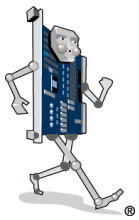
VMIVME-7486
486 PC/AT VMEbus CPU

Product Manual

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

500-007486-000 Rev. E

COPYRIGHT AND TRADEMARKS



(I/O man figure)

(Instant OPC wizard logo)

(IOWorks man figure)

VMIC

RECORD OF REVISIONS

REVISION LETTER	DATE	PAGES INVOLVED	CHANGE NUMBER
A	02 / 11 / 94	Release	94-0154
B	11 / 22 / 94	Revised	94-0516
C	05 / 01 / 96	Cover and page ii	96-0310
D	09 / 10 / 96	Cover and pages iii through xiii	96-0639
E	12 / 09 / 99	Reformat and change area code to 256	99-0962

Table of Contents

List of Figures

List of Tables

Safety

Contents

VMIC Safety Summary

Note The following general safety precautions must be observed during all phases of the operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of this product. VME Microsystems International Corporation assumes no liability for the customer's failure to comply with these requirements.

Ground the System

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

Do Not Operate in an Explosive Atmosphere

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

Keep Away from Live Circuits

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

Do Not Service or Adjust Alone

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

Do Not Substitute Parts or Modify System

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

Dangerous Procedure Warnings

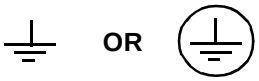
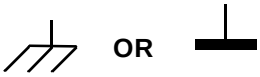
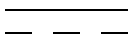
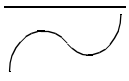
Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

W A R N I N G

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

Safety Symbols

General definitions of safety symbols used in this manual:

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

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

CHAPTER

1

Introduction

Contents

Introduction to the VMIVME-7486

The VMIVME-7486 single-board microcomputer is a complete IBM PC / AT 80486 compatible microcomputer with the additional benefits of Eurocard construction and full compatibility with the VMEbus Specification Rev. C.1.

VMIC's VMIVME-7486 has two operating modes: a standard PC / AT compatible mode, and a VMEbus controller mode. Upon power-up it functions as a standard PC / AT. It executes a PC / AT-type power-on self-test, then boots up MS-DOS, Windows, OS / 2, NextStep, XENIX, or any other PC / AT compatible operating system. Its keyboard and video console interaction with the user is typical of a PC / AT. This PC / AT mode of the VMIVME-7486 is discussed in Chapter Three, *PC/AT Functions* on page 55of this manual.

After booting, the VMIVME-7486 may take on the additional functions of a VMEbus controller and interact with other VMEbus modules. The VMEbus controller functions are available by programming the VMIVME-7486's VMEbus interface registers according to Chapter Four, *VMEbus Functions* on page 71 of this manual.

With VMIC's VMIVME-7420 VMEaccess software, the VMIVME-7486 programmer may quickly and easily control VMEbus slave boards simply by linking to a library of VMIVME-7486 VMEbus interrupt and control functions.

About This Manual

Because this product bridges the traditionally divergent worlds of Intel-based PCs and Motorola-based VMEbus controllers, some confusion over *conventional* notation and terminology may exist. We have made every effort to make this manual consistent by adhering to conventions typical for the Motorola/VMEbus world; nevertheless, users in both camps should review the following notes:

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

  Segment:Offset to Linear Address

  Linear Address = (Segment × 16) + Offset

  Linear Address to Segment:Offset

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

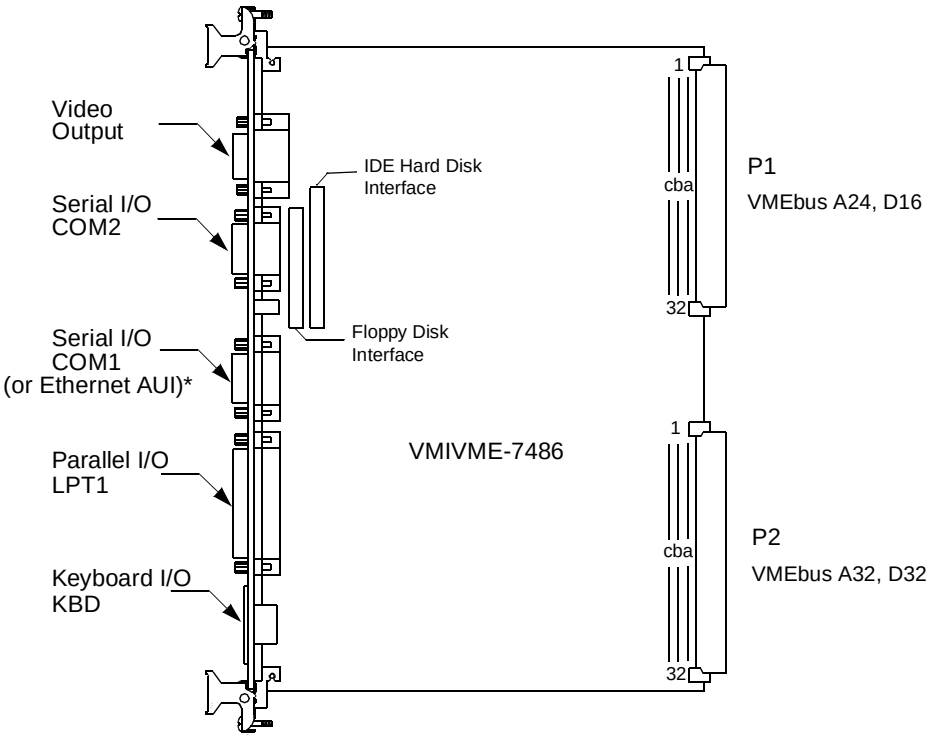
  Offset = <remainder> × 65536

  Where <remainder> = the remainder of (Linear Address ÷ 65536)

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

Product Family

Figure 1-1 shows a simplified view of the VMIVME-7486 board. The VMIVME-7486 is one member of VMIC's line of PC / AT compatible VMEbus controllers, all of which combine a standard PC / AT architecture with the ability to control VMEbus slave boards.



* Serial port COM1 is unavailable if the Ethernet Mezzanine option is ordered, since the Ethernet AUI connector replaces the COM1 connector. See Appendix A, *Ethernet Option* on page 99 for Ethernet Mezzanine option details.

Figure 1-1 VMIVME-7486 Board View

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

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

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

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

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

References

For the most up-to-date physical description and specifications for the VMIVME-7486, please refer to VMIC specification number 800-017486-000.

There are many books widely available on the subject of general PC / AT use and programming. Some reference sources which may be particularly helpful in using or programming the VMIVME-7486 are listed below.

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

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

PC/AT Features

The VMIVME-7486 performs all the functions of a standard IBM PC / AT motherboard with the following features:

- Single-Slot 6U Size
- High performance 80486 processor
  Available in speeds from 33 MHz to 66 MHz
  Standard 8 Kbyte internal cache
  High performace math coprocessor on 80486DX versions
- Up to 16 Mbytes of RAM
- Super-VGA video
  1 Megabyte of video DRAM
  Resolutions up to 1024x768, non-interlaced, 256 colors
- Battery-backed clock / calendar
- Front panel reset switch, and indicators for hard drive activity, power, and status
- On-board ports for keyboard, IDE hard drive, floppy drive, serial and parallel I / O

The VMIVME-7486 also supports standard PC / AT I / O features such as those listed in Table 1-1 on page 26. Figure 1-2 on page 25 also shows a partial block diagram of the VMIVME-7486, emphasizing the I / O features.

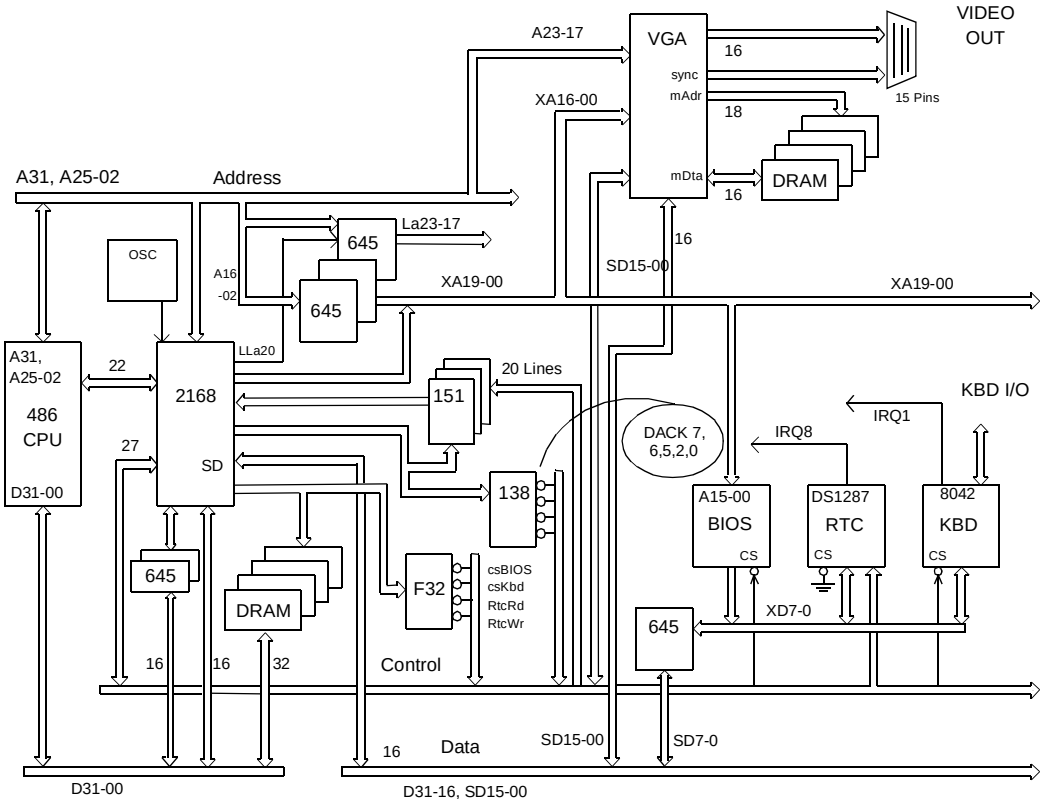


Figure 1-2 VMIVME-7486 Partial Block Diagram

Table 1-1 PC/AT I/O Features

I/O Feature	MS-DOS Identifier	Physical Access
Two Serial Ports (16450-compatible RS-232C)	COM1, COM2	Front Panel DB9P (male) X 2
One Parallel Port (standard Centronics TTL)	LPT1	Front Panel DB25S (female)
AT-style Keyboard Controller with PS/2-style Adapter	KBD	Front Panel PS/2-Style Mini-DIN Circular (female)
Real-time Clock/Calendar with Battery and Watchdog Timer	Date, Time	
SuperVGA Video Controller with 1 Megabyte DRAM	Display	Front Panel DB15HD High Density (female)
Floppy Disk Controller (2 drives maximum)	Drives A, B	34-pin Header Flat Cable type
IDE Fixed Disk Controller (2 drives maximum)	Drives C, D	40-pin Header Flat Cable type
Hardware Reset	Cold Boot	Front Panel Pushbutton
IBM/PC Sound	Beep	On-board Small Speaker

VMEbus Features

The VMIVME-7486 incorporates the following VMEbus functions:

- Single-slot, 6U-height VMEbus board
- Complete 6-line Address Modifier (AM-Code) programmability
- 32-bit data interface with 3-way byte/word swap in hardware
- User-configured interrupt handler
- User-configured interrupter
- System Controller mode (jumper enabled) as level 3 single-level arbiter
- Complete VMEbus access through
  Real-mode memory window, or
  Protected-mode linear addressing above local DRAM
- Front panel "vital sign" indicators
  (+5V Power, programmable SYSFAIL* signal)

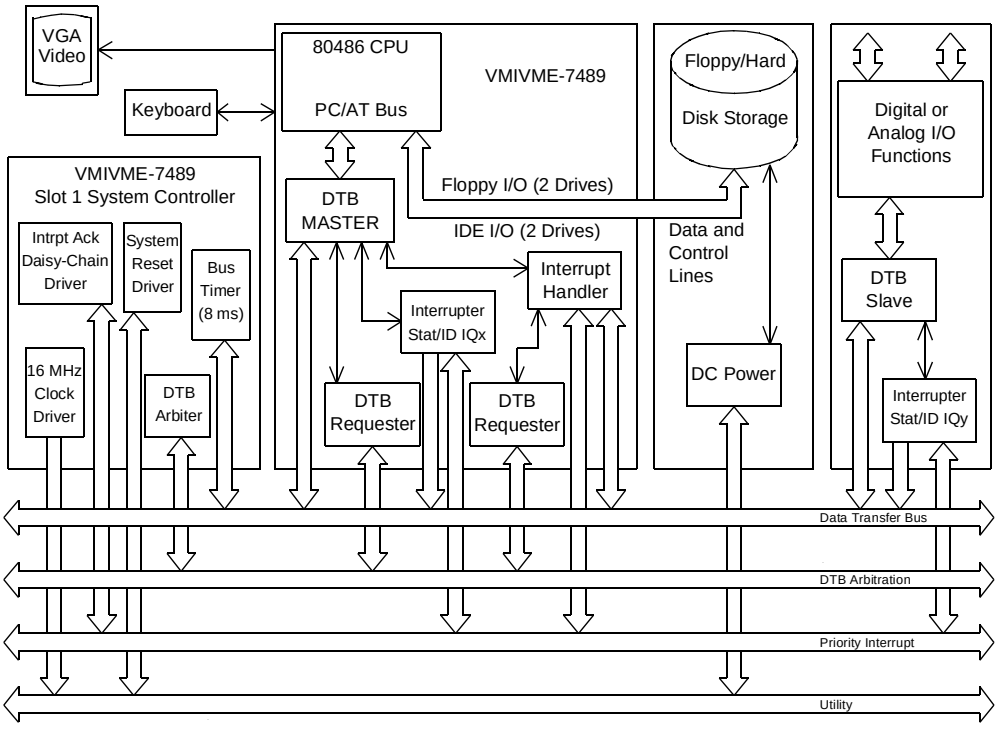


Figure 1-3 VMIVME-7486 System VMEbus Functions

Figure 1-3 shows the VMIVME-7486 functions in a typical VMEbus system. The VMIVME-7486 is a versatile single-board solution for VMEbus control with familiar PC / AT operation.

CHAPTER
2

Installation and Setup

Contents

Introduction

This chapter describes unpacking, inspection, hardware jumper settings, connector definitions, installation, system setup, and operation of the VMIVME-7486.

Unpacking Procedures

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

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

Hardware Setup

The VMIVME-7486 has been tested for system operation and shipped with factory-installed header jumpers. Figure 2-1 illustrates the physical location of the user-configurable jumpers and connectors on the board. Table 2-1 on page 32 lists each jumper designator, its function, and the factory-installed default configuration.

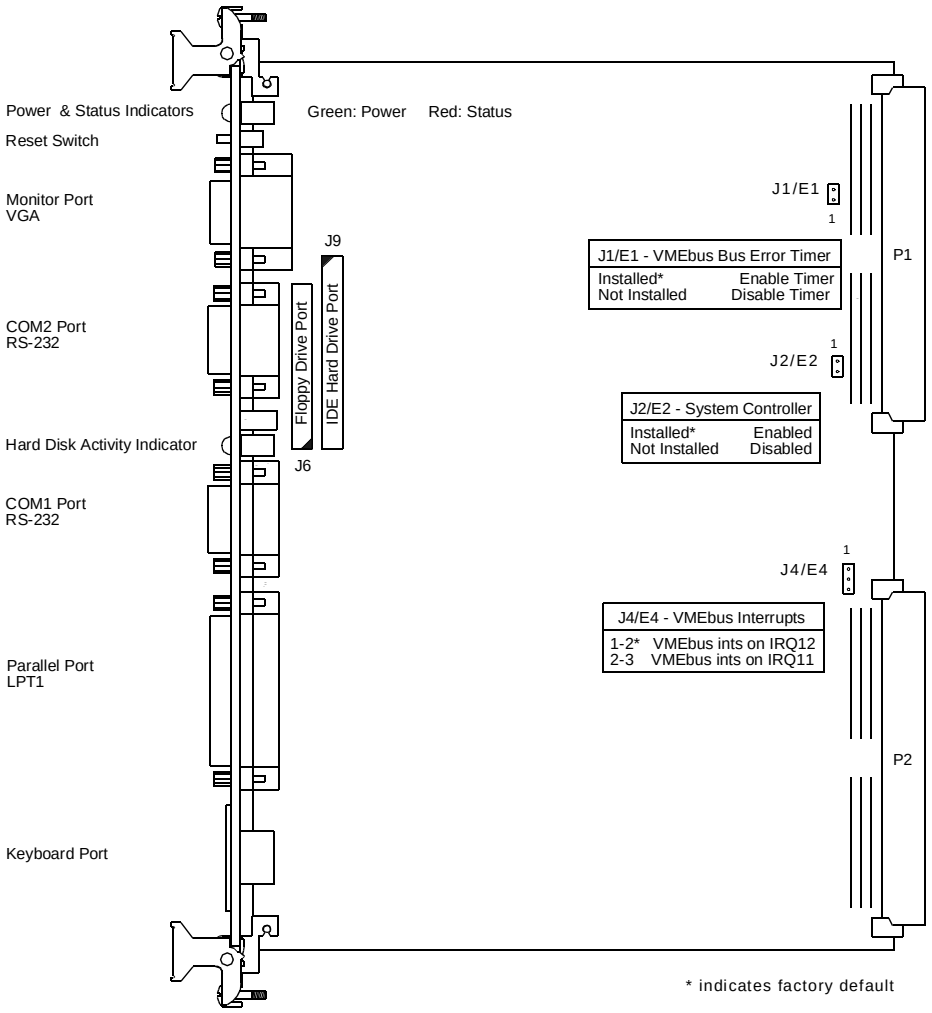


Figure 2-1 I/O Port and Jumper Locations

Table 2-1 VMIVME-7486 Jumper Header Functions & Factory Settings

Header Number	FUNCTION	FACTORY SETTING
J1 / E1	Enable VMEbus Bus Error 10 µS Timeout Timer	Installed
J2 / E2	Enable System Controller	Installed
J4 / E4	Selects VMEbus interrupts through either IRQ11 or IRQ12 of the PC/AT interrupt controller.	1-2: Select IRQ12 (Default) 2-3: Select IRQ11

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

Physical Installation

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

De-energize the equipment and insert the board into an appropriate slot of the chassis. While ensuring that the board is properly aligned and oriented in the supporting board guides, slide the board smoothly forward against the mating connector until firmly seated. Note that if a drive module or other similar expansion module is to be used, it must typically be mounted and connected to the VMIVME-7486 first. Follow the instructions for the expansion module and ensure all connectors are properly mated before inserting the whole assembly into the VMEbus.

VMEbus Connector Pinout

The VMIVME-7486 conforms to the VMEbus physical specification for a 6U x 4HP dual Eurocard (dual height, single-slot width). It can be plugged directly into any standard chassis accepting this type of board.

The VMIVME-7486 may be attached to a dual P1/P2 VMEbus backplane or a single P1 backplane. For 16-bit data transfers and 24-bit addressing a single P1 backplane is sufficient, but for 32-bit data transfers and 32-bit addressing a dual P1/P2 backplane is required.

Front panel connections for the parallel, video, and serial ports are typical for any PC/AT. The keyboard connector is a standard mini-DIN PS/2 style connector; an adapter is supplied to connect a keyboard with a larger connector to the VMIVME-7486.

Table 2-2 shows the pin assignments for the VMEbus P1 and P2 connectors. Note that only Row B of connector P2 is used: all other pins on P2 are reserved and should not be connected. A detailed discussion of VMEbus hardware signals (such as the Bus Grant and Interrupt Acknowledge Daisy Chains) is included in Chapter Four.

Table 2-2 VMEbus P1 & P2 Connector Pin Assignments

Pin Number	P1 Row A Signal	P1 Row B Signal	P1 Row C Signal	P2 Row B Signal
1	D00	BBSY*	D08	+5V
2	D01	BCLR* (Not Used)	D09	GND
3	D02	ACFAIL*	D10	Reserved
4	D03	BG0IN* (Ignored)	D11	A24
5	D04	BG0OUT* (Ignored)	D12	A25
6	D05	BG1IN* (Ignored)	D13	A26
7	D06	BG1OUT* (Ignored)	D14	A27
8	D07	BG2IN* (Ignored)	D15	A28
9	GND	BG2OUT* (Ignored)	GND	A29
10	SYSCLK	BG3IN*	SYSFAIL*	A30
11	GND	BG3OUT*	BERR*	A31
12	DS1*	BR0* (Ignored)	SYSRESET*	GND
13	DS0*	BR1* (Ignored)	LWORD*	+5V
14	WRITE*	BR2* (Ignored)	AM5	D16
15	GND	BR3*	A23	D17

Table 2-2 VMEbus P1 & P2 Connector Pin Assignments (Continued)

Pin Number	P1 Row A Signal	P1 Row B Signal	P1 Row C Signal	P2 Row B Signal
16	DTACK*	AM0	A22	D18
17	GND	AM1	A21	D19
18	AS*	AM2	A20	D20
19	GND	AM3	A19	D21
20	IACK*	GND	A18	D22
21	IACKIN*	SERCLK (Ignored)	A17	D23
22	IACKOUT*	SERDAT* (Ignored)	A16	GND
23	AM4	GND	A15	D24
24	A07	IRQ7*	A14	D25
25	A06	IRQ6*	A13	D26
26	A05	IRQ5*	A12	D27
27	A04	IRQ4*	A11	D28
28	A03	IRQ3*	A10	D29
29	A02	IRQ2*	A09	D30
30	A01	IRQ1*	A08	D31
31	-12V	+5V STDBY	+12V	GND
32	+5V	+5V	+5V	+5V

IDE Connector Pinout

Table 2-3 describes the pin assignment for the IDE / ATA hard drive header (J9 on Figure 2-1). Like the floppy header it has an odd-numbered row and an even-numbered row (20-pins per row, 40-pins total).

