chassis Ground

Figure A-6 Keyboard Connector Pinout

Mouse Connector Pinout

The mouse connector is a standard 6-pin female mini-DIN PS/2 connector as shown in Figure A-7.

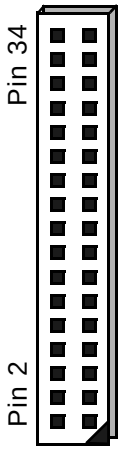


MOUSE CONNECTOR		
PIN	**DIR**	**FUNCTION**
1	In/Out	Data
2		Reserved
3		Ground
4		+5 V
5	Out	Clock
6		Reserved
Shield		Chassis Ground

Figure A-7 Mouse Connector Pinout

Floppy Interface Connector Pinout

The floppy drive connector is a dual-row 34-pin header connector and is located at site J10 on Figure A-1 on page 80. Pin 1 marks the beginning of the odd-numbered row. Most standard PC-compatible floppy disk drives use this pinout, which is illustrated in Figure A-8. The user should exercise caution when aligning the floppy drive connector and cable.

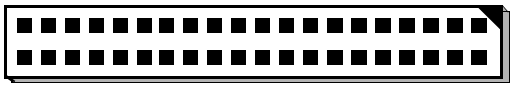


FLOPPY DRIVE CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1		Ground
2	From Controller	Drive Density 0 (not connected on some controllers)
3		Ground
4		Reserved
5		Key
6	From Controller	Drive Density 1 (not connected on some controllers)
7		Ground
8	From Drive	Index
9		Ground
10	From Controller	Motor Enable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

Figure A-8 Floppy Drive Connector Pinout

EIDE Hard Drive Connector Pinout

Figure A-9 describes the pin assignment for the 40-pin EIDE / ATA hard drive header connector. The connector is located at the J13 site in Figure A-1 on page 80. Like the floppy header connector it has an odd-numbered row and an even-numbered row. The user should exercise caution to align the hard drive connector and cable.



Pin 40 Pin 2

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	Out	IDE Request #0	22	Out	Signal Ground
23	Out	Write Strobe	24	Out	Signal Ground
25	Out	Read Strobe	26	Out	Signal Ground
27	Out	I / O Ready	28	Out	Address Latch Enable
29	Out	Data Acknowledge	30	Out	Signal Ground
31	In	Interrupt Request #14	32	In	IOCS16A#
33	Out	Address Line #1	34		N / C
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-9 EIDE Hard Drive Connector Pinout

Caution: If not using a VMIC supplied connector cable, the cable length should not exceed 18 inches.

VMEbus Connector Pinout

Figure A-10 shows the location of the VMEbus P1 and P2 connectors and their orientation. Table A-1 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.

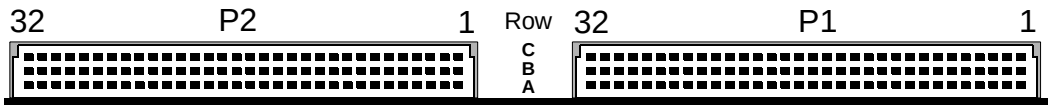


Figure A-10 VMEbus Connector Diagram

Table A-1 VMEbus Connector Pinout

PIN NUMBER	P1 ROW A SIGNAL	P1 ROW B SIGNAL	P1 ROW C SIGNAL	P2 ROW A SIGNAL	P2 ROW B SIGNAL	P2 ROW C SIGNAL
1	D00	BBSY	D08	GND	+5 V	Reset Drive
2	D01	BCLR	D09	Data 8	GND	Data 7
3	D02	ACFAIL	D10	Data 9	Reserved	Data 6
4	D03	BG0IN	D11	Data 10	A24	Data 5
5	D04	BG0OUT	D12	Data 11	A25	Data 4
6	D05	BG1IN	D13	Data 12	A26	Data 3
7	D06	BG1OUT	D14	Data 13	A27	Data 2
8	D07	BG2IN	D15	Data 14	A28	Data 1
9	GND	BG2OUT	GND	Data 15	A29	Data 0
10	SYSCLK	BG3IN	SYSFAIL	IDE Request #0	A30	IOCS16A#
11	GND	BG3OUT	BERR	Write Strobe	A31	GND
12	DS1	BR0	SYSRESET	Read Strobe	GND	GND
13	DS0	BR1	LWORD	I/O Ready	+5 V	GND
14	WRITE	BR2	AM5	GND	D16	Address Latch Enable
15	GND	BR3	A23	GND	D17	Data Acknowledge
16	DTACK	AM0	A22	GND	D18	Interrupt Request #14
17	GND	AM1	A21	Address Line #1	D19	Address Line #2

Table A-1 VMEbus Connector Pinout (Continued)

PIN NUMBER	P1 ROW A SIGNAL	P1 ROW B SIGNAL	P1 ROW C SIGNAL	P2 ROW A SIGNAL	P2 ROW B SIGNAL	P2 ROW C SIGNAL
18	AS	AM2	A20	Chip Select #0	D20	Address Line #0
19	GND	AM3	A19	Slave Activity Status	D21	Chip Select #1
20	IACK	GND	A18	Drive Density 1	D22	Drive Density 0
21	IACKIN	SERCLK	A17	GND	D23	Index
22	IACKOUT	SERDAT	A16	Drive Select B	GND	Motor Enable A
23	AM4	GND	A15	GND	D24	Drive Select A
24	A07	IRQ7	A14	GND	D25	Motor Enable B
25	A06	IRQ6	A13	GND	D26	Step Pulse
26	A05	IRQ5	A12	GND	D27	Write Data
27	A04	IRQ4	A11	GND	D28	Track 0
28	A03	IRQ3	A10	GND	D29	Read Data
29	A02	IRQ2	A09	Disk Change	D30	Select Head 1
30	A01	IRQ1	A08	GND	D31	Step Motor Dir.
31	-12 V	+5 V STDBY	+12 V	VCC	GND	Write Enable
32	+5 V	+5 V	+5 V	VCC	+5 V	Write Protect

PCI Expansion Connector Pinout

The following PCI Expansion Female Connector diagram illustrates orientation and Table A-2 defines the connector pin assignments for connector J23.

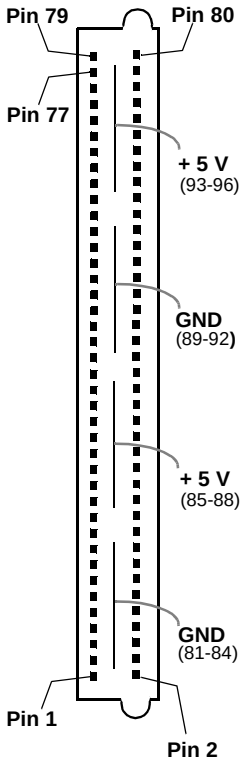


Table A-2 PCI Expansion Connector Pin Description

Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	+12 V	2	-12 V	49	C1BE0#	50	AD[8]
3	INTA#	4	INTB#	51	AD[6]	52	AD[7]
5	INTC#	6	INTD#	53	AD[4]	54	AD[5]
7	+12 V	8	CLK	55	AD[2]	56	AD[3]
9	RST#	10	REQ#	57	AD[0]	58	AD[1]
11	6NT#	12	AD[31]	59	NC	60	NC
13	AD[30]	14	AD[29]	61	BMODE1A#	62	BMODE1B#
15	AD[28]	16	AD[27]	63	IRQ4	64	IRQ3
17	AD[26]	18	AD[25]	65	IRQ7	66	IRQ5
19	AD[24]	20	C1BG3#	67	IRQ10	68	IRQ9
21	AD[22]	22	AD[23]	69	IRQ12	70	IRQ11
23	AD[20]	24	AD[21]	71	IRQ15	72	IRQ14
25	AD[18]	26	AD[19]	73	NC	74	NC
27	AD[16]	28	AD[17]	75	NC	76	NC
29	FRAME#	30	C1BG2#	77	NC	78	NC
31	TRDY#	32	IRDY#	79	NC	80	NC
33	STOP#	34	DEVSGL#	81	GND	82	GND
35	NC	36	LOCK#	83	GND	84	GND
37	NC	38	PERR#	85	+5 V	86	+5 V
39	PAR	40	SERR#	87	+5 V	88	+5 V
41	AD[15]	42	C1BG1#	89	GND	90	GND
43	AD[13]	44	AD[14]	91	GND	92	GND
45	AD[11]	46	AD[12]	93	+5 V	94	+5 V
47	AD[9]	48	AD[10]	95	+5 V	96	+5 V

Appendix

B

System Driver Software

Contents

Introduction

The VMIVME-7589A provides high-performance video, and Local Area Network (LAN) access by means of on-board PCI-based adapters and associated software drivers. The PCI-based video adapter used on the VMIVME-7589A is the Cirrus Logic 5480. High-performance LAN operation including 10BaseT and 100BaseTx, is provided by the DEC 21143 Ethernet controller chip.

To optimize performance of each of these PCI-based subsystems, the VMIVME-7589A is provided with software drivers compatible with DOS, Windows for Workgroups Version 3.11, Windows 95, and Windows NT operating systems. The following paragraphs provide instructions for loading and installing the adapter software.

Driver Software Installation

In order to properly use the Video and LAN adapters of the VMIVME-7589A, the user must install the driver software provided as distribution diskettes with the unit. Detailed instructions for installation of the drivers during installation of Windows for Workgroups Version 3.11, Windows 95, or Windows NT (Versions 3.5x and 4.0) operating systems are described in the following sections.

Windows for Workgroups (Version 3.11)

1. Format and load the IDE hard drive with MS-DOS.
2. Begin installation of Windows for Workgroups 3.11, following the instruction provided by Microsoft for Express Setup. When you reach the 'NETWORK SETUP' screen, the statement 'No Network Installed' will be displayed beneath the sentence starting with 'setup will install Windows support.

Note

If you do not require LAN operation, click on 'CONTINUE' and skip steps 3 through 12. Otherwise from the main 'NETWORK SETUP' screen, click the 'NETWORKS' button.

3. The following steps will load the DEC Ethernet drivers, and configure the DEC 21143 adapter.
4. Under 'NETWORKS', click on the circle to install 'MICROSOFT WINDOWS NETWORK'.
5. Then Click 'OK'.
6. The 'NETWORK SETUP SCREEN' appears again with the option for 'SHARING'. Click on 'SHARING' and choose the options that fit your system requirements by placing an X in the boxes shown. Then click on 'OK'.
7. Again at the 'NETWORK SETUP SCREEN', click on 'CONTINUE'.
8. Under 'ADD NETWORK ADAPTER' click on 'UNLISTED or UPDATED NETWORK ADAPTER'. Then click 'OK'.
9. Insert the VMIVME-7589A distribution disk labeled 320-500023-008 into drive A: and type: `A:\32bit\WFW311\.` Click 'OK'.
10. Under 'UNLISTED' or 'UPDATED NETWORK ADAPTER', choose 'Digital Semiconductor 21143-based 10/100 mpbs Ethernet Controller (ND153)'. Then click 'OK'.
11. Under 'MICROSOFT WINDOWS NETWORK NAMES', enter the network names you want for computer name, group name, etc. Then click 'OK'.
12. Windows for Workgroups should now continue with regular installation. During the remaining installation steps, use the full path name, A:\32bit\WFW311\, whenever prompted for the DEC 21143 driver diskette.
13. After Windows for Workgroups installation is complete, you may choose to install the Cirrus Logic video drivers. If you do not require the Cirrus Logic video drivers for operation, please skip steps 14 through 21.
14. Insert the VMIVME-7589A distribution diskette labeled 320-500023-001 into drive A:
15. From the Program Manager Screen double-click on the 'MAIN' icon.

16. Next double-click on the 'MS-DOS' icon.
17. Type the following command: **A:\INSTALL <ENTER>**. This will create directories on the C: drive and copy the video drivers and utilities. The default selections may be chosen for simplicity.
18. After the files are installed, the installation utilities boots WinMode which will allow the selection of the system monitor type.
19. In the 'WINMODE' screen, adjust the monitor settings as required and then click on 'OK'.
20. Click 'OK' to restart windows.
21. There will now be a 'VGA DISPLAY' icon in the 'PROGRAM MANAGER' window. In the 'VGA DISPLAY' window, there will be a 'WINMODE' icon that will allow the user to adjust the VGA display.
22. In order for the network to be setup properly, it is necessary to set the connection type for your system.

Proceed with the following steps to set your network connection type.

Note

If you are not running a network, please skip steps 23 through 27.

23. From the 'PROGRAM MANAGER' screen, double-click on the 'NETWORK' icon.
24. Then double-click on 'NETWORK SETUP' icon.
25. Under 'NETWORK SETUP', double-click on 'Digital Semiconductor 21143-based 10/100 mpbs Ethernet Controller (ND153)'.
26. Under 'ADVANCED NETWORK-ADAPTER SETTINGS', choose 'CONNECTION TYPE' and choose the 'CONNECTION TYPE VALUE' of '*AUTO SENSE TYPE*'. Click on 'SET' and then 'OK'.
27. Under 'NETWORK SETUP', click on 'OK'. If the network connection type has changed, then click 'OK' at the next screen for the information message about 'SYSTEM.INI' and 'PROTOCOL.INI' and click on 'RESTART COMPUTER' for the new network settings to take effect.

The unit should now be configured for operation in the WINDOWS FOR WORKGROUPS 3.11 environment.

Windows 95

1. Format the hard drive with MS-DOS.
2. Begin installation of Windows 95, following the instructions provided by the Windows 95 manual.
3. When you reach the 'WINDOWS 95 SETUP WIZARD SCREEN', choose 'TYPICAL' under 'SETUP OPTIONS' then click on 'NEXT'.
4. When you reach the 'ANALYZING YOUR COMPUTER' screen, place an X in the box for 'NETWORK ADAPTER', then click on 'NEXT'.
5. Under the 'WINDOWS COMPONENTS SCREEN', select 'INSTALL THE MOST COMMON COMPONENTS' and then click on 'NEXT'.
6. Continue with the installation until Windows 95 is completely installed and has rebooted.

Note

If you do not require LAN operation, skip steps 7 through 24.

7. From the main Windows 95 screen, click on 'START'.
8. Click on 'SETTINGS' and then 'CONTROL PANEL'.
9. Double-click on the 'SYSTEM' icon and select the 'DEVICE MANAGER' tab.
10. Double-click on 'OTHER DEVICES' and then double-click on 'PCI ETHERNET CONTROLLER'.
11. Select the 'DRIVER' tab and then select 'CHANGE DRIVER'.
12. In the 'SELECT HARDWARE TYPE' screen, choose 'NETWORK ADAPTERS' and click on 'OK'.
13. Insert the Diskette labeled 320-500023-008 into drive A:
14. In the 'SELECT DEVICE' window, click on 'HAVE DISK' and type `A:\32bit\WIN95\INF` and then click on 'OK'.
15. Under 'SELECT NETWORK ADAPTERS' choose 'Digital Semiconductor 21143-BASED 10/100 MPBS ETHERNET CONTROLLER', then click on 'OK'.
16. Under 'PCI ETHERNET CONTROL PROPERTIES' select 'OK'. Then the system will prompt you for computer and workgroup names. Type in the names you wish to use in the spaces shown and then choose 'CLOSE'.
17. Windows 95 will then prompt for diskettes. Follow all instructions.
18. At the 'SYSTEM PROPERTIES' window, click on 'OK'.
19. The driver files are located at A:\32bit\WIN95\. If prompted for the Windows 95 system disk, indicate the location of the files. For example, if you are loading from a CD-ROM located a D:, the desire response is D:\WIN95.