Table 2-3 IDE/ATA Hard Drive Header Pin-out

PIN	SIGNAL	Direction	DESCRIPTION
1	Host RESET	Out	Reset Drive
2	GND	Out	Signal Ground
3	Host DATA7	In / Out	Bidirectional Data 07
4	Host DATA8	In / Out	Bidirectional Data 08
5	Host DATA6	In / Out	Bidirectional Data 06
6	Host DATA9	In / Out	Bidirectional Data 09
7	Host DATA5	In / Out	Bidirectional Data 05
8	Host DATA10	In / Out	Bidirectional Data 10
9	Host DATA4	In / Out	Bidirectional Data 04
10	Host DATA11	In / Out	Bidirectional Data 11
11	Host DATA3	In / Out	Bidirectional Data 03
12	Host DATA12	In / Out	Bidirectional Data 12
13	Host DATA2	In / Out	Bidirectional Data 02
14	Host DATA13	In / Out	Bidirectional Data 13
15	Host DATA1	In / Out	Bidirectional Data 01
16	Host DATA14	In / Out	Bidirectional Data 14
17	Host DATA0	In / Out	Bidirectional Data 00
18	Host DATA15	In / Out	Bidirectional Data 15
19	GND	Out	Signal Ground
20	Key	None	Unused, Keying Position
21	Reserved		
22	GND	Out	Signal Ground
23	Host IOW-	Out	Write Strobe
24	GND	Out	Signal Ground

Table 2-3 IDE/ATA Hard Drive Header Pin-out (Continued)

PIN	SIGNAL	Direction	DESCRIPTION
25	Host IOR-	Out	Read Strobe
26	GND	Out	Signal Ground
27	Reserved		
28	Host ALE	Out	Address Latch Enable
29	Reserved		
30	GND	Out	Signal Ground
31	IRQ14	In	Interrupt Request #14
32	Host IO16-	In	16-bit Data Word Size
33	A1	Out	Address Line #1
34	Host Diag-	In	Diagnostic Test Passed
35	A0	Out	Address Line #0
36	A2	Out	Address Line #2
37	Host CS0-	Out	Chip Select #0
38	Host CS1-	Out	Chip Select #1
39	Host SLV/ACT-	In	Slave/Activity Status
40	GND	Out	Signal Ground

Up to two IDE hard drives are directly supported in a multitude of storage capacities. User selection of the type of hard drive(s) and specific storage format is done during general BIOS Setup procedures.

For convenient disk storage for the VMIVME-7486, VMIC has drive modules that support a floppy drive and hard drive in a VMEbus 6U form factor. Power is drawn from the VMEbus P1 connector and ribbon cables from the VMIVME-7486 connect directly to the floppy and hard disk drives.

BIOS Setup

The VMIVME-7486 has an on-board 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.

Quick BIOS Setup

The BIOS Setup program has many powerful options, but only a few are critical for normal VMIVME-7486 operation, and these are stored as default values in the BIOS Setup program. What follows is a summary of the procedure necessary to load the default BIOS Setup values and configure the system for normal VMIVME-7486 operations. For more detailed descriptions of all the BIOS Setup options, read the subsequent sections on BIOS Features and BIOS Utilities.

1. Reboot the system. This may be accomplished either by applying power to the system, pressing the front panel reset switch, or pressing **<**CTRL+ALT+DEL**>** from the keyboard if the system is already running.
2. Press **<**DEL**>** to enter the setup program as soon as the screen displays the **Press <Del> to run Setup** message. You may have to press **<**F1**>** to clear the keyboard error first
3. When the opening screen appears (see Figure 2-2 on page 41), press **<**F2**>** or **<**F3**>** repeatedly to select a readable screen color.
4. Select the **Auto Configuration with BIOS Defaults** option and confirm at the prompt by pressing **<**Y**>** then **<**ENTER**>**.
5. After the default values are loaded, press any key to continue.
6. Select the **Auto Detect Hard Disk** utility and confirm the auto-detect settings for the hard disks. Note that if no hard disk is present, it may take up to a minute to determine this fact.
7. Select the **Standard CMOS Setup** utility and confirm at the warning screen (see Figure 2-3 on page 43).
8. From the Standard CMOS Setup screen, set the date and time.
9. From the Standard CMOS Setup screen, configure the floppy drive type(s) (and the hard drive type if it was not properly auto-detected). The type of drive must be properly entered or boot errors will occur.
10. Press **<**ESC**>** to return to the opening screen.
11. Select the **Write to CMOS and Exit** option and confirm.
12. The VMIVME-7486 will automatically reboot with the new values in effect.

BIOS OVERVIEW

The system BIOS (Basic Input/Output System) is a collection of device drivers, initialization routines, system data, and other code that controls the interface between the operating system (e.g., MS-DOS, UNIX, XENIX, etc.) and the system hardware.

Several types of BIOS may exist in a PC system. Every PC has a system BIOS like that described above. Usually a video BIOS also exists to control the interface between the video adapter and the CPU. In addition, option ROMs (which can also be BIOS ROMs) may be present to control specific hardware devices, such as hard drives or network interfaces.

PC/AT compatible systems, also called ISA (Industry Standard Architecture) systems, must have a place to store system information when the computer is turned off. The original IBM AT had 64 bytes of nonvolatile memory storage in CMOS RAM. All PC/AT compatible systems have at least 64 bytes of CMOS RAM, which is usually part of the real-time clock device. The VMIVME-7486, for example, has 114 bytes of CMOS RAM. The BIOS Setup program uses this nonvolatile storage area to keep configuration data – this is why the term *CMOS Setup* is often used synonymously with *BIOS Setup*.

The system BIOS of the VMIVME-7486 is stored in a ROM (Read Only Memory) chip. The BIOS is located at physical address $F0000 and occupies 64 Kbytes of main memory, thus occupying the topmost 64 Kbytes of the 1 Mbyte Real Mode addressing space. An image of this BIOS also appears in the topmost 64 Kbytes of Protected Mode address space, beginning at address $FFFF 0000.

BIOS Features

Auto Detect

The BIOS automatically detects all system DRAM, the type of microprocessor used in the system, and on-board floppy, IDE, serial I/O, and parallel I/O controllers. It automatically configures on-board controllers to prevent conflicts with controllers in I/O expansion slots.

User-Definable Hard Disk Types

The BIOS Setup allows the user to define hard disk types for both hard drives in a system.

Supports Nonstandard Systems

The BIOS can be configured to bypass keyboard, floppy, and video boot errors so specialized systems (such as file servers) without keyboards, floppies, or monitors can be configured easily.

Shadow RAM Support

The BIOS can selectively shadow 32 Kbytes of ROM at $C0000 and $C8000. The 64 Kbyte system BIOS ROM at $F0000 is automatically shadowed to RAM.

ROM shadowing is a technique whereby the contents of ROM are copied from slow ROM to RAM, where the data can be accessed more than four times faster. Often the ROM is only eight bits or sixteen bits wide while the RAM is as wide as the processor's memory bus. In any case the wider and speedier RAM shadow is accessed much faster than the narrower and slower ROM.

In the BIOS Setup, shadow RAM settings appear on the Advanced CMOS Setup screen. Each option permits an address segment to be shadowed from ROM to RAM.

Typematic Rate and Delay

The BIOS Setup can configure the repetition speed of a keystroke and the time delay before the repeating starts.

Num Lock

The keyboard's **Num Lock** function may be set to on or off after system boot by the BIOS Setup.

Boot Sequence and POST Control

The BIOS Setup allows the VMIVME-7486 to boot from drive C: or drive A: first. Various elements of the power-on self-test may also be disabled in order to speed up the reboot process.

Fast Gate A20 Support

The BIOS supports the Fast Gate A20 option of the VMIVME-7486. Normally, the keyboard controller is used to switch between Real and Protected addressing modes with Gate A20. Fast Gate A20 substitutes a faster method for address mode switching using I/O ports.

BIOS Utilities

The BIOS in the VMIVME-7486 includes an advanced setup program that not only allows CMOS configuration, but contains other useful utilities as well. Here is a summary of the BIOS Setup functions:

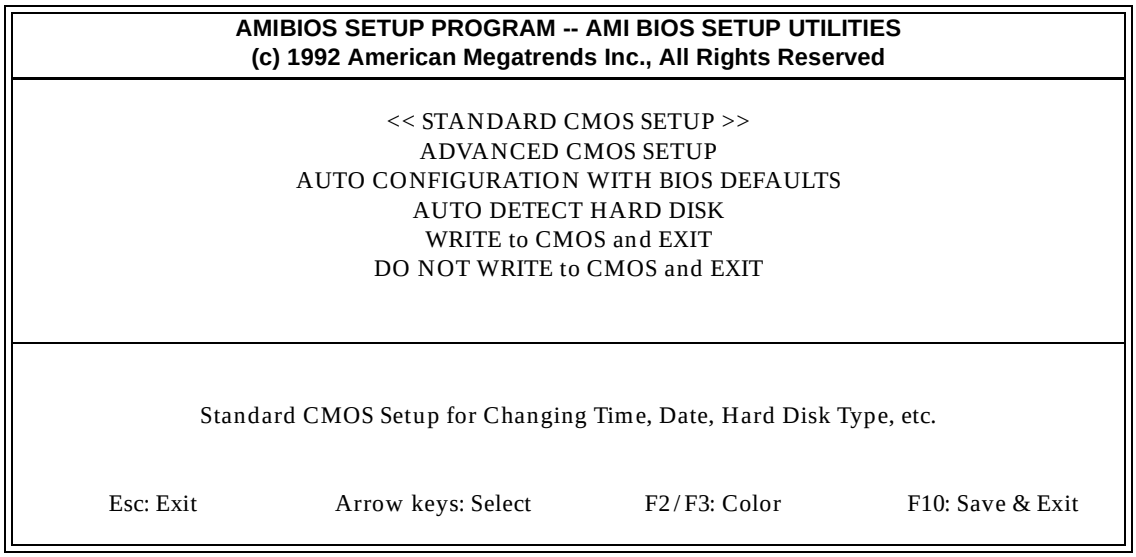
- Automatic CMOS setup and configuration
- Manual CMOS setup and configuration
- Automatic hard disk setup and configuration

Accessing the BIOS Utilities

The BIOS Setup program may only be activated immediately after power up, reset, or reboot. To access the opening screen, perform the following steps:

1. Reboot the system. This may be accomplished either by applying power to the system, pressing the front panel reset switch, or pressing **<**CTRL+ALT+DEL**>** from the keyboard if the system is already running.
2. Press the **<**DEL**>** key during subsequent power-on self-testing (POST processing). The keystroke will generate a keyboard error, and the BIOS will stop execution and inquire whether the user wants to enter setup. You may have to press **<**F1**>** to clear the keyboard error first.
3. Press **<**DEL**>** to enter the setup program and display the opening screen as shown in Figure 2-2 on page 41.

Figure 2-2 AMI BIOS Setup Opening Screen



Using the BIOS Setup Program

While running the setup program, the following keystroke actions are defined. Note that some keystrokes only work with certain screens.

Table 2-4 BIOS Setup Keystroke Actions

Keystroke	Action
<ESC>	Returns to previous screen (or exit from Opening Screen).
Cursor Keys	Moves the cursor from one option to the next.
<PG UP> <PG DN> <CTRL-PGUP> <CTRL-PGDN>	Modifies the default value of the options for the highlighted parameter. If there are fewer than ten options, <CTRL-PGUP> and <CTRL-PGDN> operate just like plain <PG UP> and <PG DN>.
<F1>	Displays help messages.
<F2>	Changes background colors.
<F3>	Changes foreground colors.
<F5>	Restores the values resident when the current Setup session began. These values are taken from CMOS memory if CMOS memory was uncorrupted at the start of the session. Otherwise, they will be the default values.
<F6>	Loads all features in the Advanced CMOS Setup screen with the BIOS Setup defaults.
<F7>	Loads all features in the Advanced CMOS Setup screen with the Power-On defaults, which are the same as the BIOS defaults on the VMIVME-7486.
<F10>	Saves all changes made to BIOS Setup and returns to DOS.

Note The default value for <F5>, <F6>, and <F7> is always *N*. To execute these options, press <Y> then <ENTER>.

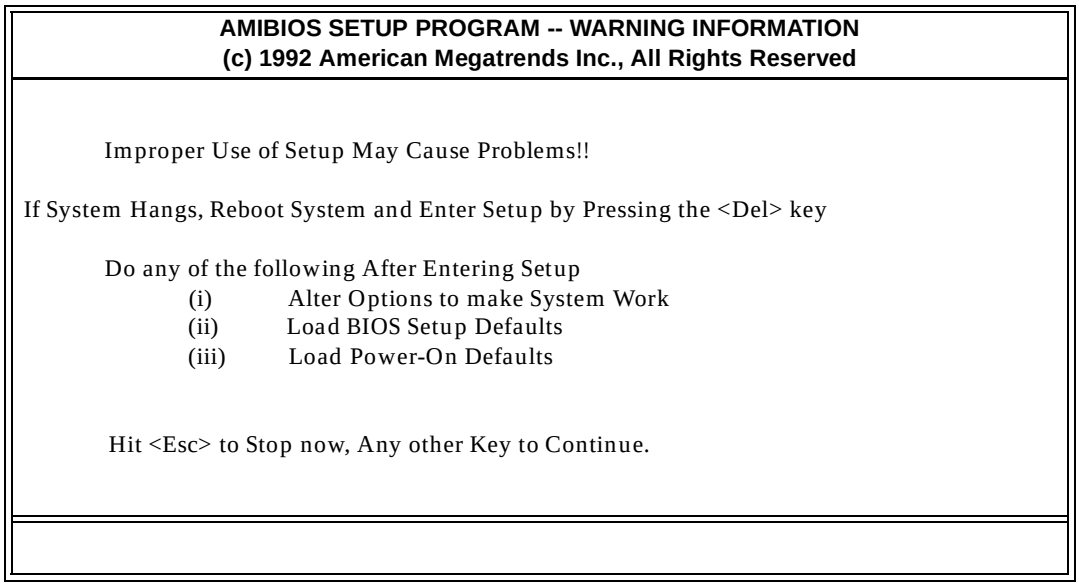
The Setup Utility is divided into the following options:

- **Standard CMOS Setup**
- **Advanced CMOS Setup**
- **Auto Configure with BIOS Defaults**
- **Auto Detect Hard Disk**
- **Exit and Save CMOS Settings**
- **Exit Without Saving**

Figure 2-2 on page 41 shows the **Standard CMOS Setup** option highlighted in the menu list area and the corresponding status message in the message area. The first two CMOS setup options allow manual configuration of the VMIVME-7486. The **Auto Configuration with BIOS Defaults** option restores the VMIVME-7486 with all its factory default CMOS settings (see the section Quick BIOS Setup on page 38 for an example using this option). The **Auto Detect Hard Disk** option is very convenient for configuring a hard drive of unknown type. The last two options exit from the Setup Screen.

Each time the user selects **Standard CMOS Setup** or **Advanced CMOS Setup**, the warning screen shown in Figure 2-3 is displayed. Press any key to remove the warning message and proceed to the selected setup screen.

Figure 2-3 BIOS Setup Warning Screen



Standard CMOS Setup

The Standard CMOS Setup screen permits the user to configure and set system components such as the time and date, floppy / hard disk drives, video adapter type, and keyboard type. **Standard CMOS Setup** is the topmost option on the setup opening screen. Press **<**ENTER**>** when this option is highlighted and a screen similar to the one shown in Figure 2-4 appears after the warning screen.

Figure 2-4 Standard CMOS Setup Screen

AMIBIOS SETUP PROGRAM -- STANDARD CMOS SETUP
(c) 1992 American Megatrends, Inc., All Rights Reserved

Date (mm / date / year)	: Thu, Jan 31 1991	Base memory	: 640 KB
Time (hour / min / sec)	: 15 : 23 : 15	Ext. Memory	: 0 KB

	Cyln	Head	WPcom	LZone	Sect	Size
Hard disk C: type	: Not Installed					
Hard disk D: type	: Not Installed					
Floppy drive A:	: Not Installed					
Floppy drive B:	: Not Installed					
Primary display	: Not Installed					
Keyboard	: Not Installed					

Month	: Jan, Feb, ... , Dec
Date	: 01, 02, 03, ... , 31
Year	: 1901, 1902, ... , 2099

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Date and Day Configuration

Move the cursor to the **Date** field with the cursor keys and set the date and day by pressing **<**PG UP**>** or **<**PG DN**>**.

Time Configuration

Move the cursor to the **Time** field with the cursor keys and set the time by pressing **<**PG UP**>** or **<**PG DN**>** to change values upward or downward.

Hard Disk Configuration

Hard disk drive types are identified by the following seven parameters:

- Drive Type
- Number of Cylinders

- **Number of Heads**
- **Write Pre-compensation**
- **Landing Zone**
- **Number of Sectors**
- **Capacity**

For drive types 1 through 46 the BIOS Setup utility automatically prescribes the remaining six parameters. The parameters for type 47 under hard disk C: and hard disk D: can be different. Drive type 47 thus allows two different user-defined hard disk drives to reside in the computer system. The user must enter the correct drive parameters when using type 47. Values for user-defined type hard disks may be easily determined using the **Auto Detect Hard Disk** option from the opening screen (described on page 50).

The Not Installed disk option should be selected for diskless microcomputer applications, diskless workstations, and SCSI hard drives.

Floppy

Floppy drive A: and floppy drive B: may each have one of five settings:

- **360 KB 5-1/4 inch**
- **1.2 MB 5-1/4 inch**
- **720 KB 3-1/2 inch**
- **1.44 MB 3-1/2 inch**
- **Not Installed** (used for diskless workstations)

Monitor

The **Primary display** settings are:

- **Monochrome**
- **Color 40x25**
- **VGA/PGA/EGA**
- **Color 80x25**
- **Not Installed** (used for network file servers)

Keyboard

The possible keyboard settings are **Installed** and **Not Installed**. If **Not Installed** is selected, the BIOS does not test for the presence of a keyboard in the system, permitting keyboardless systems such as embedded VMEbus controllers to boot normally.

Memory Display

The memory display is not configured by the user since the BIOS automatically senses and reports system memory. Memory is reported in 64 Kbyte increments. The BIOS will report up to 640 Kbytes in the **Base Memory** field and 65,472 Kbytes in the **Extended Memory** field. Extended memory is addressed from just above the 1 Mbyte Real Mode boundary to the 64 Mbyte level.

Advanced CMOS Setup

The **Advanced CMOS Setup** option of the BIOS Setup utility allows the user to configure more advanced components of memory configuration, peripheral support, and power management. **Advanced CMOS Setup** is the second option on the setup opening screen. When this option is selected, the screen shown in Figure 2-4 appears after the warning screen.

Figure 2-5 Advanced CMOS Setup Screen

AMIBIOS SETUP PROGRAM -- ADVANCED CMOS SETUP
(c) 1992 American Megatrends Inc., All Rights Reserved

Typematic Rate Programming	: Disabled	AT Bus Clock Source	: SCLK / 4
Typematic Rate Delay (msec)	: 500	Shadow RAM Cacheable	: Enabled
Typematic Rate (Chars / Sec)	: 15		
Above 1 MB Memory Test	: Disabled		
Memory Test Tick Sound	: Enabled		
Memory Parity Error Check	: Enabled		
Hit <Del> Message Display	: Enabled		
Hard Disk Type 47 RAM Area	: 0:300		
Wait for <F1> If Any Error	: Enabled		
System Boot Up Num Lock	: On		
Floppy Drive Seek At Boot	: Disabled		
System Boot Up Sequence	: A:, C:		
Internal Cache Memory	: Enabled		
Fast Gate A20 Option	: Enabled		
Video ROM Shadow C000, 32K	: Enabled		
Adaptor ROM Shadow C800, 32K	: Disabled		
Boot Sector Virus Protection	: Disabled		
Auto-Configuration	: Enabled		

Esc: Exit Arrow keys: Sel (Cntl)PgUp / PgDn: Modify F1: Help F2: Color
F5: Old Values F6: BIOS Setup Defaults F7: Power-On Defaults

Note

Note The values shown in Figure 2-4 represent the factory defaults.

The **Advanced CMOS Setup** is equipped with a series of help screens. Access help by pressing **<**F1**>** while the feature under question is selected. The help screen will display the settings available for a particular configuration feature and provide special help for some options.

Typematic Rate and Delay

Typematic Rate Delay and **Typematic Rate** control the speed at which a keystroke is repeated. When a key is pressed and held down, the character is displayed and begins to repeat after the time set by the **Typematic Rate Delay**. The actual rate of repeat is set by the **Typematic Rate** value. Available delay options are: 250, 500, 750, and 1000 milliseconds. Available rate options are: 6, 8, 10, 12, 15, 20, 24, and 30 characters per second. The **Typematic Rate Programming** option enables or disables typematic control. Note that some keyboards do not respond to typematic programming – on such keyboards, the typematic settings simply have no effect.

Above 1 MB Memory Test

The **Above 1 MB Memory Test** option, when enabled, will execute the POST memory routines on the RAM above 1 Mbyte (if present on the system). If disabled, the BIOS will only check the first 1 Mbyte of RAM. This option should normally be disabled on the VMIVME-7486 to avoid unwanted VMEbus accesses through the Protected Mode VMEbus Window, which exists just beyond available extended memory.

Memory Test Tick Sound

The **Memory Test Tick Sound** option will enable or disable the ticking sound during the memory test.

Memory Parity Error Checking

Note The **Memory Parity Error Checking** option should *ALWAYS BE ENABLED* to allow the VMIVME-7486 to process VMEbus interrupts.

The **Memory Parity Error Checking** option enables or disables parity error checking for all system RAM. This setting should always be enabled on the VMIVME-7486 in order to enable the NMI interrupts used for VMEbus interrupt processing.

Hit <Del> Message Display

Disabling the **Hit <Del> Message Display** option will prevent the message:

Hit <Del> if you want to run Setup

from appearing when the system boots. Even with this message disabled, the system still responds normally to the **<**DEL**>** key request for setup; it just doesn't display the message.

Hard Disk Type 47 RAM Area

In the Hard Disk Drive configuration in Standard CMOS Setup, there is a user-definable hard disk type (type 47) which can be used to configure a non-standard hard disk drive for either drive C: or D:. Ordinarily, the data for disk type 47 is stored in $0300 in lower system RAM, but it can be kept in the top 1 Kbyte of applications memory, starting at address $9FC00. These settings take effect only if the shadow RAM options are disabled.