20. From ‘CONTROL PANEL’, double-click on the ‘NETWORK’ icon.
21. Under ‘NETWORK’, click on ‘FILE AND PRINT SHARING’ and choose the appropriate items for your system, click on ‘OK’.
22. Under the ‘NETWORK’ window, double-click on ‘DEC 21143-based 10/100 mpbs Ethernet Controller (ND153)’.
23. At the ‘NETWORK’ window, click on ‘OK’. When prompted, insert the diskettes needed to complete the network installation.
24. When the system prompts you to restart your computer, click on ‘YES’ for the network settings to take effect.
25. From the main Windows 95 screen, click on ‘START’.
26. Next, click on ‘SETTINGS’ and the ‘CONTROL PANEL’.
27. Double-click on the ‘DISPLAY’ icon and select the ‘SETTINGS’ tab.
28. Click on ‘CHANGE DISPLAY TYPE’, then click on ‘CHANGE’ in the ‘Adapter Type’ field.
29. Select ‘Cirrus Logic’ as the manufacturer, then click ‘HAVE DISK’.
30. Insert the diskette labeled 320-500023-004 into drive A:. Type ‘A:\’ as the files source (if not already displayed) and click on ‘OK’.
31. Under ‘SELECT DEVICE’, choose ‘Cirrus Logic 5480 PCI W/VPM (v.1.01b)’, then click on ‘OK’.
32. Insert the diskette labeled 320-500023-005 when prompted for the second disk, then click on ‘OK’.
33. When the CHOOSE DISPLAY TYPE’ screen returns, click on ‘CLOSE’.
34. Restart the computer, if prompted, to allow the new settings to take effect.

The unit should now be properly configured for operation in Windows 95.

Windows NT (Version 3.51)

1. Format the hard drive with MS-DOS.
2. Install the Windows NT boot disk in drive A:, and reboot the computer.
3. When prompted, insert Windows NT Setup Disk 2 into drive A:.
4. From the 'WELCOME TO SETUP' screen, press <ENTER> to set up Windows NT.
5. From the 'WINDOWS NT SETUP METHODS' screen, select the 'EXPRESS SETUP'.
6. Continue the installation procedure until the 'NETWORK ADAPTER CARD DETECTION' window is displayed. Click on 'CONTINUE' to manually select the network adapter card.
7. Choose 'CONTINUE' again when prompted.
8. In the 'ADD NETWORK ADAPTER' window, select the '<OTHER> REQUIRES DISK FROM MANUFACTURER' option, then click on 'CONTINUE'.
9. Insert disk 320-500023-008 into drive A:. Type A:\32bit\wnt351, then click 'OK'.
10. In the 'SELECT OEM OPTION' window, select Digital Semiconductor 21143-based 10/100 mpbs Ethernet Controller, then click on 'OK'.
11. In the 'Digital Semiconductor 21143-BASED 10/100 MPB ETHERNET CONTROLLER SETUP' window, select 'AUTOSENSE', then click on 'CONTINUE'.
12. A window will appear to allow the selection of network protocols. Deselect 'TCP/IP TRANSPORT' and select 'NetBEUI TRANSPORT'. Click on 'Continue'.
13. Enter the network information and account management information as required for the installation.
14. From the 'PROGRAM MANAGER' window, double-click on the 'MAIN' icon.
15. Then double-click on the 'CONTROL PANEL' icon.
16. Double-click on 'DISPLAY'.
17. Under 'DISPLAY SETTINGS' screen, click on 'CHANGE DISPLAY TYPE'.
18. Under 'DISPLAY TYPE' window, click on 'CHANGE'.
19. Under 'SELECT DEVICE' window, click on 'OTHER' button.
20. Insert disk 320-500023-003, then click on 'OK'.
21. Highlight 'CIRRUS LOGIC 256/64K/16M COLORS' and click on 'INSTALL'.
22. The message window 'INSTALLING DRIVERS' appears letting you know that the drivers were successfully installed. Click on 'OK'.
23. At the next screen, press 'OK' to restart.
24. Then press 'RESTART NOW'.

The unit should now be properly configured for Windows NT 3.51.

Windows NT (Version 4.0)

Windows NT 4.0 includes drivers for the on-board LAN, and video adapters. The following steps are required to configure the LAN for operation.

1. Follow the normal Windows NT 4.0 installation until you reach the 'WINDOWS NT WORKSTATION SETUP' window which states that 'WINDOWS NT NEEDS TO KNOW HOW THIS COMPUTER SHOULD PARTICIPATE ON A NETWORK'.
2. Place a dot next to 'THIS COMPUTER WILL PARTICIPATE ON A NETWORK'.
3. Place a check mark next to 'WIRED TO THE NETWORK' and then click on 'NEXT'.
4. At the next screen, click on the 'SELECT FROM LIST' button.
5. Click on the 'HAVE DISK' button.
6. Insert disk 320-500023-008 into drive A:.
7. Type A:\32bit\wnt40 and click on 'OK'.
8. In the "SELECT OEM OPTION', choose 'Digital Semiconductor 21143-based 10/100 mpbs Ethernet Controller (NDIS3)', then click 'OK'.
9. Select the above entry on the displayed list, click on 'NEXT'.
10. Select the NetBEUI Protocol (only), click on 'NEXT'.
11. Click 'CONTINUE' to allow an Autosense connection type.
12. Step through the remaining screens, providing the data pertinent to your network.
13. Continue through the setup procedure until the 'DETECTED DISPLAY' window appears, click 'OK' to continue.
14. In the 'DISPLAY PROPERTIES' window, click on 'TEST'.

Please note that Windows NT 4.0 does not allow the selection of the Cirrus Logic 5480 drivers during initial setup.

If the display test is successful, click on 'OK' to continue. If the display test is not successful, you may have to adjust the display parameters to find a functional setting, for example a lower resolution or lower number of colors.

15. Continue with the procedure to the 'Windows NT Setup' window. Click on 'RESTART COMPUTER'.
16. When the computer reboots, double-click on 'MY COMPUTER' window.
17. Double-click on the 'CONTROL PANEL' icon in the 'MY COMPUTER' window.

18. Double-click on the ‘DISPLAY’ icon in the ‘CONTROL PANEL’.
19. Select the ‘SETTINGS’ tab in the ‘DISPLAY PROPERTIES’ window, then click on the ‘DISPLAY TYPE’ button.
20. In the ‘DISPLAY TYPE’ window, click on ‘CHANGE’.
21. In the ‘CHANGE DISPLAY’ window, click on ‘HAVE DISK’.
22. Insert disk 320-500023-003 into drive A:, then click on ‘OK’.
23. “Cirrus Logic CL-GD54xx Graphics Adapter’ will be displayed in the ‘CHANGE DISPLAY’ window. Click on ‘OK’.
24. Proceed as directed, removing the driver disk from the floppy drive, and the computer will restart to activate the new settings. When the system reboots, the display settings may be configured for your display type.

The unit should now be configured for operation under Windows NT 4.0.

Appendix

C

LANWorks BIOS

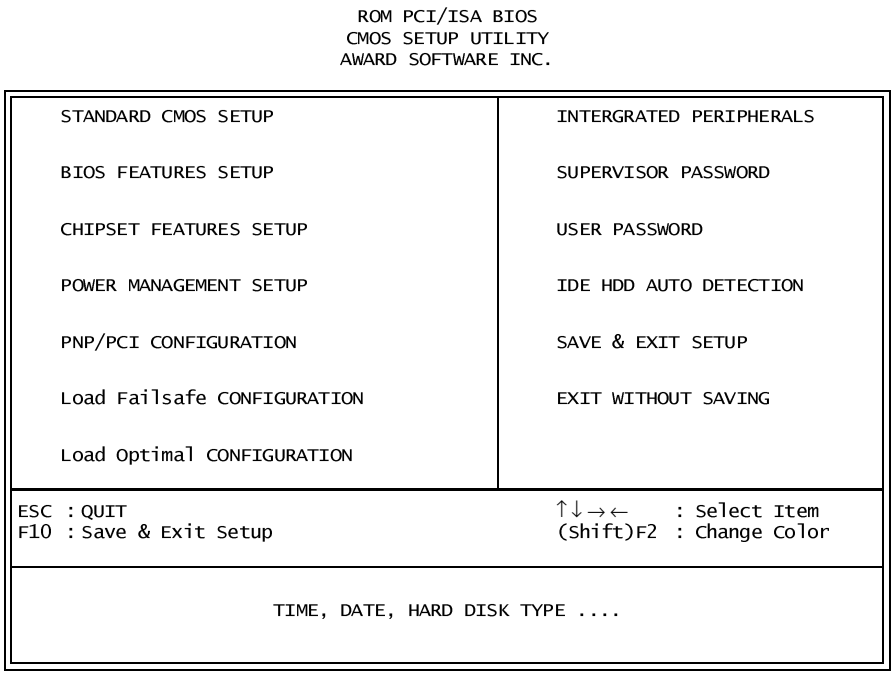
Contents

Introduction

The VMIVME-7589A includes the LANWorks option which allows the VMIVME-7589A to be booted from a network. This appendix describes the LANWorks BIOS Setup screen, and the procedures to enable this option.

System BIOS Setup Utility

To enable the LANWorks BIOS option reboot the VMIVME-7589A and when prompted, press the Delete key to access the System BIOS Setup Utility screen shown below.



Using the arrow keys, highlight *BIOS Features Setup*, and press the Return key. This will display the BIOS Features Setup screen.

BIOS Features Setup

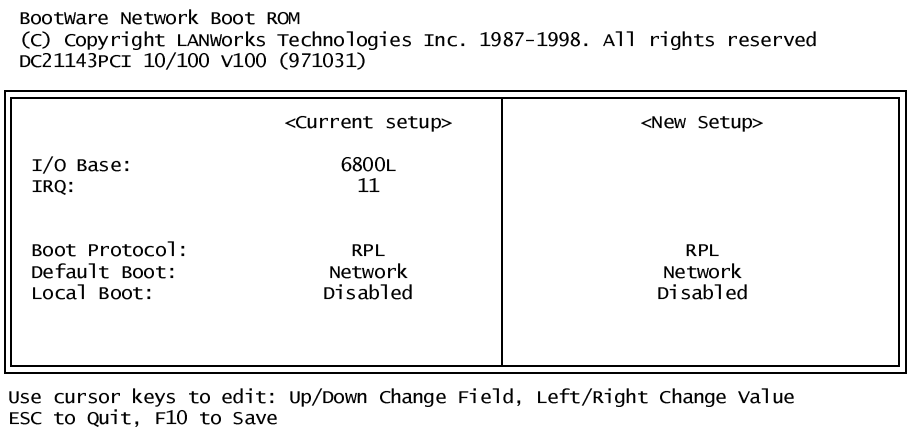
```
                    ROM PCI/ISA BIOS
                BIOS FEATURES SETUP
                AWARD SOFTWARE INC.

Virus Warning                : Disabled     Video  BIOS Shadow   :  Enabled
CPU Internal Cache           : Enabled      C8000-CBFFF Shadow   :  Disabled
External Cache               : Enabled      CC000-CFFFF Shadow   :  Disabled
Quick Power On Self Test     : Enabled      D0000-D3FFF Shadow   :  Disabled
Boot From LAN First          : Enabled      D4000-D7FFF Shadow   :  Disabled
Load Network Boot Code       : Enabled      D8000-DBFFF Shadow   :  Disabled
Boot Sequence                : Floppy       DC000-DFFFF Shadow   :  Disabled
Swap Floppy Drive            : Disabled
Boot Up Floppy Seek          : Enabled
Boot Up NumLock Status       : On
Boot Up System Speed         : High
Typematic Rate Setting       : Disabled
Typematic Rate (Chars/Sec)   : 6
Typematic Delay (Msec)       : 250
Security Option              : Setup
PCI/VGA Palette Snoop        : Disabled
OS Select For DRAM > 64MB    : Non-OS2
Report No FDD For WIN 95     : No

ESC  : Quit          ↑↓→←:Select Item
F1   : Help          PU/PD/+/- : Modify
F5   : Old Values    (Shift)F2 : Color
F6   : Load Failsafe Defaults
F7   : Load Optimal Defaults
```

Using the arrow keys, enable *Boot From LAN First* and *Load Network Boot Code*. Exit the BIOS setup, saving changes.

When prompted, press “Control-Alt-B” as the VMIVME-7589A reboots. This will activate the LANWorks BIOS setup screen.

LANWorks BIOS Setup

Below is the screen in which the various options for booting through LANWorks are set.



Boot Protocol

The Boot Protocol required is dependent on the network system being utilized, check with your network administrator for the correct protocol. The following options are available in the Boot Protocol field:

- **RPL** **(Default)**
- TCP/IP BootP Used for Real Time operating systems
- TCP/IP DHCP Not Supported
- Netware 802.3 Not Supported
- Netware 802.2 Not Supported
- Netware EthII Not Supported

Default Boot

The following options are available in the Default Boot field:

- **Network (Default)** - Selects Network Boot as the default boot device
- Local - Selects a Local Drive as the default boot device

Local Boot

The following options are available in the Local Boot field:

- **Disabled (Default)** - Disables Local Boot from a Hard Drive or Floppy
- Enabled - Enables Local Boot from a Hard Drive or Floppy

Appendix

D

Award - BIOS

Contents

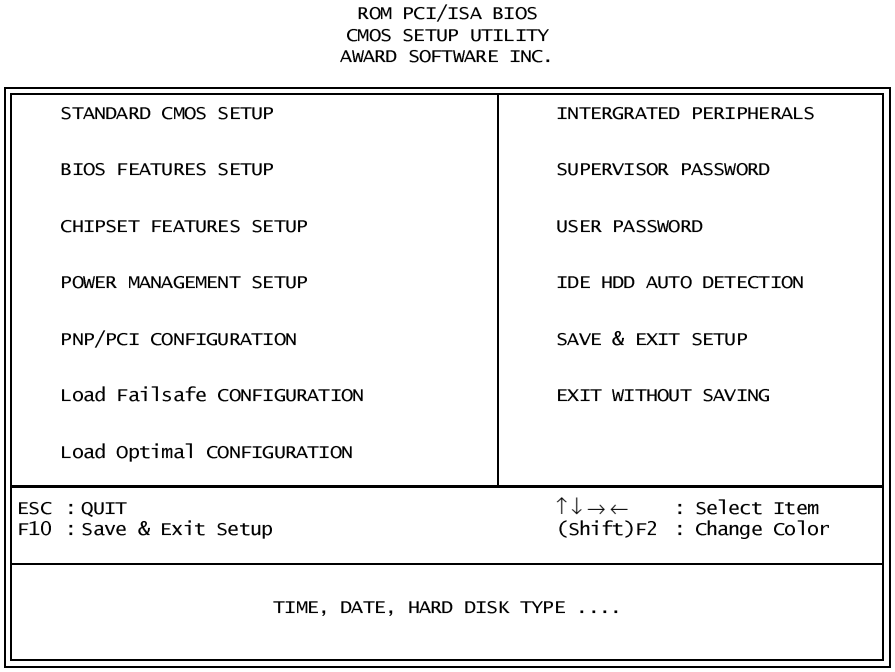
Introduction

The VMIVME-7589A utilizes the BIOS (Basic Input / Output system) in the same manner as other PC / AT compatible computers. This appendix describes the menus and options associated with the VMIVME-7589A BIOS.

System BIOS Setup Utility

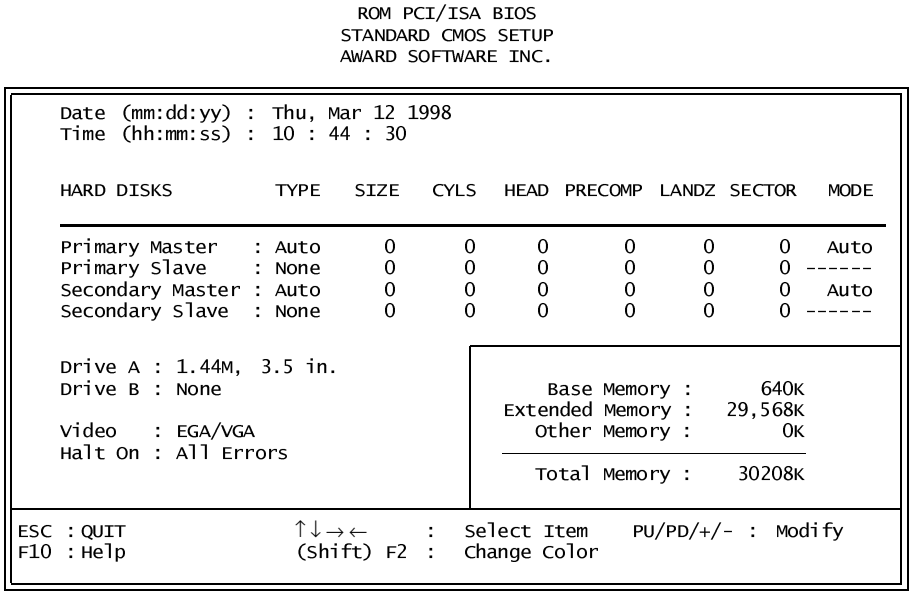
During system bootup, press the Delete key to access the Award *Elite*BIOS CMOS Setup Utility screen. From this screen, the user can select any section of the Award (system) BIOS for configuration, such as floppy drive configuration or system memory.

The parameters shown throughout this section are the default values.



Standard CMOS Setup

Selection of the first main menu item, the standard CMOS Setup, allows the user to set the system clock and calendar, record disk drive parameters, video subsystem type, and select the type of errors that will cause a system halt.



Setting The Date

Press the left or right arrow key to move the cursor to the desired field (month, day, year). Press the PgUp or PgDn key to step through the available choices, or type in the information. Note that the day of the week field is for information purposes only, and will be set based on the other date information.

Setting The Time

The time format is based on the 24-hour military-time clock. For example, 1 PM is 13:00:00. Press the left or right arrow key to move the cursor to the desired field (hour, minute, seconds). Press the PgUp or PgDn key to step through the available choices, or type in the information.

Primary Master/Slave

The VMIVME-7589A has the capability of utilizing one IDE hard disk drive on the Primary Master bus. The default setting is Auto. The Primary Slave is not used with the VMIVME-7589A.

Secondary Master/Slave

The Secondary Master is the resident 8 Mbyte flash disk. The default setting is Auto. The Secondary Slave is not assignable.

Floppy Disk Drive

Floppy Drive A

The VMIVME-7589A supports one floppy disk drive. The options are:

- None — No diskette drive installed
- 360K, 5.25 in — 5-1/4 inch PC-type standard drive; 360 kilobyte capacity
- 1.2M, 5.25 in — 5-1/4 inch AT-type high-density drive; 1.2 megabyte capacity
- 720K, 3.5 in — 3-1/2 double-sided drive; 720 kilobyte capacity
- 1.44M, 3.5 in — 3-1/2 inch double-sided drive; 1.44 megabyte capacity
- 2.88M, 3.5 in — 3-1/2 inch double-sided drive; 2.88 megabyte capacity

Use PgUp or Pgdn to select the floppy drive. The default is 1.44M, 3.5 inch.

Floppy Drive B

The VMIVME-7589A does not support a second floppy drive. The default is None.

Video

The VMIVME-7589A has an EGA/VGA graphics chip onboard. The BIOS supports a secondary video subsystem, but it is not selected in Setup. The default is EGA/VGA. Use the PgUp or PgDn key to select the video.

Halt On

During the power-on self-test (POST), the computer may be made to halt if the BIOS detects a hardware error. The options are:

- All Errors — If the BIOS detects any non-fatal error, POST stops and prompts you for corrective action
- No Errors — POST does not stop for any errors
- All, But Keyboard — Post does not stop for a keyboard error, but stops for all other errors
- All, But Diskette — POST does not stop for diskette drive errors, but stops for all other errors
- All, But Disk/Key — POST does not stop for a keyboard or disk error, but stops for all other errors

Use the PgUp or PgDn key to select the errors which will halt the system. The default is All Errors.

Memory

The Memory field at the lower right of the screen is for informational purposes only and can not be modified by the user. This field displays the total RAM installed in the system, and the amounts allocated to base, extended, and other (high) memory.

BIOS Features Setup

This screen, selected from the CMOS Setup Utility screen, allows the user to configure options that are in addition to the basic BIOS features.

```
                    ROM PCI/ISA BIOS
                 BIOS FEATURES SETUP
                AWARD SOFTWARE INC.

Virus Warning                : Enabled      Video  BIOS Shadow   :  Enabled
CPU Internal Cache           : Enabled      C8000-CBFFF Shadow   :  Disabled
External Cache               : Enabled      CC000-CFFFF Shadow   :  Disabled
Quick Power On Self Test     : Enabled      D0000-D3FFF Shadow   :  Disabled
Boot From LAN First          : Disabled     D4000-D7FFF Shadow   :  Disabled
Load Network Boot Code       : Disabled     D8000-DBFFF Shadow   :  Disabled
Boot Sequence                : A,C,SCSI     DC000-DFFFF Shadow   :  Disabled
Swap Floppy Drive            : Disabled
Boot Up Floppy Seek          : Enabled
Boot Up NumLock Status       : On
Boot Up System Speed         : High
Typematic Rate Setting       : Disabled
Typematic Rate (Chars/Sec)   : 6
Typematic Delay (Msec)       : 250
Security Option              : Setup
PCI/VGA Palette Snoop        : Disabled
OS Select For DRAM > 64MB    : Non-OS2
Report No FDD For WIN 95     : No

ESC  : Quit          ↑↓→←:Select Item
F1   : Help          PU/PD/+/- : Modify
F5   : Old Values    (Shift)F2 : Color
F6   : Load Failsafe Defaults
F7   : Load Optimal Defaults
```

Virus Warning

When enabled the system provides a warning if an attempt is made to write to the boot sector or the partition table of the hard disk drive the system is booted from. If a warning is received, it is recommended that an anti-virus program be run. Please note that this feature protects only the boot sector, not the entire hard drive. The default is Enabled.

CPU Internal Cache

Enabling the cache memory enhances the speed of the processor. When the CPU requests data, the system transfers the requested data from the main DRAM into the cache memory were it is stored until processed by the CPU. The default is Enabled.

External Cache

The external cache (up to 512K) provides additional cache for use by the CPU. This cache functions in the same manner as internal cache. The default is Enabled.

Quick Power On Self Test

When enabled, certain checks normally performed during the POST are omitted, decreasing the time required to run the POST. Enabling this function is not recommended. The default is Enabled.

Boot From LAN First

When enabled, this option allows the CPU to boot off of a connected network. The default is Enabled.

Boot Sequence

Determines the order in which the BIOS will seek a bootable drive. The default order is the floppy disk (A:), then the internal hard drive (C:), followed by the SCSI drive, if attached.

Swap Floppy Drive

The option is functional only in a system with two floppy drives. The VMIVME-7589A supports only one floppy drive. Changing this option will have no effect on the system. The default is Disabled.

Boot Up Floppy Seek

When enabled, the BIOS will test the floppy to determine if the floppy drive has 40 or 80 tracks. Only 360K floppy drives have 40 tracks. It is recommended that this option be set to disabled unless a third party 360K floppy drive is being used in the system. The default is Disabled.

Boot Up NumLock Status

Toggle between On or Off to control the state of the NumLock key when the system boots. When toggled On, the numeric keypad generates numbers instead of controlling the cursor operations.

Boot Up System Speed

This option determines if the system boots at the default CPU speed (high) or at the AT bus speed (low). Some add-on peripherals or legacy software may require the system to be booted at the lower speed. The default is High.

Typematic Rate Setting

This option enables or disables the Typematic Rate and Delay settings. When disabled the values in the Typematic Rate and Delay are ignored. The default is Disabled.

Typematic Rate (Chars/Sec)

If the Typematic rate Setting is enabled this determines the rate a character is repeated when a key is held down. The options are: 6, 8, 10, 12, 15, 20, 24, or 30 characters per second.

Typematic Delay (Msec)

If the Typematic rate Setting is enabled this determines the delay before a character starts repeating when a key is held down. The options are: 250, 500, 750, or 1000 milliseconds.

Security Option

If a password has been set, this determines whether the password is required every time the system boots, or only when the Setup is accessed. The default is Setup.

PCI/VGA Palette Snoop

Enabling the video palette snoop allows the ISA add-in card to share a common palette with the on-board graphics controller. The default is Disabled.

OS Select For DRAM>64MB

Select OS2 only if you are running OS/2 operating system with greater than 64 MB of RAM on the system. The default is Non-OS2.

Report No FDD For WIN 95

Selecting Yes for this option releases IRQ 6 when the system does not contain a floppy drive, for compatibility with Windows 95 logo certification. In the Integrated Peripherals screen, select Disabled for the Onboard FDC Controller field. The default is No.

Chipset Features Setup

This section describes features of the Intel 82430TX PCIset.

Advanced Options

The parameters in this screen are for system designers, service personnel, and technically competent users only. Do not reset these values without a complete understanding of the consequences.

```
                    ROM PCI/ISA BIOS
                CHIPSET FEATURES SETUP
                AWARD SOFTWARE INC.

Auto Configuration          : Enabled
DRAM Timing                 : 70ns

DRAM Leadoff Timing         : 10/6/4
DRAM Read Burst (EDO/FP)    : X333/X444
DRAM Write Burst Timing     : X222
Refresh RAS# Assertion      : 5 Clks
Fast RAS to CAS Delay       : 3
DRAM Page Idle Timer        : 2 Clks
DRAM Enhanced Paging        : Enabled
Fast MA to RAS# Delay       : 2 Clks
SDRAM(CAS Lat/RAS-to-CAS)   : 2/2
SDRAM Speculative Read      : Disabled
System BIOS Cacheable       : Disabled   ESC  : Quit       ↑↓→←:Select Item
Video BIOS Cacheable        : Disabled   F1   : Help       PU/PD/+/- : Modify
8 Bit I/O Recovery Time     : 1          F5   : Old Values (Shift)F2 : Color
16 Bit I/O Recovery Time    : 2          F6   : Load Failsafe Defaults
Memory Hole At 15M-16M      : Disabled   F7   : Load Optimal Defaults
PCI 2.1 Compliance          : Disabled
```

Auto Configuration

Auto Configuration selects predetermined optimal values of chipset parameters. When disabled, chipset parameters revert to the setup information stored in CMOS. The default is Enabled.

DRAM Timing

The value in this field depends on performance parameters of the installed memory chips (DRAM). Do not change the value from the factory setting unless new memory has been installed with a different performance rating than the original DRAMs. The default is 70ns.

DRAM Leadoff Timing

Select the combination of CPU clocks the DRAM on the system board requires before each read from or write to the memory. Changing the value of this setting from the default may cause memory errors. The default is 10/6/4.

DRAM Read Burst (EDO/FP)

Sets the timing for reads from EDO (Extended Data Output) or FP[M] (Fast Page Mode) memory. The lower the timing numbers, the faster the system addresses memory. Selecting timing numbers lower than the installed DRAM is able to support can result in memory errors. The default is x333/x444.

DRAM Write Burst Timing

Sets the timing for writes to memory. The lower the timing numbers, the faster the system addresses memory. Selecting timing numbers lower than the installed DRAM is able to support can result in memory errors. The default is x333.

Refresh RAS# Assertion

Select the number of clock cycles in which RAS# is asserted for refresh cycles. The default is 5 Clks.

Fast RAS to CAS Delay

This field inserts a timing delay between the CAS and RAS strobe signals, used when DRAM is written to, read from, or refreshed. The default is 2.

DRAM Page Idle Timer

Select the amount of time in HCLKs that the DRAM controller waits to close a DRAM page after the CPU becomes idle. The default is 2 Clks.

DRAM Enhanced Paging

When Enabled, the chipset keeps the page open until a page/row miss. When Disabled, the chipset uses additional information to keep the DRAM page open when the host may be “right back.” The default Enabled.

Fast MA to RAS# Delay

The values in this field are set by the system board designer, depending on the DRAM installed. Do not change the values in this field unless you change specifications of the installed DRAM or the installed CPU. The default is 2 Clks

SDRAM(CAS Lat/RAS-to-CAS)

You can select a combination of CAS latency and RAS-to-CAS delay in HCLKs of 2/2 or 3/3. The values in this field depend on the DRAM installed. Do not change the values in this field unless you change specifications of the installed DRAM or the installed CPU. The default is 2/2.

SDRAM Speculative Read

The chipset can “speculate” on a DRAM read address, thus reducing read latencies. The CPU issues a read request containing the data memory address. The DRAM controller receives the request. When this field is Enabled, the controller issues the read command slightly before it has finished decoding the data address. The default is Disabled.

System BIOS Cacheable

Selecting Enabled allows caching of the system BIOS ROM at F0000h-FFFFFh. If a program writes to this memory area, a system error may occur. The default is Disabled.

Video BIOS Cacheable

Selecting Enabled allows caching of the video BIOS ROM at C0000h to C7FFFh. If a program writes to this memory area, a system error may occur. The default is Disabled.

8 Bit I/O Recovery Time

The I/O recovery mechanism adds bus clock cycles between PCI-originated I/O cycles to the ISA bus. This delay takes place because the PCI bus is faster than the ISA bus. The default is 1.

16 Bit I/O Recovery Time

The I/O recovery mechanism adds bus clock cycles between PCI-originated I/O cycles to the ISA bus. This delay takes place because the PCI bus is faster than the ISA bus. The default is 2.

These two fields (8 Bit and 16 Bit) let you add recovery time (in bus clock cycles) for 8-bit and 16-bit I/O.

Memory Hole at 15M-16M

This area of system memory may be reserved for ISA adapter ROM. When this area is reserved, it cannot be cached. Refer to the documentation that came with the peripheral that require the use of this area of system memory for memory requirements. The default is Disabled.

PCI 2.1 Compliance

Select Enabled to support compliance with PCI specification version 2.1. The default is Disabled.