Wait for <F1> If Any Error

Before the system boots, the BIOS executes POST (Power-On Self-Test), a series of system diagnostic routines. If any of these tests generate a nonfatal error and the system can still function, the BIOS will respond with an appropriate error message followed by:

Press <F1> to continue...

If the **Wait for <F1> if Any Error** option is disabled, nonfatal errors will not generate the above statement, but the BIOS will still display the appropriate error message. Thus, the nonfatal errors will not halt the POST to wait for a user response.

System Boot Up Num Lock

If the **System Boot Up Num Lock** option is disabled, the NUM LOCK on an enhanced keyboard will be off when the system is booted, causing the numeric keypad arrow keys to serve as cursor movement keys. The BIOS default is NUM LOCK on.

Floppy Drive Seek At Boot

The **Floppy Drive Seek at Boot** option determines whether or not a seek test is performed on the floppy drive(s) at system boot time. This feature is disabled by default to promote a fast boot and to decrease the possibility of damaging the floppy drive heads from excessive seek testing.

System Boot Up Sequence

The BIOS normally attempts to boot from floppy drive A: (if present), and if unsuccessful, it attempts to boot from hard disk drive C:. This sequence can be reversed. If set to C:, A:, the system attempts to boot from the hard drive C: then floppy drive A:. The latter speeds up the boot process slightly, but prevents booting from a floppy.

Internal Cache Memory

The **Internal Cache Memory** option enables or disables the 80486 processor's internal 8 Kbyte cache memory. The cache is enabled by default for faster processing speed.

Fast Gate A20

Fast Gate A20 controls the ability to access memory addresses above 1 Mbyte by enabling or disabling access to the processor address line A20. To remain PC/XT compatible and be able to access conventional memory (from 0 to 1024 Kbytes), address line A20 must always be low. Disable **Fast Gate A20** to force address line A20 low.

Some application programs enter and exit Protected Mode through the BIOS. Using the BIOS, all RAM accesses to extended memory addresses above 1 Mbyte require that Gate A20 be enabled via the keyboard controller chip. For this software, Gate A20 must be repeatedly enabled and disabled via the keyboard controller chip. Such mode switching is a slow process.

Fast Gate A20 provides an alternate, faster method of enabling and disabling Gate A20 found in the chip set and some software products. Enable this BIOS option to use Fast Gate A20. **Fast Gate A20** is enabled by default.

ROM Shadow

ROM shadow is a technique for relocating BIOS code from slower ROM to faster RAM. The BIOS is then executed from the faster RAM.

The BIOS permits shadowing of the 32 Kbyte video BIOS at $C0000 and also the 32 Kbyte option ROM space at $C8000. If a shadowing option is enabled, the code that resides in that particular segment of ROM will be shadowed to faster RAM. The 64 Kbyte system ROM at $F0000 is automatically shadowed. By default, **Video ROM Shadow** is enabled and **Adaptor ROM Shadow** is disabled.

Boot Sector Virus Protection

When the **Boot Sector Virus Protection** option is enabled, the VMIVME-7486 system BIOS intercepts calls to the disk subsystem that would overwrite the boot sector of the hard drive. When such a request is intercepted, the user will be prompted to confirm the action. This can help prevent the system from being infected by certain viruses that attempt to rewrite the boot sector, but it also slows down disk accesses slightly in order for double-checking. The option is therefore disabled by default.

Auto-Configuration

The **Auto-Configuration** feature simply enables or disables the **Auto Configuration with BIOS Defaults** option from appearing on the opening screen. The **Auto-Configuration** feature is enabled by default.

AT Bus Clock Source

The **AT Bus Clock Source** option controls the speed of the AT/ISA bus on the VMIVME-7486, including the video chip, drive controller, and I/O port hardware, but excluding the VMEbus interface. This option should not be changed from the default of SCLK/4. Note that the EXT16M setting is not valid on the VMIVME-7486 and should never be selected.

Shadow RAM Cache

This option enables or disables the ability of the 80486 processor to cache anything in shadow RAM. The shadow RAM cache is enabled by default.

Auto Configuration with BIOS Defaults

Selecting the **Auto Configuration with BIOS Defaults** option allows the setup program to reset all CMOS variables to their original factory values. The only CMOS variables not reset are the date, time, and drive types as set in the standard CMOS setup screen. A confirmation prompt allows the user to reconsider this option.

Note Even though the default BIOS setting for the External Cache Memory option is *ENABLED*, this option should *ALWAYS BE DISABLED* on VMIVME-7486 boards with no cache RAM installed.

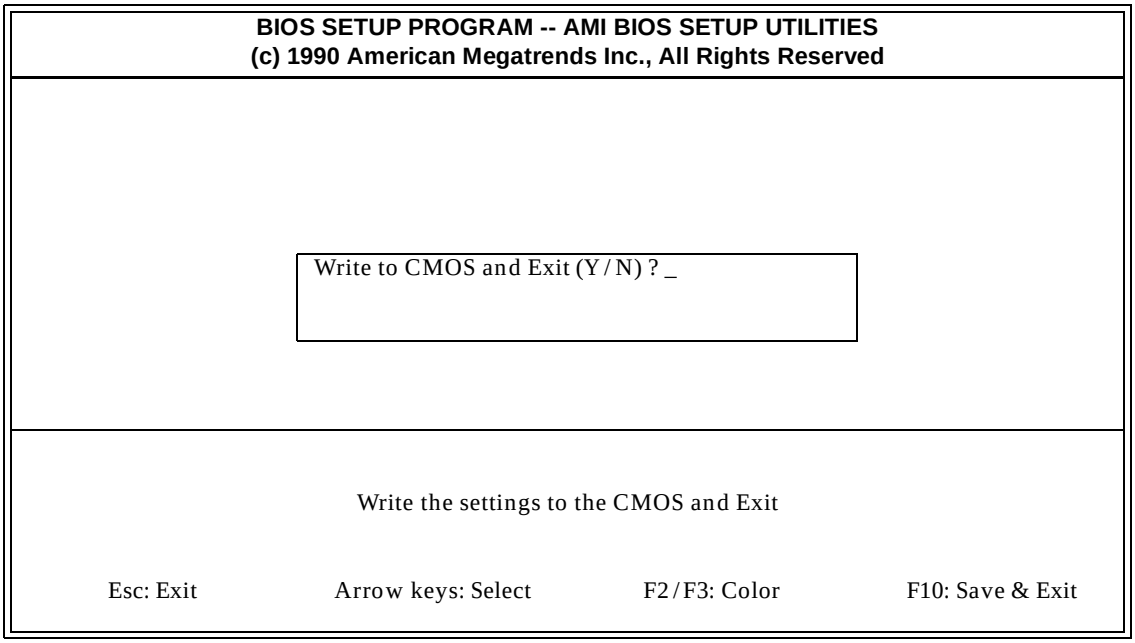
Auto Detect Hard Disk

Selecting the **`Auto Detect Hard Disk`** option causes the Setup program to attempt to determine the proper hard drive parameters for all existing hard drives. Upon finding a hard drive, the program displays the determined parameters and requests confirmation. If confirmed, the drive parameters are saved as listed. If the parameters are incorrect, they may be edited from the Standard CMOS Setup screen. Note that if either the primary or secondary hard drive is not present, the program will have to wait up to a minute to determine this fact.

Exiting BIOS Setup

There are two ways to exit the BIOS Setup program: **Write to CMOS and Exit** and **Do Not Write to CMOS and Exit**. When the **Write to CMOS and Exit Option** is selected, the Exit Screen shown in Figure 2-6 appears.

Figure 2-6 BIOS Setup Write to CMOS and Exit Screen



The features selected and configured in the Standard CMOS Setup and Advanced CMOS Setup are stored in CMOS memory when this exit option is selected. A CMOS RAM checksum is calculated and written to CMOS RAM. Control is then passed to the ROM BIOS.

Press **<**Y**>** and **<**ENTER**>** to save the system parameters, exit the setup program, and reboot the system. Press **<**N**>** and **<**ENTER**>** to return to the opening screen without saving any system parameters.

The **Do Not Write to CMOS and Exit** option passes control to the ROM BIOS without writing any changes to CMOS RAM.

Press **<**Y**>** and **<**ENTER**>** to exit the Setup utility without saving any system parameters. Press **<**N**>** and **<**ENTER**>** to return to the opening screen.

Configuring Operating Systems

The VMIVME-7486 is capable of running many popular operating systems, including MS-DOS, PC-DOS, DR-DOS, Microsoft Windows and Windows NT, IBM's OS/2, UNIX versions designed to run on Intel microprocessors, and UNIX variants such as XENIX. In general, any operating system designed to run on an Intel-based PC should work well on the VMIVME-7486.

The operating systems should generally be installed according to the publisher's instructions, preferably with VMEbus access disabled (see Chapter Four, *VMEbus Functions* on page 71). After installation, however, certain modifications to a standard installation of any of these operating systems may be necessary due to the special nature of the VMIVME-7486 as a combination PC and VMEbus controller.

General Rule Regarding Operating Systems

The VMIVME-7486 has two VMEbus Windows that should be off-limits to the operating system, the Real Mode VMEbus Window and the Protected Mode VMEbus Window.

General Rule Regarding Operating Systems

The most important modification that should be made to any operating system running on the VMIVME-7486 is to exclude the two VMEbus Windows from the operating system's memory map.

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

The complete VMIVME-7486 memory map is presented in Chapter Three, *PC/AT Functions* on page 55, but for reference the address ranges to be excluded for the VMEbus Windows are listed in Figure 2-5.

Table 2-5 VMEbus Window Addresses

VMEbus Window	Address Range to Exclude
Real Mode VMEbus Window	$E0000 - $EFFFF
Protected Mode VMEbus Window	$0200 0000 - $FFFD FFFF

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

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

Configuration Examples

Configuring MS-DOS for the VMIVME-7486

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

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

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

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

If EMS expanded memory support is desired, it is also useful to specify the 64 Kbyte page frame location, which should be at $D0000 on the VMIVME-7486. Specify the page frame for EMM386.EXE by adding the parameter `FRAME=D000`. The resulting EMM386.EXE line might then look something like this:

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

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

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

Configuring Windows for the VMIVME-7486

Microsoft Windows will run well on the VMIVME-7486, but the Real Mode VMEbus Window needs to be excluded from its memory manager. Edit the SYSTEM.INI file in the main Windows directory and add the following lines to the [386Enh] section:

EMMExclude=E000-EFFF

ReservedHighArea=E000-EFFF

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

CHAPTER

3

PC/AT Functions

Contents

CPU Register Model

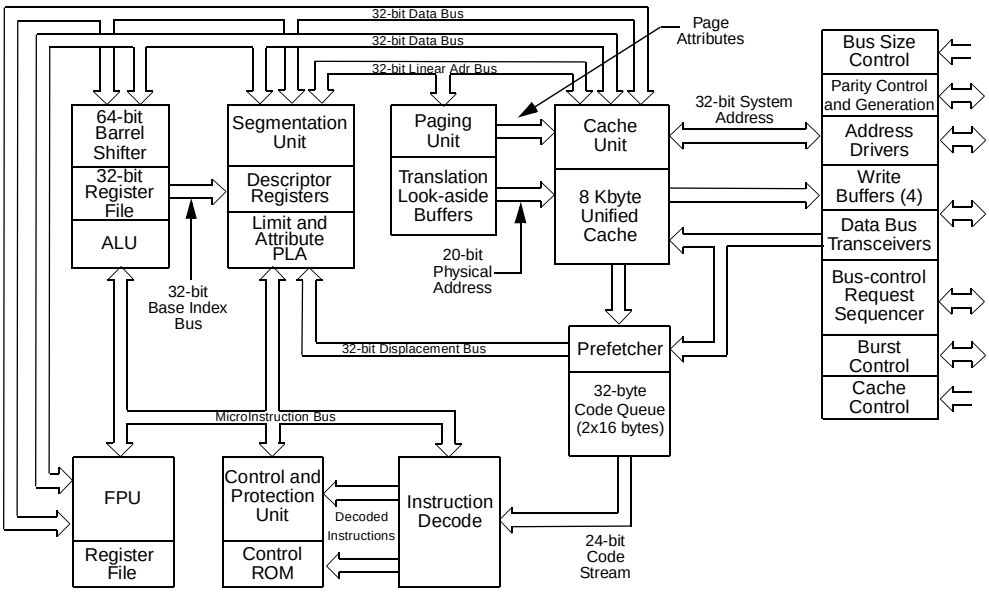


Figure 3-1 80486 CPU Block Diagram

Physical Memory

The VMIVME-7486 has four 30-pin SIMM sockets which support 256Kx9, 1Mx9 or 4Mx9 SIMM modules for a total of 1 MByte, 4 MBytes, or 16 MBytes of DRAM. All four memory modules must be of the same type and they should be rated at 70 ns or faster. Memory can be accessed as bytes, words, or longwords. Note that this memory is not dual-ported to the VMEbus. For applications requiring dual-ported access ask about the VMIVME-7487 board from VMIC.