Power Management

This section discusses the power management features provided with the installed Award BIOS.

```
                    ROM PCI/ISA BIOS
                POWER MANAGEMENT SETUP
                AWARD SOFTWARE INC.

ACPI Function              : Disabled      ** Reload Global Timer Events **
Power Management           : Disabled      IRQ  [3-7,9-15],NMI: Enabled
PM Control by APM          : Yes           Primary IDE 0      : Disabled
Video Off Method           : V/H SYNC+Blank Primary IDE 1     : Disabled
Video Off After            : Standby                          : Secondary IDE 0:D
Modem Use IRQ              : 3                                : Secondary IDE 1:0
Doze Mode                  : Disabled      Flopy disk         : Disabled
Standby Mode               : Disabled      Serial Port        : Disabled
Suspend Mode               : Disabled      Parallel Port      : Disabled
HDD Power Down             : Disabled
Throttle Duty Cycle        : 62.5%
22 Active in Suspend       : Disabled
PCI/VGA Act-Monitor        : Enabled
Soft-off by PWR-BTTN       : Instant-Off
Power On by Ring           : Enabled
IRQ 8 Break Suspend        : Disabled

                                           ESC  : Quit        ↑↓→←:Select Item
                                           F1   : Help        PU/PD/+/- : Modify
                                           F5   : Old Values  (Shift)F2 : Color
                                           F6   : Load Failsafe Defaults
                                           F7   : Load Optimal Defaults
```

ACPI Function

Select Enabled only if your computer's operating system supports the Advanced Configuration and Power Interface (ACPI) specification. Currently, Windows 98 and Windows NT 5.0 support ACPI. The default is Disabled.

Power Management

This option disables or sets the power management options. The options are:

- Max SavingMaximum power savings. This option is only available for SL CPUs.
  Inactivity period is 1 minute in each mode.
- User defineSet each mode individually. Select time-out periods in the PM timers section.
- Min SavingsMinimum power savings. Inactivity period is 1 hour in each mode (except the hard drive).

- DisabledTurns off all power management features.

The default is Disabled.

PM Control by APM

Advanced Power Management (APM) provides better power savings. The default is Yes.

Video Off Method

Determines the manner in which the monitor is blanked. The options are:

- V / H SYNC+BlankSystem turns off vertical and horizontal synchronization ports and writes blanks to the video buffer.
- DPMS SupportSelect this option if your monitor supports the Display Power Management Signaling (DPMS) standard of the Video Electronics Standards Association (VESA). Use the software supplied for your video subsystem to select video power management values.
- Blank ScreenSystem only writes blanks to the video buffer.

The default setting is V / H SYNC+Blank.

Video Off After

As the system moves from lesser to greater power-saving modes, select the mode in which you want the monitor to blank. The default is Standby.

Modem Use IRQ

Name the interrupt request (IRQ) line assigned to the modem (if any) on your system. Activity of the selected IRQ awakens the system. The default setting is 3.

Doze Mode

After the selected period of system inactivity (1 minute to 1 hour), the CPU clock runs at a slower speed while all other devices still operate at full speed. The default is Disabled.

Standby Mode

After the selected period of system inactivity (1 minute to 1 hour), the fixed disk drive and the video shut off while all other devices still operate at full speed. The default setting is Disabled.

Suspend Mode

After the selected period of system inactivity (1 minute to 1 hour), all devices except the CPU shut off. The default setting is Disabled.

HDD Power Down

After the selected period of drive inactivity (1 to 15 minutes), the hard disk drive powers down while all other devices remain active. The default is Disabled.

Throttle Duty Cycle

When the system enters Doze mode, the CPU clock runs only part of the time. You may select the percent of time that the clock runs. The default is 62.5%.

ZZ Active in Suspend

When Enabled, the ZZ signal is active during Suspend mode. The default is Disabled.

PCI/VGA Active Monitor

When Enabled, any video activity restarts the global timer for Standby mode. The default is Enabled.

Soft-Off by PWR-BTTN

When you select Instant Off or Delay 4 Sec., turning the system off with the on / off button places the system in a very low-power-usage state, either immediately or after 4 seconds, with only enough circuitry receiving power to detect power button activity or Resume by Ring activity. The default is Instant-Off.

Power On by Ring

When Enabled, an input signal on the serial Ring Indicator (RI) line (in other words, an incoming call on the modem) awakens the system from a soft off state. The default is Enabled.

IRQ8 Break Suspend

You can select Enabled or Disabled for monitoring of IRQ8 (the Real Time Clock) so it does not awaken the system from Suspend mode. The default is Disabled.

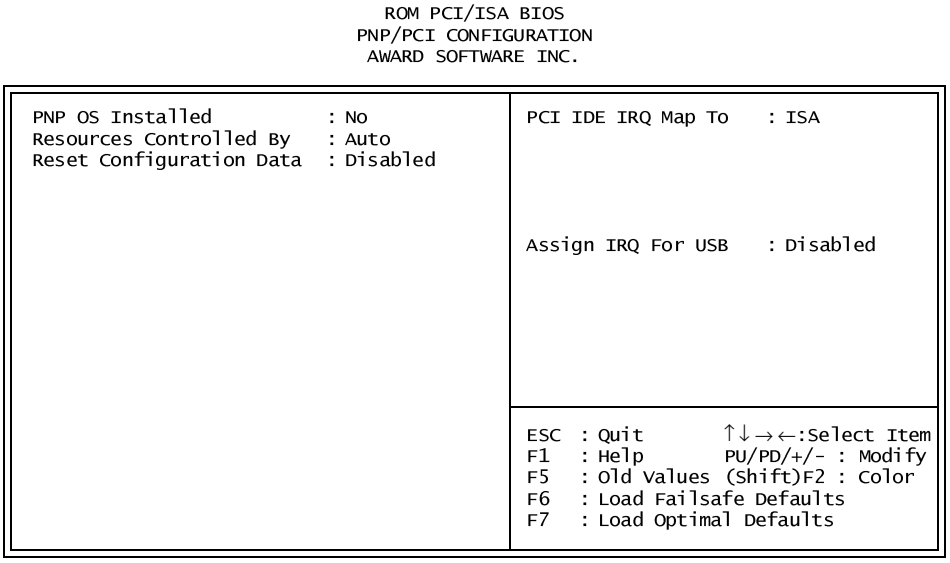
Reload Global Timer Events

When Enabled, an event occurring on each device listed below restarts the global timer for Standby mode.

- IRQ3-7, 9-15, NMI
- Primary IDE 0
- Primary IDE 1
- Secondary IDE 0
- Secondary IDE 1
- Floppy Disk
- Serial Port
- Parallel Port

PnP/PCI Configuration

This section describes the PNP / PCI options available.



PNP OS Installed

Select Yes if the system operating environment is Plug-and-Play aware (e.g., Windows 95). The default is No.

Resources Controlled By

The Plug-and-Play BIOS can automatically configure all the boot and Plug-and-Play compatible devices. If Auto is selected, all the interrupt request (IRQ) and DMA assignment and memory base address fields disappear, as the BIOS automatically assigns them. The default is Manual.

Reset Configuration Data

Select Enabled to reset Extended System Configuration Data (ESCD) when exiting Setup if a new add-on has been installed and the system reconfiguration has caused such a serious conflict that the operating system cannot boot. The default is Disabled.

IRQ *n* Assigned to

When resources are controlled manually, assign each system interrupt as one of the following types, depending on the type of device using the interrupt:

- Legacy ISA Devices compliant with the original PC AT bus specification, requiring a specific interrupt (such as IRQ4 for serial port 1).
- PCI/ISA PnPDevices compliant with the Plug-and-Play standard, whether designed for PCI or ISA bus architecture.

DMA *n* Assigned to

When resources are controlled manually, assign each system DMA channel as one of the following types, depending on the type of device using the interrupt:

- Legacy ISA Devices compliant with the original PC AT bus specification, requiring a specific DMA channel
- PCI/ISA PnPDevices compliant with the Plug-and-Play standard, whether designed for PCI or ISA bus architecture.

PCI IDE IRQ Map to

This field lets you select PCI IDE IRQ mapping or PC AT (ISA) interrupts. If your system does not have one or two PCI IDE connectors on the system board, select values according to the type of IDE interface(s) installed in your system (PCI or ISA). Standard ISA interrupts for IDE channels are IRQ14 for primary and IRQ15 for secondary. The default is PCI-AUTO.

Primary/Secondary IDE INT#

Each PCI peripheral connection is capable of activating up to four interrupts: INT# A, INT# B, INT# C and INT# D. By default, a PCI connection is assigned INT# A. Assigning INT# B has no meaning unless the peripheral device requires two interrupt services rather than just one. Because the PCI/IDE interface in the chipset has two channels, it requires two interrupt services. The primary and secondary IDE INT# fields default to values appropriate for two PCI/IDE channels, with the primary PCI/IDE channel having a lower interrupt than the secondary. The defaults are:

- Primary IDE INT#A
- Secondary IDE INT#B

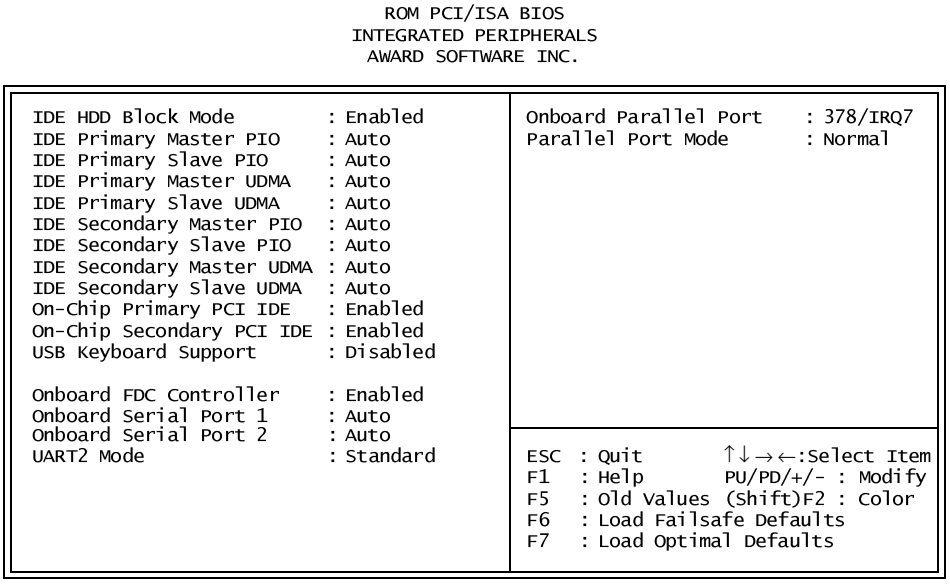
Used Mem base addr

Select a base address for the memory area used by any peripheral that requires high memory. The defaults are:

- Used MEM base addrN/A
- Assign IRQ For USBDisabled

Integrated Peripherals

This section describes the setup for integrated peripherals in the system.



IDE HDD Block Mode

Block mode is also called block transfer, multiple commands, or multiple sector read / write. If the IDE hard drive supports block mode, select Enabled for automatic detection of the optimal number of block read / writes per sector the drive can support. The default is Enabled.

IDE Primary Master PIO

This IDE PIO (Programmed Input / Output) field allows setting a PIO mode (0-4) for the IDE devices that the onboard IDE interface supports. Modes 0 through 4 provide successively increased performance. In Auto mode, the system automatically determines the best mode for each device. The default is Auto

IDE Primary Slave PIO

This IDE PIO (Programmed Input / Output) field allows setting a PIO mode (0-4) for the IDE devices that the onboard IDE interface supports. Modes 0 through 4 provide successively increased performance. In Auto mode, the system automatically determines the best mode for each device. The default is Auto

IDE Primary Master UDMA

Ultra DMA / 33 implementation is possible only if the IDE hard drive supports it and the operating environment includes a DMA driver (Windows 95 OSR2 or a third-party IDE bus master driver). If the hard drive and the operating system both support Ultra DMA / 33, select Auto to enable BIOS support. The default is Auto.

IDE Primary Slave UDMA

Ultra DMA / 33 implementation is possible only if the IDE hard drive supports it and the operating system includes a DMA driver (Windows 95 OSR2 or a third-party IDE bus master driver). If the hard drive and the operating system both support Ultra DMA / 33, select Auto to enable BIOS support. The default is Auto.

IDE Secondary Master PIO

This IDE PIO (Programmed Input / Output) field allows setting a PIO mode (0-4) for the IDE devices that the onboard IDE interface supports. Modes 0 through 4 provide successively increased performance. In Auto mode, the system automatically determines the best mode for each device. The default is Auto

IDE Secondary Slave PIO

This IDE PIO (Programmed Input / Output) field allows setting a PIO mode (0-4) for the IDE devices that the onboard IDE interface supports. Modes 0 through 4 provide successively increased performance. In Auto mode, the system automatically determines the best mode for each device. The default is Auto

IDE Secondary Master UDMA

Ultra DMA / 33 implementation is possible only if the IDE hard drive supports it and the operating environment includes a DMA driver (Windows 95 OSR2 or a third-party IDE bus master driver). If the hard drive and the operating system both support Ultra DMA / 33, select Auto to enable BIOS support. The default is Auto.

IDE Secondary Slave UDMA

Ultra DMA / 33 implementation is possible only if the IDE hard drive supports it and the operating system includes a DMA driver (Windows 95 OSR2 or a third-party IDE bus master driver). If the hard drive and the operating system both support Ultra DMA / 33, select Auto to enable BIOS support. The default is Auto.

On-Chip Primary PCI IDE

The integrated peripheral controller contains an IDE interface with support for two IDE channels. Select Enabled to activate each channel separately. The default is Enabled.

On-Chip Secondary PCI IDE

The integrated peripheral controller contains an IDE interface with support for two IDE channels. Select Enabled to activate each channel separately. When this option is disabled, the IDE Primary Master PIO and IDE Primary Slave PIO options in this screen are removed. The default is Enabled.

USB Keyboard Support

Select Enabled if the operating system contains a Universal Serial Bus (USB) controller and you have a USB keyboard. The default is Enabled.

Onboard FDC Controller

Select Enabled to activate the floppy disk controller (FDC) installed on the system board. If an add-in FDC was installed or the system has no floppy drive, select Disabled in this field. The default is Enabled.

Onboard Serial Port 1/2

Select an address and corresponding interrupt for the first and second serial ports. The default for both ports is Auto.

UART 2 Mode

Select an operating mode for the second serial port. Check the specifications of the infrared port to select the appropriate infrared mode. The options are:

- StandardRs-232C serial port
- IrDA 1.0IrDA-compliant serial infrared port
- MIR 0.57MMid-infrared 0.57 MB port
- MIR 1.15MMid-infrared 1.15 MB port
- FIRFast infrared standard
- ASK IRAmplitude shift keyed infrared port

The default for this setting is Standard.

Onboard Parallel Port

Select an address and corresponding interrupt for the physical parallel (printer) port. The default setting is 378/IRQ7.

Parallel Port Mode

Select an operating mode for the onboard parallel (printer) port. The default setting is Normal.

CHAPTER

E

Device Configuration: I/O and Interrupt Control

Contents

Introduction

This appendix provides the user with the information needed to develop custom applications for the VMIVME-7589A. The CPU board on the VMIVME-7589A is unique in that the BIOS can not be removed; it must be used in the initial boot cycle. A custom application, like a revised operating system for example, can only begin to operate after the BIOS has finished initializing the CPU. The VMIVME-7589A will allow the user to either maintain the current BIOS configuration or alter this configuration to be more user specific, but this alteration can only be accomplished after the initial BIOS boot cycle has completed.

BIOS Operations

When the VMIVME-7589A is powered on, control immediately jumps to the BIOS. The BIOS initiates a Power-on Self-Test (POST) program which instructs the microprocessor to initialize system memory as well as the rest of the system. The BIOS establishes the configuration of all on-board devices by initializing their respective I/O and Memory addresses and interrupt request lines. The BIOS then builds an interrupt vector table in main memory, which is used for interrupt handling. The default interrupt vector table and the default address map is described in Chapter 3 of this manual. Finally, the BIOS jumps to the hard drive or floppy drive to execute the operating system's boot program. This is the point at which a custom operating system could take over control of the board and proceed with a custom configuration and/or custom application. A user application could override the configuration set by the BIOS and reconfigure the system or it could accept what the BIOS initialized.

BIOS Control Overview

There are two areas on the VMIVME-7589A in which the user must be familiar in order to override the initial BIOS configuration. These include the device addresses and the device interrupts. This appendix reviews the details of these addresses and interrupts, and provides a reference list for the individual devices used on the board.

The VMIVME-7589A utilizes the high-performance Peripheral Component Interconnect (PCI) bus along with the Industrial Standard Architecture (ISA) bus. In general, the PCI bus is plug-and-play compatible. The components that are connected to the PCI bus are not always placed at a standard I/O or Memory address, nor are they connected to a standard interrupt request line as is the case with ISA bus devices. These PCI bus devices are re-established by the BIOS meaning that these devices will not always be located at the same address or connected to the same interrupt request line every time the CPU is booted. This appendix lists the defaults that are found by powering up a specific VMIVME-7589A.

Functional Overview

The block diagram included in Figure E-1 on page 129 illustrates the VMIVME-7589A emphasizing the I/O features, including the optional PCI-to-VMEbus bridge.

The circled number in the upper left corner of a function block references the appropriate data book necessary for the programming of the function block.

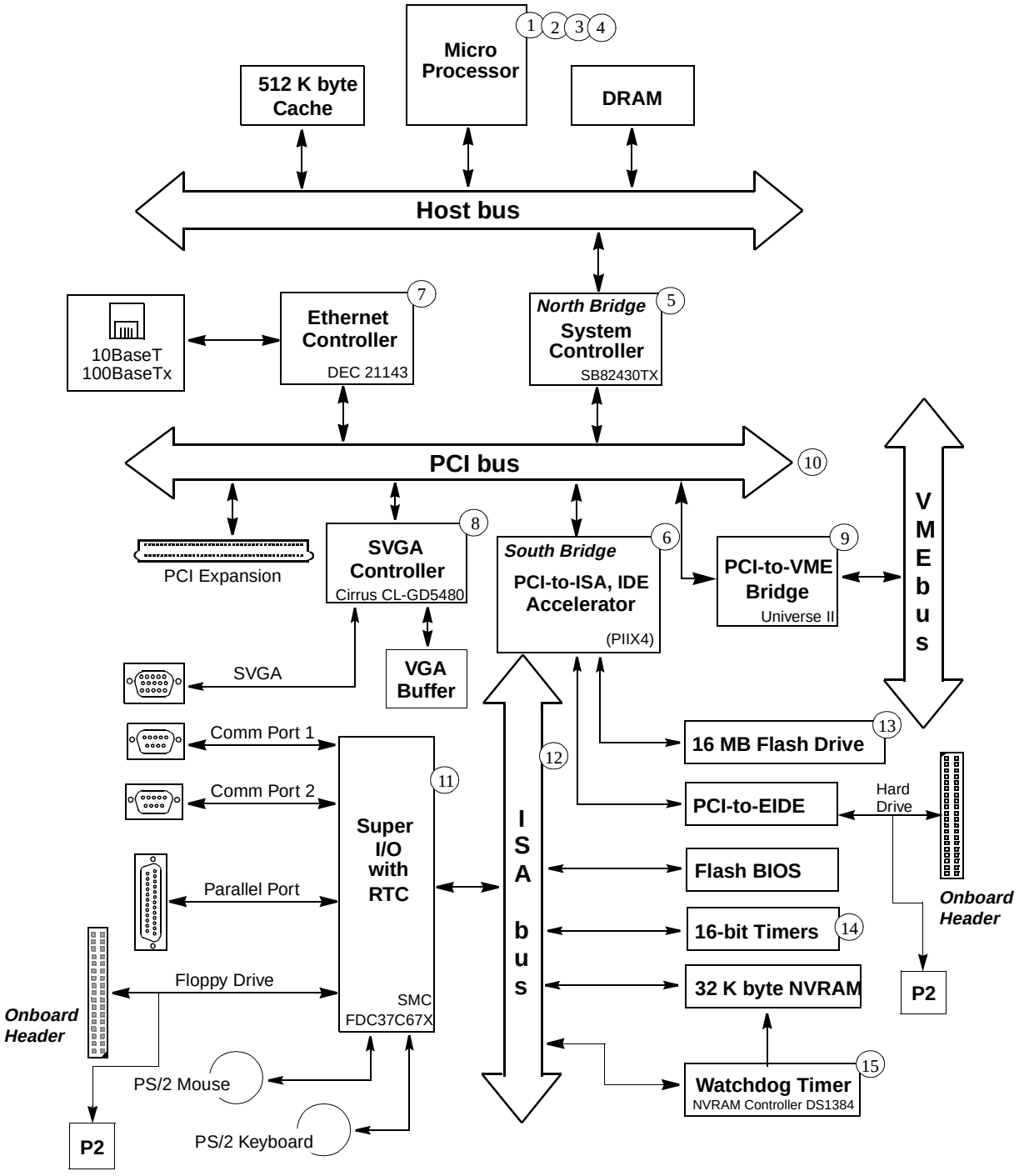


Figure E-1 VMIVME-7589A Block Diagram

Data Book References

1. AMD-K6 Processor with MMX™ Technology
   One AMD Place
   P.O. Box 3453
   Sunnyvale, CA 94088-3453
   (800) 538-8450

2. Pentium and Pentium Pro Processors and Related Products
   Intel Literature Sales
   P.O. Box 7641
   Mt. Prospect, IL 60056-7641
   (800) 548-4752
   www.intel.com

3. Pentium Processor Family Developer's Manual
   Order Number 241428
   Intel Corporation
   2200 Mission College Blvd.
   P.O. Box 58119
   Santa Clara, CA 95052-8119
   (408) 765-8080
   www.intel.com

4. Pentium Processor at 150 MHz, 166 MHz, and 200 MHz
   February 1996, Order Number 242769-003
   Intel Corporation
   2200 Mission College Blvd.
   P.O. Box 58119
   Ssanta Clara, CA 95052-8119
   (408) 765-8080
   www.intel.com

5. Intel 430TX PCISet 82439TX System Controller (MTXC)
   Intel Corporation
   2200 Mission College Boulevard
   P.O. Box 58119
   Santa Clara, CA 95052-8119

6. Intel PIIX4
   82371AB PCI ISA IDE Xcellerator (PIIX4)
   2200 Mission College Boulevard
   P.O. Box 58119
   Santa Clara, CA 95052-8119

7. Digital Semiconductor 21143 10/100 Mb/s Ethernet LAN Controller
   Digital Equipment Corporation
   Maynard, Mass.
   1-800-332-2717
   www.digital.com

8. Cirrus Logic - CL-GD5480
Technical Reference Manual
Cirrus Logic
3100 West Warren Avenue
Fremont, CA 94538
(510) 623-8300

9. VMIVME-7589A User Manual
500-007589A-000 Product Manual
500-007589A-001 VMIVME-7589A, Tundra Universe II-Based VMEbus Interface Option Product Manual

10. PCI Local Bus Specification, Rev. 2.1
PCI Special Interest Group
P.O. Box 14070
Portland, OR 97214
(800) 433-5177 (U.S.)
(503) 797-4207 (International)
(503) 234-6762 (FAX)

11. SMC FDC37C67X Enhanced Super I/O Controller
SMC Component Products Division
300 Kennedy Drive
Hauppauge, NY 11788
(516) 435-6000
(516) 231-6004 (FAX)

12. ISA & EISA, Theory and Operation
Solari, Edward
Annabooks
15010 Avenue of Science, Suite 101
San Diego, CA 92128 USA
ISBN 0-929392 -15-9

13. Flash ChipSet Product Manual
SanDisk Corporation
140 Caspian Court
Sunnyvale, CA 94089-9820

14. 82C54 CHMOS Programmable Internal Timer
Intel Corporation
2200 Mission College Boulevard
P.O. Box 58119
Santa Clara, CA 95052-8119

15. DS 1384 Watchdog Timekeeping Controller
Dallas Semiconductor
4461 South Beltwood Pwky
Dallas, TX 75244-3292

Device Address Definition

The standard PC / AT architecture defines two distinctive types of address spaces for the devices and peripherals on the CPU board. These spaces have typically been named Memory address space and I / O address space. The boundaries for these areas are limited to the number of address bus lines that are physically located on the CPU board. The VMIVME-7589A has 32 address bus lines located on the board, thereby defining the limit of the address space as 4 Gbyte. The standard PC / AT architecture defines Memory address space from zero to 4 Gbyte and the separate I / O address space from zero to 64 Kbyte.

ISA Devices

The ISA devices on the VMIVME-7589A are configured by the BIOS at boot-up and fall under the realm of the standard PC / AT architecture. They are mapped in I / O address space within standard addresses and their interrupts are mapped to standard interrupt control registers. However, all of the ISA devices with the exception of the real-time clock, keyboard, and programmable timer are relocatable to almost anywhere within the standard 1 Kbyte of I / O address space. Table E-3 defines the spectrum of addresses available for reconfiguration of ISA devices.

As previously stated, in the standard PC / AT system, all I / O devices are mapped in I / O address space; however, one exception exists. The Dynamic Random Access Memory (DRAM), Battery-Backed SRAM, and Watchdog Timer are addressed in Memory address space. The BIOS places DRAM at address zero and extends to the physical limit of the on-board DRAM.

Table E-1 ISA Device Mapping Configuration

Device	Memory Space	I/O Address Space	PIC Interrupt Options	Byte Address Boundary	Default
Floppy	N / A	[0x100 - 0xFF8]	IRQ1 - IRQ15	8	$3F0
Parallel Port	N / A	[0x100 - 0xFFC] [0x100 - 0xFF8]	IRQ1 - IRQ15	4 8	$378
Serial Port 1	N / A	[0x100 - 0xFF8]	IRQ1 - IRQ15	8	$3F8
Serial Port 2	N / A	[0x100 - 0xFF8]	IRQ1 - IRQ15	8	$2F8
Real-Time Clock	Nonrelocatable				$070
Keyboard	Nonrelocatable				$060

Table E-1 ISA Device Mapping Configuration (Continued)

Device	Memory Space	I/O Address Space	PIC Interrupt Options	Byte Address Boundary	Default
Auxiliary I/O	N/A	- Primary I/O [0x000 - 0xFFF] - Secondary I/O [0x000 - 0xFFF]	IRQ1, IRQ3-IRQ15	1 1	
Programmable Timer		Nonrelocatable 0X500	IRQ5	4	0X500
Watchdog Timer	Nonrelocatable 0XD8000	N/A	N/A	0XD	0XD8000
Battery-Backed SRAM	Nonrelocatable 0XD800E	N/A	N/A	(32K-0XD)	0XD800E

PCI Devices

PCI devices are fully configured under I/O and/or Memory address space. Table E-3 describes the PCI bus devices that are on-board the VMIVME-7589A along with each device's configuration spectrum.

The PCI bus includes three physical address spaces. As with ISA bus, PCI bus supports Memory and I/O address space, but PCI bus includes an additional Configuration address space. This address space is defined to support PCI bus hardware configuration (refer to the PCI bus Specification for complete details on the configuration address space). PCI bus targets are required to implement Base Address registers in configuration address space to access internal registers or functions. The BIOS uses the Base Address register to determine how much space a device requires in a given address space and then assigns where in that space the device will reside. This functionality enables PCI devices to be located in either Memory or I/O address space.

Table E-2 PCI Device Mapping Configuration

Device	Memory Space	I/O Address Space	PIC Interrupt Options
SVGA	Anywhere	N/A	N/A
Universe II** PCI-to-VMEbus Bridge	Anywhere	N/A	PCI Defined
Ethernet	Anywhere	Anywhere	PCI Defined
Timer Interrupt Registers	N/A	Fixed	N/A

** *Refer to the VMIVME-7589A-001 User's Manual.*

Device Interrupt Definition

PC/AT Interrupt Definition

The interrupt hardware implementation on the VMIVME-7589A is standard for computers built around the PC / AT architecture. The PC / AT evolved from the IBM PC / XT architecture. In the IBM PC / XT systems, only eight Interrupt Request (IRQ) lines exist, numbered from IRQ0 to IRQ7. These interrupt lines were included originally on a 8259A Priority Interrupt Controller (PIC) chip.

The IBM PC / AT computer added eight more IRQx lines, numbered IRQ8 to IRQ15, by cascading a second slave 8259A PIC into the original master 8259A PIC. The interrupt line IRQ2 at the master PIC was committed as the cascade input from the slave PIC. This master / slave architecture, the standard PC / AT interrupt mapping, is illustrated in Figure E-2 on page 135 within the PCI-to-ISA Bridge PIIX4 82371AB section of the diagram.

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 on the AT Expansion Bus (or ISA bus).

The BIOS defines the PC / AT interrupt line to be used by each device. The BIOS writes to each of the two cascaded 8259A PIC chips an 8-bit vector which maps each IRQx to its corresponding interrupt vector in memory.

ISA Device Interrupt Map

The VMIVME-7589A BIOS maps the IRQx lines to the appropriate device per the standard ISA architecture. Reference Figure E-2 on page 135. This initialization operation cannot be changed; however, a custom application could reroute the interrupt configuration after the BIOS has completed the initial configuration cycle.

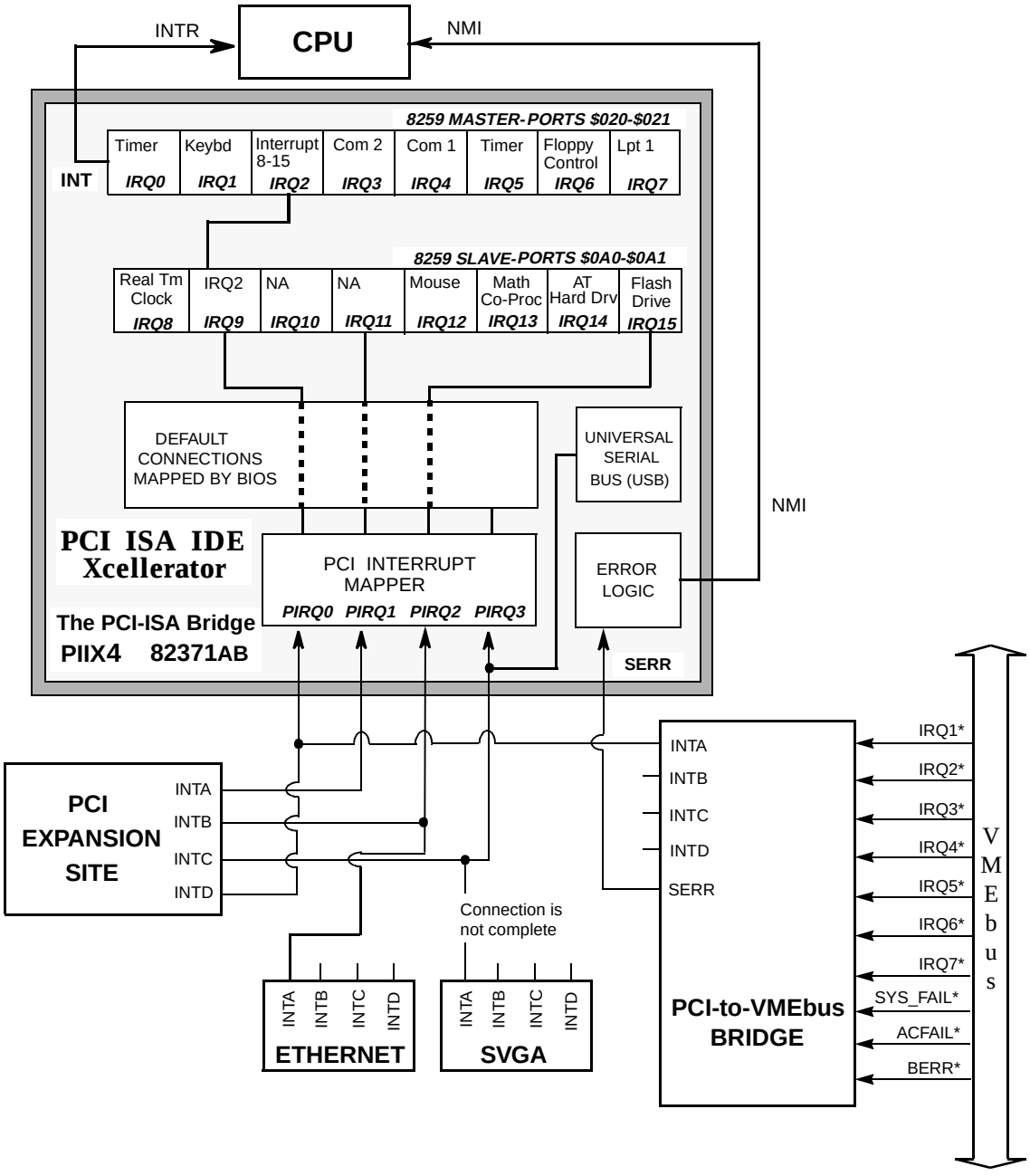


Figure E-2 BIOS Default Connections for the PC Interrupt Logic Controller

PCI Device Interrupt Map

The PCI bus-based external devices include the PCI expansion site, the PCI-to-VMEbus bridge, and the VGA reserved connection. The default BIOS maps these external devices to the PCI Interrupt Request (PIRQx) lines of the PIIX4. This mapping is illustrated in Figure E-2 on page 135 and is defined in Table E-3.