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

The VMIVME-7486 includes both the system and video BIOS in a single 128K x 8 EPROM. The system portion of this ROM (at $F0000) is automatically shadowed, but for higher performance it is recommended that shadow RAM for the video BIOS be enabled from the Advanced CMOS Setup menu in BIOS Setup (see Chapter Two, *Installation and Setup* on page 29).

Memory and Port Maps

Memory Map

The real mode (20-bit) address memory map for the VMIVME-7486 is shown in Table 3-1. All systems share this same real mode memory map. The protected mode (32-bit) memory maps vary according to the amount of RAM on board. Table 3-2, Table 3-3, and Table 3-4 show the protected mode memory maps for the 1 MByte, 4 MByte, and 16 MByte RAM options, respectively.

Table 3-1 VMIVME-7486 Real Mode Memory Map

Mode	Size	Description	Address Range
REAL MODE	64 KBytes	ROM BIOS	$F0000 - $FFFFF
	64 KBytes	Real Mode VMEbus Window	$E0000 - $EFFFF
	64 KBytes	Reserved for Expansion BIOS (alternately may be used as a LIM / EMS page frame)	$D0000 - $DFFFF
	32 KBytes	Reserved for Expansion BIOS	$C8000 - $CFFFF
	32 KBytes	Video ROM	$C0000 - $C7FFF
	128 KBytes	Video RAM	$A0000 - $BFFFF
	640 KBytes	User RAM / DOS RAM	$00000 - $9FFFF

Table 3-2 VMIVME-7486 1 MByte RAM Protected Mode Memory Map

Mode	Size	Description	Address Range
PROTECTED MODE	64 KBytes	ROM BIOS Image	$FFFF 0000 - $FFFF FFFF
	64 KBytes	Reserved	$FFFE 0000 - $FFFE FFFF
	4078+ MBytes	Reserved (Images of the Protected Mode VMEbus Window)	$0110 0000 - $FFFD FFFF
	16 MBytes	Protected Mode VMEbus Window	$0010 0000 - $010F FFFF

Table 3-3 VMIVME-7486 4 MByte RAM Protected Mode Memory Map

Mode	Size	Description	Address Range
PROTECTED MODE	64 KBytes	ROM BIOS Image	$FFFF 0000 - $FFFF FFFF
	64 KBytes	Reserved	$FFFE 0000 - $FFFE FFFF
	4075+ MBytes	Reserved (Images of the Protected Mode VMEbus Window)	$0140 0000 - $FFFD FFFF
	16 MBytes	Protected Mode VMEbus Window	$0040 0000 - $013F FFFF
	3 MBytes	On-Board Extended Memory	$0010 0000 - $003F FFFF

Table 3-4 VMIVME-7486 16 MByte RAM Protected Mode Memory Map

Mode	Size	Description	Address Range
PROTECTED MODE	64 KBytes	ROM BIOS Image	$FFFF 0000 - $FFFF FFFF
	64 KBytes	Reserved	$FFFE 0000 - $FFFE FFFF
	4063+ MBytes	Reserved (Images of the Protected Mode VMEbus Window)	$0200 0000 - $FFFD FFFF
	16 MBytes	Protected Mode VMEbus Window	$0100 0000 - $01FF FFFF
	15 MBytes	On-Board Extended Memory	$0010 0000 - $00FF FFFF

I/O Port Map

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

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

The VMIVME-7486 incorporates all standard I/O peripherals of the PC/AT architecture such as keyboard, DMA, interrupt controllers, timers and real-time clock, as well as parallel and serial I/O ports, video registers, and floppy and hard drive task registers. The BIOS initializes and configures all these registers properly, so beware of adjusting these I/O ports directly. The assigned and user-available I/O addresses are summarized in the I/O Address Map, Table 3-5. Some I/O hardware can be addressed at several different locations determined by a hardware jumper, notably the parallel ports and drive controllers. See Chapter Two for details concerning hardware jumpers.

Table 3-5 VMIVME-7486 I/O Address Map

I/O Address Range	Size in Bytes	Hardware Device	PC/AT Function
$000 - $00F	16	ACC Micro 2168 Chip	DMA Controller 1 (Intel 8237A Compatible)
$010 - $01F	16		Reserved
$020 - $021	2	ACC Micro 2168 Chip	Master Interrupt Controller (Intel 8259A Compatible)
$022 - $03F	30		Reserved
$040 - $043	4	ACC Micro 2168 Chip	Programmable Timer (Intel 8254 Compatible)
$044 - $05F	30		Reserved
$060 - $064	5	ACC Micro 2168 Chip	Keyboard, Speaker, Eqpt. Config (Intel 8042 Compatible)
$065 - $06F	11		Reserved
$070 - $071	2	ACC Micro 2168 Chip	Real-time Clock, NMI Mask
$072 - $07F	14		Reserved
$080 - $08F	16	ACC Micro 2168 Chip	DMA Page Registers
$090 - $091	2		Reserved
$092	1	ACC Micro 2168 Chip	Alt. Gate A20 / Fast Reset Register
$093	1		Reserved
$094	1	Super VGA Chip	POS102 Access Control Register
$095 - $09F	11		Reserved
$0A0 - $0A1	2	ACC Micro 2168 Chip	Slave Interrupt Controller (Intel 8259A Compatible)
$0A2 - $0BF	30		Reserved
$0C0 - $0DF	32	ACC Micro 2168 Chip	DMA Controller 2 (Intel 8237A Compatible)

Table 3-5 VMIVME-7486 I/O Address Map (Continued)

I/O Address Range	Size in Bytes	Hardware Device	PC/AT Function
$0E0 - $0EF	16		Reserved
$0F0 - $0FF	16	ACC Micro 2168 Chip	Math Coprocessor / 2168 Config
$100 - $101	2		Reserved
$102	1	Super VGA Chip	POS102 Register
$103 - $13F	61		Reserved
$140 - $14F	16	Custom VMIC Hardware	VMEbus Interface Registers (see Chapter Four for details)
$150 - $16F	32		Reserved
$170 - $177	8	Super I/O Chip	Secondary Hard Disk Controller
$178 - $1EF	120		User I/O
$1F0 - $1F7	8	Super I/O Chip	Primary Hard Disk Controller
$1F8 - $277	128		User I/O
$278 - $27F	8	I/O Chip*	LPT2 Parallel I/O*
$280 - $2E7	104		Reserved by VMIC
$2E8 - $2EE	7	UART*	COM4 Serial I/O*
$2EF - $2F7	9		User I/O
$2F8 - $2FE	7	Super I/O Chip	COM2 High-Speed Serial I/O (16550A Compatible)
$2FF - $31F	33		Reserved by VMIC
$320 - $36F	80		User I/O
$370 - $377	8	Super I/O Chip	Secondary Floppy Disk Controller
$378 - $37F	8	Super I/O Chip	LPT1 Parallel I/O

Table 3-5 VMIVME-7486 I/O Address Map (Continued)

I/O Address Range	Size in Bytes	Hardware Device	PC/AT Function
$380 - $3AF	48		Reserved
$3B0 - $3BB	16	Monochrome Adapter* or Super VGA Chip	Monochrome Adapter* or VGA Monochrome Video
$3BC - $3BF	4	I/O Chip*	Alternate LPT1 Parallel I/O*
$3C0 - $3CF	16	Super VGA Chip	VGA Adapter
$3D0 - $3DF	16	CGA Adapter* or Super VGA Chip	CGA Adapter* or VGA Color Video
$3E0 - $3E7	8		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 (16450 Compatible)
$3FF	1		Reserved
ADDRESSES BEYOND $3FF ARE NOT USABLE BY EXPANSION BOARDS			
$4000 - $FFFF	49152		Reserved

* While these I/O ports are reserved for the listed functions, they are not implemented on the VMIVME-7486. They are listed here to make the user aware of the standard PC/AT usage of these ports.

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

PC/AT Interrupts

In addition to an I/O port address, an I/O device often has a separate hardware interrupt line assignment. Assigned to each interrupt line is a corresponding interrupt vector in the 256-vector interrupt table at $00000 to $003FF in memory. The 16 maskable interrupts and the single nonmaskable interrupt (NMI) are listed in Table 3-6 along with their functions. Table 3-7 on page 64 details the vectors in the interrupt vector table.