The device PCI interrupt lines (INTA through INTD) that are present on each device *cannot* be modified.

Table E-3 Device PCI Interrupt Mapping by the BIOS

DEVICE	COMPONENT	VENDOR ID	DEVICE ID	CPU ADDRESS MAP ID SELECT	DEVICE PCI INTERRUPT	MOTHER-BOARD PCI INTERRUPT MAPPER	DATA BOOK REF. #	REVISION ID
PCI-to-VME Bridge Option: Tundra Universe II	Universe CA91C142	0x10E3	0x0	AD19	INTA	PIRQ0	9	N / A
PCI Bus Master I/O Controller	PLX PCI9060ES	0x114A	0x0001	AD20	N / A	N / A	9	N / A
PCI Expansion Site	N / A	Board Specific	Board Specific	N / A	INTA	PIRQ1	N / A	N / A
	N / A	Board Specific	Board Specific	N / A	INTB	PIRQ2	N / A	N / A
	N / A	Board Specific	Board Specific	N / A	INTC	PIRQ3	N / A	N / A
	N / A	Board Specific	Board Specific	N / A	INTD	PIRQ0	N / A	N / A
Timer INT Registers	PIIX4 82371AB Function	0x8086	0x7113	AD18	N / A	N / A	6	N / A
PCI-to-ISA Bridge	PIIX4 82371AB Function 00	0x8086	0x7000	AD18	N / A	N / A	6	N / A
SVGA Controller	Cirrus CL-GD5480	0x1013		AD25	INTA**	PIRQ3**	8	N / A
Ethernet Controller	DEC 21143	0x1011	0x0019	AD22	INTA	PIRQ2	7	N / A
PCI IDE Controller	PIIX4 82371AB Function 01	0x8086	0x7111	AD18	N / A	N / A	6	N / A
Universal Serial Bus (USB)***	PIIX4 82371AB Function 02	0x8086	0x7020	AD18	INTD	PIRQ3	6	N / A
PCI Host Bridge	Intel 430 TX	0x8086	0x7100	N / A	N / A	N / A	5	N / A

* To access these parts, use the revision number as the distinguishing factor.
** Not connected, for reference only
*** PIRQ3 interrupt is not enabled by the BIOS.

The motherboard accepts these PCI device interrupts through the PCI interrupt mapper function. The BIOS default maps the PCI Interrupt Request (PIRQx) external device lines to one of the available slave PIC Interrupt Request lines, IRQ (9, 10, or 11). The BIOS default mapping of the PIRQx to the slave PIC is defined in Table E-4.

Table E-4 Default PIRQx to IRQx BIOS Mapping

PCI INTx	PIC IRQx
PIRQ0	IRQ11
PIRQ1	IRQ10
PIRQ2	IRQ9
PIRQ3	No Connection

Using the interrupt steering registers of the 82371AB PIIX4, the user can override the BIOS defaults and map any of the PCI interrupts (PIRQ0-3) to any of the following PIC IRQx (ISA) interrupts: IRQ14, 12-9, or 7-3.

If PCI interrupts are remapped by the user, care must be taken to ensure that all ISA and PCI functions that require an interrupt are included.

Appendix

F

Sample C Software

Contents

Introduction

This appendix provides listings of a library of sample code that the programmer may utilize to build applications. These files are provided to the VMIVME-7589A user on disk 320-500023-010, Sample Application C Code for the VMIVME-7589A, included in the distribution disk set.

Because of the wide variety of environments in which the VMIVME-7589A operates, the samples provided in this appendix are not necessarily intended to be verbatim boilerplates. Rather, they are intended to give the end user an example of the standard structure of the operating code.

Directory PCVME

This directory contains listings of the source code used to generate an application that sets up the PC to VME interface in three segments using A16 addressing, A24 addressing, or A32 addressing.

** FILE: FLAT.C

```
**
*/

#include <stdio.h>
#include <dos.h>

#include "flat.h"

/*
** Keyboard controller defines
*/

#define RAMPORT    0x70
#define KB_PORT    0x64
#define PCNMIPORT  0xA0

#define INBA20     0x60
#define INBA20ON   0xDF
#define INBA20OFF  0xDD

/*
** macro to clear keyboard port
*/

#define kx() { while( inp( KB_PORT ) & 2 ); }
```

```
/*
**  define GDT pointer structure
*/

static struct fword gdtptr;         /* fword ptr to gdt */

/*
**  A20: Enable A20 line
**
**  flag: enable  = 1
**        disable = 0
*/

void a20( int flag )
{
  kx();
  outp( KB_PORT, 0xD1 );
  kx();
  outp( INBA20, flag ? INBA20ON : INBA20OFF );
  kx();
  outp( KB_PORT, 0xFF );
  kx();
}

/*
**  convert a linear address to a far pointer
*/

void far * linear_to_seg( FPTR lin )
{
   void far *p;
   FP_SEG(p) = (unsigned int)( lin >> 4 );
```

```
  FP_OFF(p) = (unsigned int)( lin & 0xF );
  return p;
}

/*
** Adjust the GS register's limit to 4GB
**
** Note: interrupts are enabled by this call.
*/

void extend_seg( void )
{
  /*
  ** compute linear address and limit of GDT
  */
  gdtptr.linear_add = seg_to_linear(( void far * ) GDT );
  gdtptr.limit = 15;
  /*
  ** disable regular interrupts
  */
  disable();
  /*
  ** disable NMI
  */
  outp( RAMPORT, inp( RAMPORT ) | 0x80 );
  /*
  ** call protected mode code
  */
  protinit( &gdtptr );
  /*
  ** Turn interrupts back on
  */
```

```
 enable();
 /*
 ** Turn NMI back on
 */
 outp( RAMPORT, inp( RAMPORT ) & 0x7F );
}

void protinit( struct fword * address )
{
 asm {
  .386P
  push ds
  lds  bx,address
  lgdt FWORD ptr [bx]
  pop  ds
  mov  eax,cr0
  or   al,0x01
  mov  cr0,eax
  jmp  short nxt
 }
 nxt:
 asm {
  .386P
  mov  bx,8
  mov  gs,bx
  mov  es,bx
  and  al,0xfe
  mov  cr0,eax
 }
}

int fr_byte( FPTR adr )
```

```
{
 int d;
 asm {
  .386P
  xor   ax,ax      /* zero gs */
  mov   gs,ax
  mov   eax, adr
  mov   al,byte ptr gs:[eax]
  mov   d,ax
 }
 return d;
}

int fr_word( FPTR adr )
{
 int d;
 asm {
  .386P
  xor   ax,ax      /* zero gs */
  mov   gs,ax
  mov   eax, adr
  mov   ax,word ptr gs:[eax]
  mov   d,ax
 }
 return d;
}

long fr_long( FPTR adr )
{
 long  d;
 asm {
  .386P
```

```
   xor   ax,ax      /* zero gs */
   mov   gs,ax
   mov   eax,adr
   mov   eax,dword ptr gs:[eax]
   mov   d,eax
  }
 return d;
}

void fw_byte( FPTR a, int d )
{
 asm {
   .386P
   xor   ax,ax      /* zero gs */
   mov   gs,ax
   mov   eax,a
   mov   bx,d
   mov   byte ptr gs:[eax],bl
  }
}

void fw_word( FPTR a, int d )
{
 asm {
   .386P
   xor   ax,ax      /* zero gs */
   mov   gs,ax
   mov   eax,a
   mov   bx,d
   mov   word ptr gs:[eax],bx
  }
}
```

```
void fw_long( FPTR a, long d )
{
 asm {
  .386P
  xor  ax,ax      /* zero gs */
  mov  gs,ax
  mov  eax,a
  mov  ebx,d
  mov  dword ptr gs:[eax],ebx
 }
}

 /*
```

** FILE: PCI.C

```
**
*/

#include <dos.h>
#include <stddef.h>
#include "pci.h"

#define HIGH_BYTE(ax) (ax >> 8)
#define LOW_BYTE(ax) (ax & 0xff)


int find_pci_device(unsigned short device_id,
        unsigned short vendor_id,
        unsigned short index,
        unsigned char *bus_number,
        unsigned char *device_and_function)
{
  int ret_status;
  unsigned short ax, bx, flags;

  _CX = device_id;
  _DX = vendor_id;
  _SI = index;
  _AH = PCI_FUNCTION_ID;
  _AL = FIND_PCI_DEVICE;

  geninterrupt(0x1a);

  ax = _AX;
  bx = _BX;
  flags = _FLAGS;
```

```
  if ((flags & CARRY_FLAG) == 0)
  {
   ret_status = HIGH_BYTE(ax);
   if (ret_status == SUCCESSFUL)
   {
    if (bus_number != NULL) *bus_number = HIGH_BYTE(bx);
    if (device_and_function != NULL) *device_and_function = LOW_BYTE(bx);
   }
  }
  else
  {
   ret_status = NOT_SUCCESSFUL;
  }
  return(ret_status);
}


int read_configuration_area(unsigned char function,
                            unsigned char bus_number,
                            unsigned char device_and_function,
                            unsigned char register_number,
                            unsigned long *data)
{
  int ret_status;
  unsigned short ax, flags;
  unsigned long ecx;

  _BH = bus_number;
  _BL = device_and_function;
  _DI = register_number;
  _AH = PCI_FUNCTION_ID;
```

```
  _AL = function;

  geninterrupt(0x1a);

  ecx = _ECX;
  ax = _AX;
  flags = _FLAGS;

  if ((flags & CARRY_FLAG) == 0)
  {
   ret_status = HIGH_BYTE(ax);
   if (ret_status == SUCCESSFUL)
   {
    *data = ecx;
   }
  }
  else
  {
   ret_status = NOT_SUCCESSFUL;
  }
  return(ret_status);
}


int write_configuration_area(unsigned char function,
                unsigned char bus_number,
                unsigned char device_and_function,
                unsigned char register_number,
                unsigned long value)
{
  int ret_status;
  unsigned short ax, flags;
```

```
  _BH = bus_number;
  _BL = device_and_function;
  _ECX = value;
  _DI = register_number;
  _AH = PCI_FUNCTION_ID;
  _AL = function;

  geninterrupt(0x1a);

  ax = _AX;
  flags = _FLAGS;

  if ((flags & CARRY_FLAG) == 0)
  {
    ret_status = HIGH_BYTE(ax);
  }
  else
  {
    ret_status = NOT_SUCCESSFUL;
  }
  return(ret_status);
}

/*
```

** FILE: PCVME.C

```
**
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <dos.h>

#include "flat.h"
#include "pci.h"
#include "sysregs.h"
#include "pc.h"


#define  PCI_BASE16  0x10000000   /* PCI BASE for A16 */
#define  PCI_BASE24  0x10020000   /* PCI BASE for A24 */
#define  PCI_BASE32  0x20000000   /* PCI BASE for A32 */

/* function prototypes */
void far interrupt (* old_break)( void );
void far interrupt on_break( void );

/* global variables */
unsigned long vme_offset;
unsigned char bus, dev_func;
FPTR un_regs;
FPTR plx_regs;

void main( void )
```

```
{
 unsigned char pci_devices;
 int test_int;
 unsigned long temp_dword;

 old_break = getvect( 0x23 );
 setvect( 0x23, on_break );
 extend_seg();
 a20( 1 );


 pci_devices = 0;
 /* locate the UNIVERSE device on the PCI bus */
 test_int = find_pci_device(UNIVERSE_DID, UNIVERSE_VID, 0,
            &bus, &dev_func);
 if(test_int == SUCCESSFUL)
 {
  pci_devices |= PCI_UNIVERSE;
  test_int = read_configuration_area( READ_CONFIG_DWORD, bus, dev_func,
                              0x10, &temp_dword );

  if(test_int == SUCCESSFUL)
  {
    un_regs = (FPTR) temp_dword;
  }
 }

 /* locate the PLX device on the PCI bus */
 test_int = find_pci_device(PLX_DID, PLX_VID, 0,
            &bus, &dev_func);
 if(test_int == SUCCESSFUL)
 {
```

```
  pci_devices |= PCI_PLX;
  test_int = read_configuration_area( READ_CONFIG_DWORD, bus, dev_func,
                                0x18, &temp_dword );

  if(test_int == SUCCESSFUL)
  {
    plx_regs = (FPTR) temp_dword;
  }
}

if( !(pci_devices & PCI_UNIVERSE) ) {
  printf("Unable to locate PCI device Tundra Universe\n");
}

if( !(pci_devices & PCI_PLX) ) {
  printf("Unable to locate PCI device PLX\n");
}

if( pci_devices != (PCI_UNIVERSE | PCI_PLX) ) exit( 1 );

 /* setup VME windows to VME base address 0000 Non-Privileged */
 fw_long( un_regs + 0x104, PCI_BASE16 ); /* pci base for A16 */
 fw_long( un_regs + 0x108, (PCI_BASE16 + 0x10000) ); /* 64K window */
 fw_long( un_regs + 0x10C, (0x00000000 - PCI_BASE16) ); /* vme translation */
 fw_long( un_regs + 0x100, (LSI_CTL_EN | LSI_CTL_VDW_32 | LSI_CTL_VAS_16)
);

 fw_long( un_regs + 0x118, PCI_BASE24 ); /* pci base for A24 */
 fw_long( un_regs + 0x11C, (PCI_BASE24 + 0x1000000) ); /* 16 Meg window */
 fw_long( un_regs + 0x120, (0x00000000 - PCI_BASE24) ); /* vme translation */
 fw_long( un_regs + 0x114, (LSI_CTL_EN | LSI_CTL_VDW_32 | LSI_CTL_VAS_24) );
```

```
  fw_long( un_regs + 0x12C, PCI_BASE32 ); /* pci base for A32 */
  fw_long( un_regs + 0x130, (PCI_BASE32 + 0x1000000) ); /* 16 Meg window */
  fw_long( un_regs + 0x134, (0x00000000 - PCI_BASE32) ); /* vme translation */
  fw_long( un_regs + 0x128, (LSI_CTL_EN | LSI_CTL_VDW_32 | LSI_CTL_VAS_32)
);

  /* enable VME and Master Endian Conversion */
  fw_word( plx_regs, (VME_EN | MEC) );

  /* use the flat write / flat read functions to access VME */
  /* EX: ch_data = (unsigned char)(fr_byte( PCI_BASE16 + vme_offset )); */
  /* EX: fw_byte( PCI_BASE16 + vme_offset, ch_dat ); */

} /* end main */


void interrupt on_break( void )
{
  /* do nothing */
}


/*
```

** FILE: FLAT.H

```
**
** Prototypes typedefs and macros for flat memory access
**
*/

typedef unsigned long FPTR;

/*
** Global descriptor table
*/

struct _GDT {
  unsigned int limit;
  unsigned int base;
  unsigned int access;
  unsigned int hi_limit;
};


static struct _GDT GDT[2]= {
  {0,0,0,0},                    /* Null selector slot */
  {0xFFFF,0,0x9200,0x8F}        /* 4 Gig data segment */
};

/*
** FWORD pointer to GDT
*/

struct fword {
  unsigned int limit;
  unsigned long linear_add;
```

```
};

/*
** convert segmented address to linear address
*/

#define seg_to_linear(fp) (((FPTR) FP_SEG(fp)<<4)+FP_OFF(fp))

/*
** flat memory function prototypes
*/

void a20( int );
void far * linear_to_seg( FPTR );
void extend_seg( void );
void protinit( struct fword * );

int  fr_byte( FPTR );
void fw_byte( FPTR, int );

int  fr_word( FPTR );
void fw_word( FPTR, int );

long fr_long( FPTR );
void fw_long( FPTR, long );

/*
```

** FILE: PC.H

```
**
*/

#define UNIVERSE_VID    0x10E3
#define UNIVERSE_DID    0x0000
#define PLX_VID         0x114A
#define PLX_DID         0x0001

#define PCI_UNIVERSE    0x01
#define PCI_PLX         0x02

/* lsi[X]_ctl - slave image control registers ( lsi0 - lsi7 ) */
#define    LSI_CTL_EN     0x80000000  /* R/W image enable                */
#define    LSI_CTL_PWEN   0x40000000  /* R/W posted write enable         */
#define    LSI_CTL_VDW_08 0x00000000  /* R/W VMEbus maximum data width
D08  */
#define    LSI_CTL_VDW_16 0x00400000  /* R/W VMEbus maximum data width
D16  */
#define    LSI_CTL_VDW_32 0x00800000  /* R/W VMEbus maximum data width
D32  */
#define    LSI_CTL_VDW_64 0x00C00000  /* R/W VMEbus maximum data width
D64  */
#define    LSI_CTL_VAS_16 0x00000000  /* R/W VMEbus address space A16      */
#define    LSI_CTL_VAS_24 0x00010000  /* R/W VMEbus address space A24      */
#define    LSI_CTL_VAS_32 0x00020000  /* R/W VMEbus address space A32      */
#define    LSI_CTL_VAS_R1 0x00030000  /* R/W VMEbus address space RSVD1    */
#define    LSI_CTL_VAS_R2 0x00040000  /* R/W VMEbus address space RSVD2    */
#define    LSI_CTL_VAS_CR 0x00050000  /* R/W VMEbus address space CR/CSR
*/
#define    LSI_CTL_VAS_U1 0x00060000  /* R/W VMEbus address space USER1    */
#define    LSI_CTL_VAS_U2 0x00070000  /* R/W VMEbus address space USER2    */
#define    LSI_CTL_PGM_D  0x00000000  /* R/W VMEbus data AM code          */
```

```
#define    LSI_CTL_PGM_P  0x00004000  /* R/W VMEbus program AM code        */
#define    LSI_CTL_SUPER  0x00001000  /* R/W VMEbus supervisory AM code    */
#define    LSI_CTL_VCT_S  0x00000000  /* R/W VMEbus single cycles only     */
#define    LSI_CTL_VCT_SB 0x00000100  /* R/W VMEbus single cycles and block */
#define    LSI_CTL_LAS_M  0x00000000  /* R/W PCIbus memory space           */
#define    LSI_CTL_LAS_IO 0x00000001  /* R/W PCIbus I/O space              */
#define    LSI_CTL_LAS_C  0x00000002  /* R/W PCIbus type 1 config space    */
#define    LSI_CTL_LAS_R  0x00000003  /* R/W PCIbus reserved               */


/*
```

** FILE: PCI.H

```
**
*/

#define TRUE    1
#define FALSE0

#define CARRY_FLAG 0x01

/* PCI Functions */
#define PCI_FUNCTION_ID        0xB1
#define PCI_BIOS_PRESENT0x01
#define FIND_PCI_DEVICE0x02
#define FIND_PCI_CLASS_CODE0x03
#define READ_CONFIG_BYTE0x08
#define READ_CONFIG_WORD0x09
#define READ_CONFIG_DWORD      0x0A
#define WRITE_CONFIG_BYTE      0x0B
#define WRITE_CONFIG_WORD      0x0C
#define WRITE_CONFIG_DWORD     0x0D

/* PCI Return codes */

#define SUCCESSFUL0x00
#define NOT_SUCCESSFUL0x01

/* Prototypes */

int find_pci_device(unsigned short device_id,
                    unsigned short vendor_id,
                    unsigned short index,
                    unsigned char *bus_number,
```

```
                unsigned char *device_and_function);

int read_configuration_area(unsigned char function,
                unsigned char bus_number,
                unsigned char device_and_function,
                unsigned char register_number,
                unsigned long *data);

int write_configuration_area(unsigned char function,
                unsigned char bus_number,
                unsigned char device_and_function,
                unsigned char register_number,
                unsigned long value);

/*
```

** FILE: SYSREGS.H

```
**
*/

/* Command register */
#define    MEC     0x0001     /* Master endian conversion      */
#define    SEC     0x0002     /* Slave endian conversion       */
#define    ABLE    0x0004     /* Auxillary BERR logic enable   */
#define    BTO     0x0008     /* Auxillary BTO enable          */
#define    BTOV0   0x0010     /* Auxillary BTO value lsb       */
#define    BTOV1   0x0020     /* Auxillary BTO value msb       */
#define    BERRIM  0x0040     /* Auxillary BERR interrupt map  */
#define    MB_MSK0 0x0080     /* Mail-box 0 interrupt mask     */
#define    MB_MSK1 0x0100     /* Mail-box 1 interrupt mask     */
#define    MB_MSK2 0x0200     /* Mail-box 2 interrupt mask     */
#define    MB_MSK3 0x0400     /* Mail-box 3 interrupt mask     */
#define    VME_EN  0x0800     /* VMEbus interface enable       */

/* BERR status register */
#define    BERR_S  0x01       /* BERR status                   */

/* BERR mask register */
#define    BERR_M  0x01       /* BERR mask                     */

/* Mail Box 0 register */
#define    MB0_INT 0x01       /* MB0 interrupt bit             */
```

Directory Timers

This directory contains sample code useful in the creation of applications involving the VMIVME-7589A's three software controlled 16-bit timers. The code is written for the control of a single timer, but can be utilized in generating code for any timer configuration. The timers are described in Chapter 4 of the manual.

** FILE: PCI.C

```
**
*/

#include <dos.h>
#include <stddef.h>
#include "pci.h"

#define HIGH_BYTE(ax) (ax >> 8)
#define LOW_BYTE(ax) (ax & 0xff)



int find_pci_device(unsigned short device_id,
        unsigned short vendor_id,
        unsigned short index,
        unsigned char *bus_number,
        unsigned char *device_and_function)
{
  int ret_status;
  unsigned short ax, bx, flags;

  _CX = device_id;
  _DX = vendor_id;
  _SI = index;
  _AH = PCI_FUNCTION_ID;
  _AL = FIND_PCI_DEVICE;
```

```
  geninterrupt(0x1a);

  ax = _AX;
  bx = _BX;
  flags = _FLAGS;

  if ((flags & CARRY_FLAG) == 0)
  {
   ret_status = HIGH_BYTE(ax);
   if (ret_status == SUCCESSFUL)
   {
    if (bus_number != NULL) *bus_number = HIGH_BYTE(bx);
    if (device_and_function != NULL) *device_and_function = LOW_BYTE(bx);
   }
  }
  else
  {
   ret_status = NOT_SUCCESSFUL;
  }
  return(ret_status);
}


int read_configuration_area(unsigned char function,
                            unsigned char bus_number,
                            unsigned char device_and_function,
                            unsigned char register_number,
                            unsigned long *data)
{
  int ret_status;
  unsigned short ax, flags;
```

```
  unsigned long ecx;

  _BH = bus_number;
  _BL = device_and_function;
  _DI = register_number;
  _AH = PCI_FUNCTION_ID;
  _AL = function;

  geninterrupt(0x1a);

  ecx = _ECX;
  ax = _AX;
  flags = _FLAGS;

  if ((flags & CARRY_FLAG) == 0)
  {
   ret_status = HIGH_BYTE(ax);
   if (ret_status == SUCCESSFUL)
   {
    *data = ecx;
   }
  }
  else
  {
   ret_status = NOT_SUCCESSFUL;
  }
  return(ret_status);
}


int write_configuration_area(unsigned char function,
            unsigned char bus_number,
```

```
                unsigned char device_and_function,
                unsigned char register_number,
                unsigned long value)
{
  int ret_status;
  unsigned short ax, flags;

  _BH = bus_number;
  _BL = device_and_function;
  _ECX = value;
  _DI = register_number;
  _AH = PCI_FUNCTION_ID;
  _AL = function;

  geninterrupt(0x1a);

  ax = _AX;
  flags = _FLAGS;

  if ((flags & CARRY_FLAG) == 0)
  {
    ret_status = HIGH_BYTE(ax);
  }
  else
  {
    ret_status = NOT_SUCCESSFUL;
  }
  return(ret_status);
}

/*
```

** FILE: TIMERS.C

```
**
*/

#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <ctype.h>
#include <conio.h>

#include "t7589At.h"

/* function prototypes */
void far interrupt irq_rcvd( void );
void init_timer_int( void );
void restore_orig_int( void );
void load_counter( int, unsigned int );
void read_counter( int, unsigned int *, unsigned char * );

/* global variables */
extern unsigned long t1_count;    /* timer 1 count */
extern unsigned long t2_count;    /* timer 2 count */
extern unsigned long t3_count;    /* timer 3 count */
extern unsigned char tmr_status;
extern unsigned int gpi_base;
extern unsigned int gpo_base;
extern unsigned int timer_base;
extern unsigned char pic1_org;
extern unsigned char gpo_org;

void far interrupt (* old_vect)(void);
```

```
/*****************************************************************/
/*  init_timer_int()                                             */
/*                                                               */
/*  purpose: Using the interrupt assigned, the original vector is  */
/*           saved and the vector to the new ISR is installed. The */
/*           programmable-interrupt-controller (PIC) is enabled.   */
/*                                                               */
/*****************************************************************/
/*  parameters: none                                             */
/*****************************************************************/
/*  return value: none                                           */
/*****************************************************************/
void init_timer_int( void )
{
 disable();
 old_vect = getvect( IRQ5 ); /* save vector for IRQ5 */
 setvect( IRQ5, irq_rcvd );

 /* enable interrupt 5 */
 outp(0x21, (pic1_org & 0xDF) ); /* 0 = enable 1 = disable */

 /* clear all three GPO inputs */
 outp( gpo_base, ( gpo_org & GPO_CLR ) );

 /* set all three GPO outputs to 1 to allow int status registers to function */
 outp( gpo_base, (gpo_org | GPO_T1 | GPO_T2 | GPO_T3 ) );

 enable();

} /* init_timer_int */

/*****************************************************************/
```

```
/*  restore_orig_int()                                  */
/*                                                      */
/*  purpose: Using the interrupt assigned, the original vector is  */
/*           restored and the programmable-interrupt-controller    */
/*           is disabled.                               */
/*                                                      */
/*********************************************************************/
/*  parameters: none                                    */
/*********************************************************************/
/*  return value: none                                  */
/*********************************************************************/
void restore_orig_int( void )
{
 disable();

 outp(0x21, pic1_org);

 setvect( IRQ5, old_vect );

 enable();

} /* restore_orig_int */

/*********************************************************************/
/*  load_counter()                                      */
/*                                                      */
/*  purpose: Loads the appropriate counter with the count passed   */
/*                                                      */
/*                                                      */
/*********************************************************************/
/*  parameters: int counter = 1, 2, 3 for COUNTER 1, 2, or 3      */
/*              unsigned int count = count to be loaded          */
```

```
/*****************************************************************/
/*  return value: none                                           */
/*****************************************************************/
void load_counter( int counter, unsigned int count )
{
 int lsb, msb;

 lsb = count & 0xff;
 msb = count >> 8;

 switch( counter )
 {
  case 1: /* select counter 1, LSB then MSB, mode 2 */
   outp( timer_base + TIMER_CNTL, (CW_SC0 | CW_LSBMSB | CW_M2) );
   outp( timer_base + TIMER_CNTR1, (unsigned char) lsb );
   outp( timer_base + TIMER_CNTR1, (unsigned char) msb );
  break;

  case 2: /* select counter 2, LSB then MSB, mode 2 */
   outp( timer_base + TIMER_CNTL, (CW_SC1 | CW_LSBMSB | CW_M2) );
   outp( timer_base + TIMER_CNTR2, (unsigned char) lsb );
   outp( timer_base + TIMER_CNTR2, (unsigned char) msb );
  break;

  case 3: /* select counter 3, LSB then MSB, mode 2 */
   outp( timer_base + TIMER_CNTL, (CW_SC2 | CW_LSBMSB | CW_M2) );
   outp( timer_base + TIMER_CNTR3, (unsigned char) lsb );
   outp( timer_base + TIMER_CNTR3, (unsigned char) msb );
  break;
 }

} /* load_counter */
```

```
/*****************************************************************/
/*  read_counter()                                               */
/*                                                               */
/*  purpose: Reads the appropriate counter in the appropriate    */
/*         bank with the remainin count and status.              */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*  parameters: int counter = 1, 2, 3 for COUNTER 1, 2, or 3     */
/*            unsigned int * count = remaining count             */
/*            unsigned char * status = counter status            */
/*****************************************************************/
/*  return value: none                                           */
/*****************************************************************/
void read_counter( int counter,
                   unsigned int * count, unsigned char * status )
{
 int lsb, msb;

 switch( counter )
 {
   case 1: /* select counter 1, LSB then MSB */
    outp( timer_base + TIMER_CNTL, ( CW_RBC | CW_RB_CNT | CW_RB_STAT |
CW_RB_C0 ) );
    *status = inp( timer_base + TIMER_CNTR1 ) & 0xFF;
    lsb = inp( timer_base + TIMER_CNTR1 ) & 0xFF;
    msb = inp( timer_base + TIMER_CNTR1 ) & 0xFF;
    msb = msb << 8;
    *count = ( lsb | msb );
   break;
```

```
   case 2: /* select counter 2, LSB then MSB */
    outp( timer_base + TIMER_CNTL, ( CW_RBC | CW_RB_CNT | CW_RB_STAT |
CW_RB_C1 ) );
    *status = inp( timer_base + TIMER_CNTR2 ) & 0xFF;
    lsb = inp( timer_base + TIMER_CNTR2 ) & 0xFF;
    msb = inp( timer_base + TIMER_CNTR2 ) & 0xFF;
    msb = msb << 8;
    *count = ( lsb | msb );
   break;

   case 3: /* select counter 3, LSB then MSB */
    outp( timer_base + TIMER_CNTL, ( CW_RBC | CW_RB_CNT | CW_RB_STAT |
CW_RB_C2 ) );
    *status = inp( timer_base + TIMER_CNTR3 ) & 0xFF;
    lsb = inp( timer_base + TIMER_CNTR3 ) & 0xFF;
    msb = inp( timer_base + TIMER_CNTR3 ) & 0xFF;
    msb = msb << 8;
    *count = ( lsb | msb );
   break;
  }

} /* read_counter */



/*****************************************************************/
/*  irq_rcvd()                                                   */
/*                                                               */
/*  purpose: Interrupt service routine used to service any of the  */
/*        counters on the 7589A.                                 */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*  parameters: none                                             */
```

```
/*****************************************************************/
/*  return value: none                                           */
/*****************************************************************/
void interrupt irq_rcvd(void)
{

 disable();

 asm {
  .386P
  push  eax
  push  ebx
 }

 tmr_status = inp( gpi_base ) & 0xFF;

 /* increment counts and clear status */
 if( tmr_status & GPI_T1 ) {
  t1_count++;
  outp( gpo_base, (gpo_org & (~GPO_T1)) );  /* clear timer 1 status bit */
 }
 if( tmr_status & GPI_T2 ) {
  t2_count++;
  outp( gpo_base, (gpo_org & (~GPO_T2)) );  /* clear timer 2 status bit */
 }
 if( tmr_status & GPI_T3 ) {
  t3_count++;
  outp( gpo_base, (gpo_org & (~GPO_T3)) );  /* clear timer 3 status bit */
 }

 outp( gpo_base, (gpo_org | GPO_T1 | GPO_T2 | GPO_T3) ); /* enable status */
```

```
  /* Non specific end of interrupt to PIC */
  outp(0x20, 0x20);  /* Master end of irq command */

  asm {
   .386P
   pop  ebx
   pop  eax
  }

  enable();

}

/*
```