Table 3-6 PC/AT Hardware Interrupts

IRQ#	AT Function	Comments
NMI	Parity Errors (Must be enabled in BIOS Setup)	Used by VMIVME-7486 VMEbus Interface
0	System Clock/Calendar	Set by BIOS Setup
1	System Clock/Calendar	Set by BIOS Setup
2	Duplexed to IRQ9	
3	COM2 / COM4	
4	COM1 / COM3	
5	LPT2	
6	Floppy Controller	
7	LPT1	
8	Real-Time Clock	
9	Old IRQ2	VGA or Network I/O
10	Not Assigned	
11	Not Assigned	Used by VMIVME-7486 VMEbus Interface (primary choice, set by jumper J4/E4)
12	Not Assigned	Used by VMIVME-7486 VMEbus Interface (alternate choice, set by jumper J4/E4)
13	Math Coprocessor	
14	AT Hard Drive	
15	Not Assigned	

Table 3-7 PC/AT Interrupt Vector Table

Interrupt #		IRQ Line	Real Mode	Protected Mode
Hex	Dec			
00	0		Divide Error	Same as Real Mode
01	1		Debug Single Step	Same as Real Mode
02	2	NMI	Memory Parity Error, VMEbus Interrupts	Same as Real Mode (Must be enabled in BIOS Setup)
03	3		Debug Breakpoint	Same as Real Mode
04	4		ALU Overflow	Same as Real Mode
05	5		Print Screen	Array Bounds Check
06	6			Invalid OpCode
07	7			Device Not Available
08	8	IRQ0	Timer Tick	Double Exception Detected
09	9	IRQ1	Keyboard Input	Coprocessor Segment Overrun
0A	10	IRQ2	BIOS Reserved	Invalid Task State Segment
0B	11	IRQ3	COM2 Serial I/O	Segment Not Present
0C	12	IRQ4	COM1 Serial I/O	Stack Segment Overrun
0D	13	IRQ5	Fixed Disk Controller	General Protection Violation
0E	14	IRQ6	Floppy Disk Controller	Page Fault
0F	15	IRQ7	LPT1 Parallel I/O	Unassigned
10	16		BIOS Video I/O	Coprocessor Error
11	17		Eqpt Configuration Check	Same as Real Mode
12	18		Memory Size Check	Same as Real Mode
13	19		XT Floppy/Hard Drive	Same as Real Mode
14	20		BIOS Comm I/O	Same as Real Mode
15	21		BIOS Cassette Tape I/O	Same as Real Mode
16	22		BIOS Keyboard I/O	Same as Real Mode
17	23		BIOS Printer I/O	Same as Real Mode
18	24		ROM BASIC Entry Point	Same as Real Mode
19	25		Bootstrap Loader	Same as Real Mode
1A	26	IRQ8	Real-time Clock	Same as Real Mode
1B	27		Control/Break Handler	Same as Real Mode

Table 3-7 PC/AT Interrupt Vector Table (Continued)

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

Table 3-7 PC/AT Interrupt Vector Table (Continued)

Interrupt #		IRQ Line	Real Mode	Protected Mode
Hex	Dec			
73	115	IRQ11	VMEbus Interrupts (primary)	Same as Real Mode
74	116	IRQ12	VMEbus Interrupts (alternate)	Same as Real Mode
75-7F	117-127		Reserved by DOS	Same as Real Mode
80-F0	128-240		Reserved for BASIC	Same as Real Mode
F1-FF	241-255		Reserved by DOS	Same as Real Mode

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

The interrupt hardware implementation on the VMIVME-7486 is standard for computers built around the PC / AT architecture, which evolved from the IBM PC / XT. In the IBM PC / XT computers only eight interrupt request lines exist, numbered from IRQ0 to IRQ7 at the PIC. The IBM PC / AT computer added eight more IRQx lines, numbered IRQ8 to IRQ15, by cascading a second slave PIC into the original master PIC. IRQ2 at the master PIC was committed as the cascade input from the slave PIC. IRQ13 at the slave is reserved for the math coprocessor.

To maintain backward compatibility with PC / XT systems, IBM chose to use the new IRQ9 input on the slave PIC to operate as the old IRQ2 interrupt line on the PC / XT Expansion Bus. Thus, in AT systems, the IRQ9 interrupt line connects to the old IRQ2 pin (pin B4) on the AT Expansion Bus (or ISA bus). This is the same implementation used on the VMIVME-7486.

Figure 3-2 depicts the VMIVME-7486 interrupt logic pertaining to VMEbus operations. Note that jumper E12 selects either IRQ11 or IRQ12 for receiving VMEbus related interrupts. Also note that the NMI interrupt generated by the VMIVME-7486 must be enabled in the BIOS Setup (see Chapter Two, *Installation and Setup* on page 29).

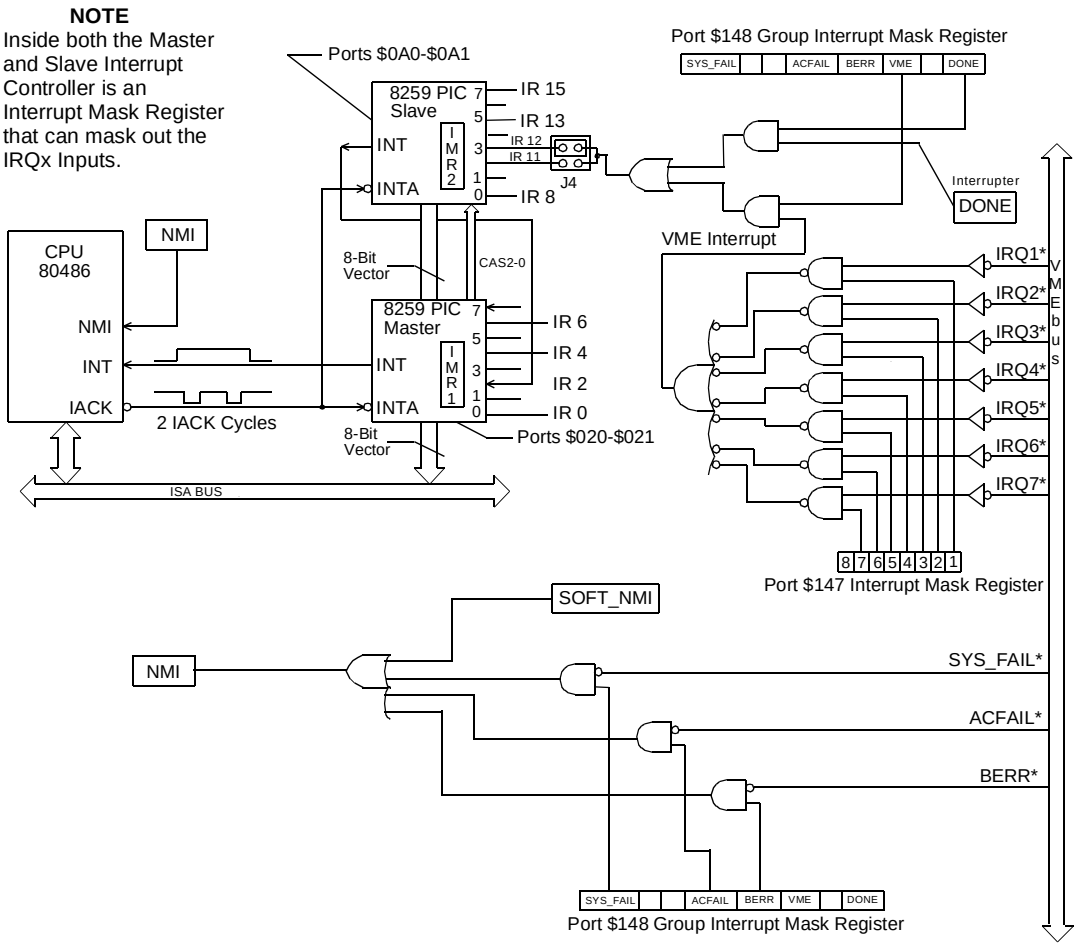


Figure 3-2 Interrupt Logic Diagram

I/O Ports

The VMIVME-7486 incorporates the Winbond W83757AF Super I/O chip. This chip provides the VMIVME-7486 with a standard floppy drive controller, IDE hard drive interface, two 16450 compatible serial ports, and one standard Centronics parallel port. All ports are present in their standard PC/AT locations, using default interrupts.

Video Graphics Adapter

The monitor port on the VMIVME-7486 is controlled by a Cirrus Logic CL-GD5420 chip with 1 Mbyte video DRAM. The video controller chip is hardware and BIOS compatible with the IBM EGA and VGA standards and also supports VESA high-resolution and extended video modes. Table 3-8 shows the popular graphics video modes supported by the Cirrus Logic video chip.

Table 3-8 Common Supported Graphics Video Resolutions

Screen Resolution	Maximum Colors	Refresh Rates
640 x 480	up to 256	60 Hz, 72 Hz
800 x 600	up to 256	56 Hz, 60 Hz, 72 Hz
1024 x 768 (Interlaced)	up to 256	43.5 Hz (Interlaced to 87 Hz)
1024 x 768	up to 256	60 Hz, 70 Hz

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

The VMIVME-7486's processor has 16 bit access to video memory with no wait states. Video I/O registers are accessed at the maximum ISA bus rate. In addition, the Cirrus Logic video controller supports up to a 64 x 64 pixel hardware cursor.

CHAPTER

4

VMEbus Functions

Contents

VMEbus Access

This section discusses how the VMIVME-7486 interfaces with the VMEbus. At power up or after a system reset (but not after a soft reset, or <CTRL+ALT+DEL> reset) the VMIVME-7486 is "isolated" from the VMEbus. The user must first set a particular bit in the VMEbus Access Control Register before access to the VMEbus is possible so that the VMIVME-7486 board does not interfere with other VMEbus modules or inadvertently write to VMEbus memory during initialization and POST.

Once the operating system is loaded into the VMIVME-7486, the user may proceed with VMEbus activities from a known state and in an orderly manner.

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

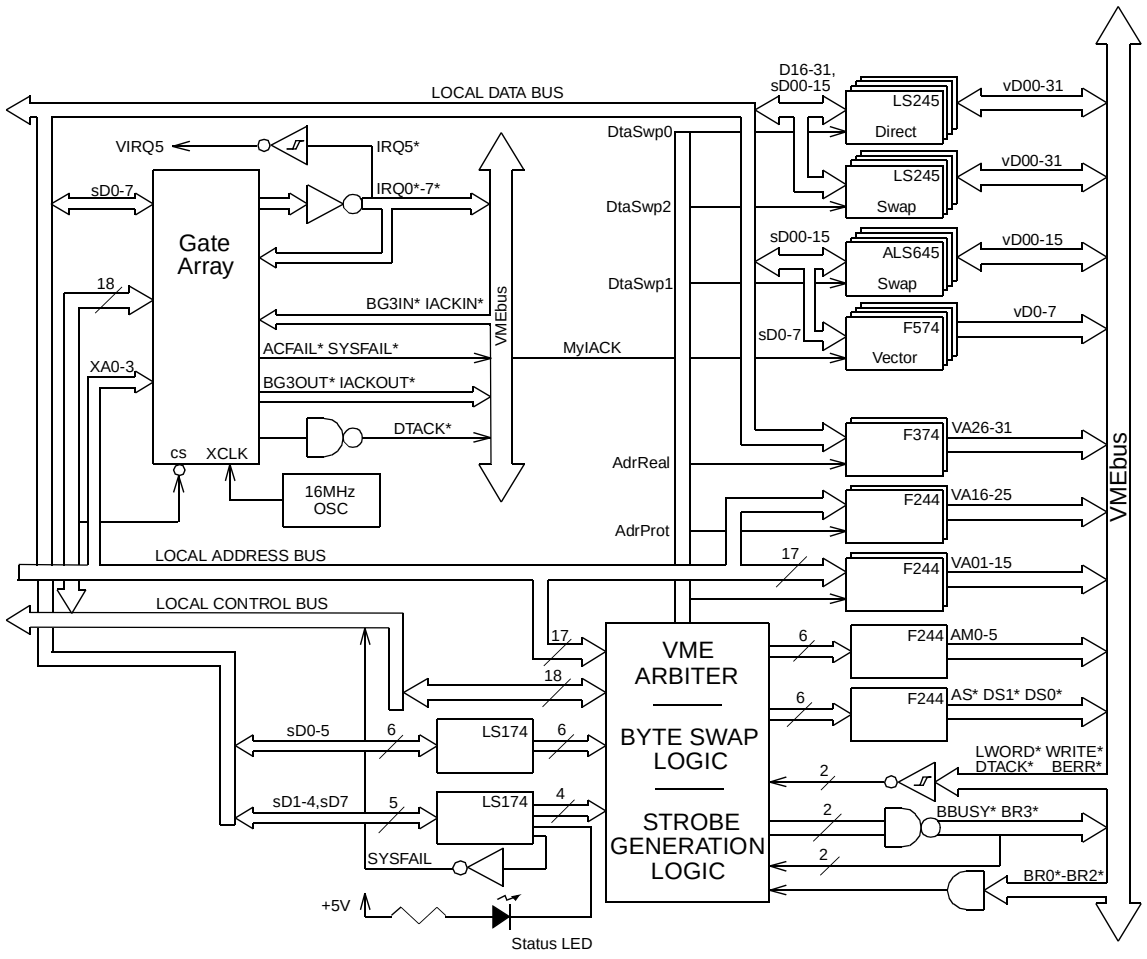


Figure 4-1 VMIVME-7486 VMEbus Interface Block Diagram

The VMIVME-7486 can access the VMEbus in Real Mode or Protected Mode. As a VMEbus interrupt processor the VMIVME-7486 can also perform IACK* cycles. Figure 4-1 shows the functional interconnections between the VMIVME-7486 and the VMEbus.