** FILE: T_TIMERS.C

```
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <dos.h>

#include "pci.h"
#include "t7589At.h"

/* TIMERS.C function prototypes */
void far interrupt irq_rcvd( void );
void init_timer_int( void );
void restore_orig_int( void );
void load_counter( int, unsigned int );
void read_counter( int, unsigned int *, unsigned char * );

/* global variables */
unsigned char bus, dev_func;

/* the following globals are used in other files as 'extern' variables */
unsigned char tmr_status;
unsigned long t1_count;     /* counts no. of times timer 1 ISR entered */
unsigned long t2_count;     /* counts no. of times timer 1 ISR entered */
unsigned long t3_count;     /* counts no. of times timer 1 ISR entered */
unsigned int pwr_mgm_base;
unsigned int gpi_base;
unsigned int gpo_base;
unsigned int timer_base;
```

```
unsigned char pic1_org;
unsigned char gpo_org;

void main( int argc, char * argv[] )
{
  unsigned long t1;
  int test_int;
  unsigned long temp_dword;

  timer_base = 0x500;

  /* locate the power management device on the PCI bus */
  test_int = find_pci_device(DID_PWR_MGM, VID_PWR_MGM, 0,
             &bus, &dev_func);
  if(test_int != SUCCESSFUL)
  {
    printf("\nUnable to locate power management device on PCI bus\n");
    exit( 1 );
  }

  /* get base address from config area */
  test_int = read_configuration_area(READ_CONFIG_DWORD,
                          bus, dev_func, 0x40, &temp_dword);
  if(test_int != SUCCESSFUL)
  {
    printf("\nUnable to read POWER MGM. BASE ADDRESS @ 0x40 in config
space\n");
    exit( 1 );
  }
  pwr_mgm_base = temp_dword & 0x0000FFC0;
  gpi_base = pwr_mgm_base + 0x31; /* PIX general purpose input bits 8-15 */
  gpo_base = pwr_mgm_base + 0x37; /* PIX general purpose output bits 24-31 */
```

```
disable();

/* Read 8259 Programmable Interrupt controller */
pic1_org = inp(0x21) & 0xFF;

/* disable interrupt 5 */
outp(0x21, (pic1_org | 0x20) ); /* 0 = enable 1 = disable */

enable();

gpo_org = inp( gpo_base ) & 0xFF; /* save original general purpose out */


/* clear all three status bits in GPI */
outp( gpo_base, ( gpo_org & GPO_CLR ) );

/* set all three GPO outputs to 1 to allow int status registers to function */
outp( gpo_base, ( gpo_org | GPO_T1 | GPO_T2 | GPO_T3 ) );


/* setup timers interrupt service routine */
init_timer_int();

/*
** setup counter to generate an interrupt (counters 1)
*/

/* setup for interrupts to occur */
t1_count = 0;
t1 = 0;
tmr_status = 0;
```

```
test_int = 100;

/* load counter */
load_counter( 1, 0xFFFF );

do
{
  if( t1_count ) {
    t1++;
    break;
  }
  test_int--;
  delay( 1 );

} while( test_int );

/* disable timers by reloading the control word */
outp( timer_base + TIMER_CNTL, (CW_SC0 | CW_LSBMSB | CW_M2) );

restore_orig_int();
outp( gpo_base, gpo_org );
exit( 1 );

} /* end main */

/*
```

** FILE: PCI.H

```
**
*/

#define TRUE    1
#define FALSE0

#define CARRY_FLAG 0x01


/* PCI Functions */
#define PCI_FUNCTION_ID       0xB1
#define PCI_BIOS_PRESENT0x01
#define FIND_PCI_DEVICE0x02
#define FIND_PCI_CLASS_CODE0x03
#define READ_CONFIG_BYTE0x08
#define READ_CONFIG_WORD0x09
#define READ_CONFIG_DWORD     0x0A
#define WRITE_CONFIG_BYTE     0x0B
#define WRITE_CONFIG_WORD     0x0C
#define WRITE_CONFIG_DWORD    0x0D

/* PCI Return codes */

#define SUCCESSFUL0x00
#define NOT_SUCCESSFUL0x01


/* Prototypes */

int find_pci_device(unsigned short device_id,
                    unsigned short vendor_id,
```

```
                unsigned short index,
                unsigned char *bus_number,
                unsigned char *device_and_function);

int read_configuration_area(unsigned char function,
                unsigned char bus_number,
                unsigned char device_and_function,
                unsigned char register_number,
                unsigned long *data);

int write_configuration_area(unsigned char function,
                unsigned char bus_number,
                unsigned char device_and_function,
                unsigned char register_number,
                unsigned long value);
```

** FILE: PIC.H

```
/*                                                      */
/*     Bit definitions for the PC/AT programable interrupt controller (PIC) */
/*                                                      */
/*                                                      */

#define       PIC1          0x20    /* I/O port addr of PIC 1 */
#define       PIC1_MASK     0x21    /* I/O port addr of PIC 1 mask reg */
#define       PIC2          0xA0    /* I/O port addr of PIC 2 (AT only) */
#define       PIC2_MASK     0xA1    /* I/O port addr of PIC 2 mask reg */

#define       IRQ0_MASK     0x01    /* mask off int 0 - timer 0 sys tic */
#define       IRQ1_MASK     0x02    /* mask off int 1 - keyboard */
#define       IRQ2_MASK     0x04    /* mask off int 2 - rsvd XT; 8-15 AT */
#define       IRQ3_MASK     0x08    /* mask off int 3 - com port */
#define       IRQ4_MASK     0x10    /* mask off int 4 - com port */
#define       IRQ5_MASK     0x20    /* mask off int 5 - HD XT; LPT AT */
#define       IRQ6_MASK     0x40    /* mask off int 6 - Floppy Disk */
#define       IRQ7_MASK     0x80    /* mask off int 7 - LPT */

#define       IRQ8_MASK     0x01    /* mask off int 8 - RTC */
#define       IRQ9_MASK     0x02    /* mask off int 9 - Re-directed IRQ2 */
#define       IRQ10_MASK    0x04    /* mask off int 10 - Unassigned */
#define       IRQ11_MASK    0x08    /* mask off int 11 - Unassigned */
#define       IRQ12_MASK    0x10    /* mask off int 12 - Unassigned */
#define       IRQ13_MASK    0x20    /* mask off int 13 - Co-processor */
#define       IRQ14_MASK    0x40    /* mask off int 14 - HD AT */
#define       IRQ15_MASK    0x80    /* mask off int 15 - Unassigned */

#define       PIC_EOI       0x20    /* PIC End Of Interrupt */

/*
```

** FILE: T7589T.H

```
**
*/

#define    DID_PWR_MGM    0x7113
#define    VID_PWR_MGM    0x8086

#define    IRQ5        0x0D

#define    GPI_T1       0x80   /* PIX General Purpose Input 15 (tmr 1)  */
#define    GPI_T2       0x40   /* PIX General Purpose Input 14 (tmr 2)  */
#define    GPI_T3       0x20   /* PIX General Purpose Input 13 (tmr 3)  */
#define    GPO_T1       0x40    /* PIX General Purpose Output 30 (tmr 1) */
#define    GPO_T2       0x10    /* PIX General Purpose Output 28 (tmr 2) */
#define    GPO_T3       0x08    /* PIX General Purpose Output 27 (tmr 3) */

#define    GPO_CLR      0xA7    /* PIX General Purpose Output CLR TMRS   */

#define    TIMER_CNTR1    0x00    /* Timer counter 1 offset          */
#define    TIMER_CNTR2    0x01    /* Timer counter 2 offset          */
#define    TIMER_CNTR3    0x02    /* Timer counter 3 offset          */
#define    TIMER_CNTL     0x03    /* Timer control offset            */

/*************************************************************************/
/* 8254 Control word                                                     */
/*************************************************************************/
#define    CW_SC0       0x00    /* W Selcect counter 0             */
#define    CW_SC1       0x40    /* W Selcect counter 1             */
#define    CW_SC2       0x80    /* W Selcect counter 2             */
#define    CW_RBC       0xC0    /* W Read back command             */
#define    CW_CLC       0x00    /* W Cntr latch command (cnt/stat)     */
#define    CW_SLC       0x00    /* W Status latch command          */
```

```
#define   CW_LSB       0x10   /* W LSB only                */
#define   CW_MSB       0x20   /* W MSB only                */
#define   CW_LSBMSB    0x30   /* W LSB first then MSB      */
#define   CW_M0        0x00   /* W Mode 0                  */
#define   CW_M1        0x02   /* W Mode 1                  */
#define   CW_M2        0x04   /* W Mode 2                  */
#define   CW_M3        0x06   /* W Mode 3                  */
#define   CW_M4        0x08   /* W Mode 4                  */
#define   CW_M5        0x0A   /* W Mode 5                  */
#define   CW_BCD       0x01   /* W Binary Coded Decimal    */
#define   CW_RB_CNT    0x00   /* W Read back count         */
#define   CW_RB_STAT   0x00   /* W Read back status        */
#define   CW_RB_C0     0x02   /* W Read back counter 0     */
#define   CW_RB_C1     0x04   /* W Read back counter 1     */
#define   CW_RB_C2     0x08   /* W Read back counter 2     */
```

Directory WATCHDOG

This directory contains sample code useful in the creation of applications involving the VMIVME-7589A's watchdog timer function as described in Chapter 4.

** FILE:WDTO.C

```
**
**     Watchdog timeout
*/

#include   <stdlib.h>
#include   <stdio.h>
#include   <dos.h>
#include   <time.h>
#include   <conio.h>
#include   <ctype.h>
#include   "watchdog.h"

unsigned char far * wd_ptr;
char usr[80];
unsigned char dog;

void main( void ) {

 wd_ptr = (unsigned char far *) MK_FP( 0xD800, 0 );

 /* set WatchDog Alarm Mask 1 - deactivated and update with 0 time */
*(wd_ptr + WD_CMD) = ( WD_TE | WD_WAM );
*(wd_ptr + WD_MSEC) = 0; /* load with 0 to disable */
*(wd_ptr + WD_SEC) = 0;  /* load with 0 to disable */
*(wd_ptr + WD_CMD) = ( WD_TE | WD_WAM ); /* allow update with 0 time */
*(wd_ptr + WD_CMD) = WD_WAM; /* set watchdog alarm mask to 1 */
```

```
  printf("\nJumper J30 installed? [Y/N]: ");
  scanf("%s%*c", usr );
  fflush( stdin );
  if( toupper( usr[0] ) != 'Y' ) exit( 1 );

  printf("\n");

  *(wd_ptr + WD_MSEC) = 0x99; /* 00.99 seconds */
  *(wd_ptr + WD_SEC) = 0x99;  /* 99.00 seconds */
  *(wd_ptr + WD_CMD) = ( WD_TE | WD_PU ); /* set for 3 ms pulse */

  printf("Reset time out in 100 seconds (1 min 40 sec)\n\n");

  /* read one of the alarm regs to cause reload and prevent time out */
  /* dog = *(wd_ptr + WD_MSEC); */

} /* end main */


/*
```

** FILE: WATCHDOG.H

```
**
** DS1384 REGISTER OFFSETS
*/
                                /*7 6 5 4 3 2 1 0*/
#define     CLK_MSEC      0x00    /* 00-99                    */
#define     CLK_SEC       0x01    /* 00-59 0                  */
#define     CLK_MIN       0x02    /* 00-59 0                  */
#define     CLK_MINAL     0x03    /* 00-59 M                  */
#define     CLK_HRS       0x04    /* 01-12+A/P OR 00-23       */
#define     CLK_HRSAL     0x05    /* 01-12+A/P OR 00-23       */
#define     CLK_DAY       0x06    /* 01-07 0 0 0 0 0          */
#define     CLK_DAYAL     0x07    /* 01-07 M 0 0 0 0          */
#define     CLK_DATE      0x08    /* 01-31 0 0                */
#define     CLK_MONTH     0x09    /* 01-12     0              */
#define     CLK_YRS       0x0A    /* 00-99                    */
#define     WD_CMD        0x0B    /* command register         */
#define     WD_MSEC       0x0C    /* milli second watchdog time   */
#define     WD_SEC        0x0D    /* seconds watchdog time        */

/*
** DS1384 COMMAND REGSITER BIT DEFINITIONS
*/

#define     WD_TE         0x80    /* transfer enable 1 - allow updates */
#define     WD_IPSW       0x40    /* interrupt switch 0 - WD out INTA  */
#define     WD_IBHL       0x20    /* int. B output 0 - current sink    */
#define     WD_PU         0x10    /* pulse/level 1 - 3 ms pulse        */
#define     WD_WAM        0x08    /* watchdog alarm mask 0 - active    */
#define     WD_TDM        0x04    /* time-of-day alarm mask 0 - active */
#define     WD_WAF        0x02    /* watchdog alram flag          */
#define     WD_TDF        0x01    /* time-of-day flag             */
```

Index

I

J

K

L

M

N

O

P

R

S

T

U

V

W