In Real Mode the VMIVME-7486 uses a paging scheme to access any part of the 4 Gbytes of VMEbus addressing space, 64 Kbytes at a time, through its Real Mode VMEbus Window at $E0000. The user can set the VMEbus Address Modifier lines (AM5-AM0) as required to define the Data Transfer Cycle Address Mode as Short (16-bit / A16), Standard (24-bit / A24), Extended (32-bit / A32), or User Defined. VMEbus address bits beyond the 16 available within the Real Mode VMEbus Window may be controlled by the VMEbus Standard Access Register (for bits A16-A23) and the VMEbus Extended / AM Access Register (for bits A24-A31 as well as the Address Modifier bits).

In Protected Mode the VMIVME-7486 can access the VMEbus through the Protected Mode VMEbus Window, which occupies 16 Mbytes beginning at $0200 0000. By supplying the extended VMEbus address bits A24-A31 in the VMEbus Extended / AM Access Register, all 4 Gbytes of VMEbus memory may be addressed in Protected Mode, 16 Mbytes at a time.

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

Byte Ordering

Byte Swapping

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

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

Table 4-1 VMEbus Byte Assignment to the Data Lines

DTB Cycle Type	D31--D24	D23--D16	D15--D08 Even Adr	D07--D00 Odd Adr
D08(EO) Even or Odd				
Single Odd Byte(3)				Byte(3)
Single Even Byte(2)			Byte(2)	
Single Odd Byte(1)				Byte(1)
Single Even Byte(0)			Byte(0)	
D08(O) Odd Only				
Single Odd Byte(3)				Byte(3)
Single Odd Byte(1)				Byte(1)
D16				
Double Byte(2-3)			Byte(2)	Byte(3)
Double Byte(0-1)			Byte(0)	Byte(1)
D32				
Quad Byte(0-3)	Byte(0)	Byte(1)	Byte(2)	Byte(3)

It is important to note the major byte ordering differences between the VMEbus and the Intel 80486 CPU. In addition, communication between Motorola compatible 680X0 VMEbus modules and the VMIVME-7486 requires special attention to avoid byte ordering conflicts.

The Byte Swapping Problem Defined

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

The VMIVME-7486 uses an Intel 80486 microprocessor, which uses Little Endian byte ordering. Byte arrangement and the byte relationship between data in the processor and transferred data in memory are shown in Figure 4-2.

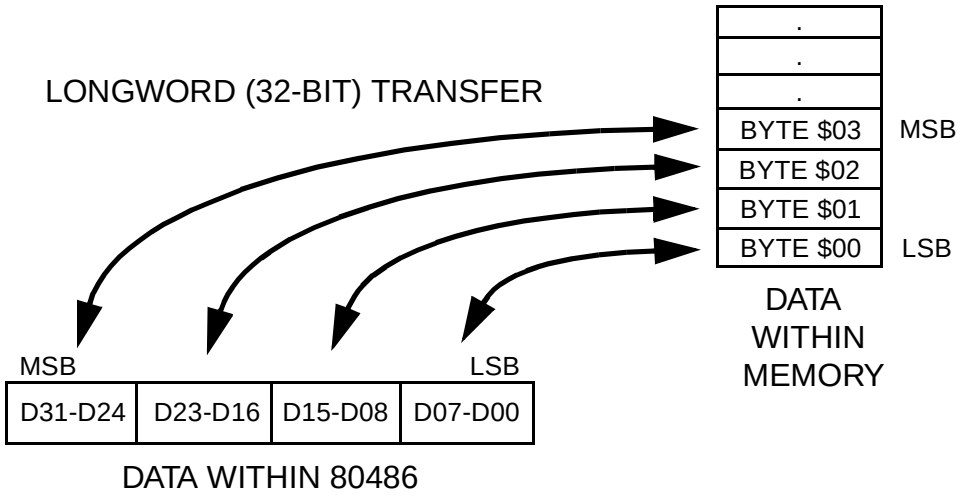


Figure 4-2 Byte Relationships using the Little Endian 80486

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

Contrast the behavior of the Little Endian 80486 in Figure 4-2 with the same longword transfer using a Big Endian processor like the Motorola 68040 in Figure 4-3 on page 76.

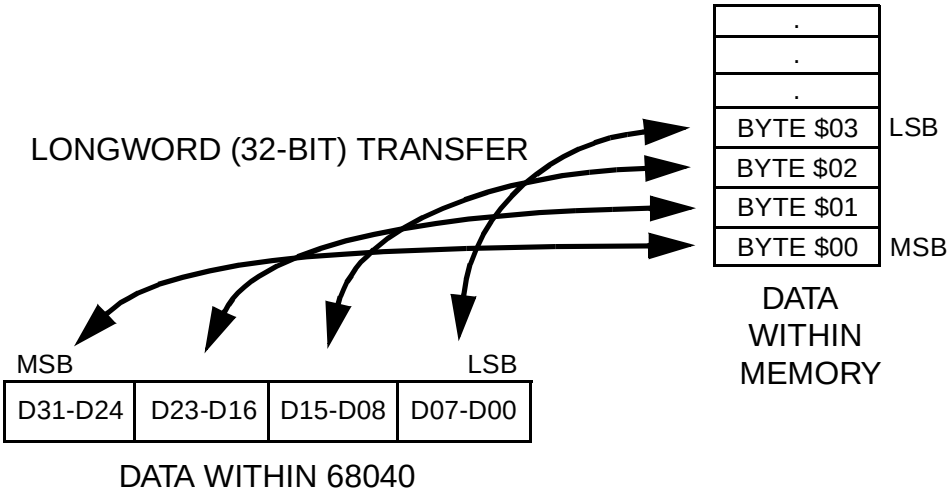


Figure 4-3 Byte Relationships using the Big Endian 68040

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

Byte Swapping and the VMEbus

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

If a Little Endian 80486 were to have its data bus directly connected to the VMEbus (i.e. D31 to D31, D30 to D30, etc.), then the most significant byte data supplied to the VMEbus D31-D24 byte lane during a longword write would be stored by the VMEbus in the lowest of the four destination byte addresses – opposite that expected by the 80486 (see Figure 4-2 on page 75). This poses no problem if the 32-bit value written is always read back using a similar longword transfer (i.e. all four bytes at once), since the swapped data gets swapped again and appears to the 80486 exactly as it should. However, if the data written by the 32-bit longword transfer were to be retrieved using any other method – say, using four separate byte transfers – a problem results, since the data at the lowest byte address would be incorrectly assumed to be the least significant, while it is actually the most significant.

The problem cannot be solved by simply connecting the 80486 to the VMEbus with its byte lanes crossed. For example, the 80486 uses D0-D7 to transfer a byte to address $00, while the VMEbus requires D8-D15 be used. For this reason, special hardware has been incorporated into the VMIVME-7486 to facilitate different kinds of byte swapping for varying circumstances.

VMIVME-7486 Byte Swapping Hardware

The VMIVME-7486 employs three-way byte swap buffers to accommodate the byte ordering inconsistencies between the 80486 and the VMEbus, but the programmer must still be aware of the problem and decide when to use the built-in byte swapping function. Figure 4-4 on page 77 diagrams the function of these swap buffers.

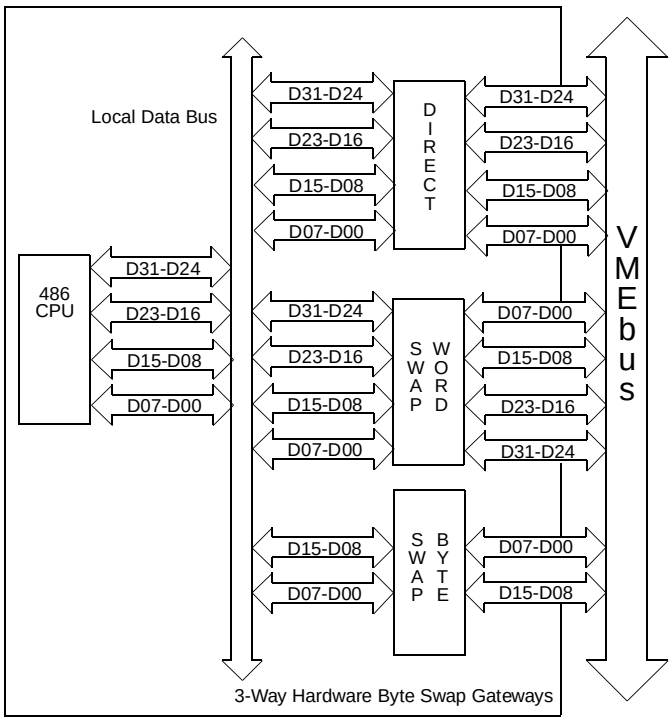


Figure 4-4 80486-to-VMEbus Data Byte Lanes

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

Table 4-2 Byte Swap Modes

Big Endian Bit Status	Size of Transfer	Swap Mode
X	single byte	byte-swap
0	word (two bytes)	byte-swap
0	longword (four bytes)	word-swap
1	word (two bytes)	direct
1	longword (four bytes)	direct

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

The byte-swapping hardware on the VMIVME-7486 is not a complete solution to the byte ordering problem in itself, since the programmer must still decide when to use each of the available swap schemes. Once decided and configured, however, the hardware does relieve the programmer from tediously swapping the bytes in software.

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

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

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

Interrupt Handling

The VMIVME-7486 has the ability to recognize and respond to VMEbus interrupt signals IRQ1*-IRQ7*. In addition, the 80486 CPU can be interrupted by other sources associated with the VMEbus interface.

There are six possible VMEbus-related interrupts on the VMIVME-7486: SOFT_NMI, ACFAIL, SYSFAIL, BERR, VMEbus, and DONE. The first four of these (SOFT_NMI, ACFAIL, SYSFAIL, and BERR) interrupt the 80486 using the processor's NMI, allowing normal interrupt processing to be interrupted in the event of a serious error. Specifically, the PC/AT IOCHK signal is activated which in turn produces an NMI. The VMEbus and DONE conditions interrupt the CPU using the INT signal via the PC/AT IRQ 11 or IRQ 12 signal (jumper selectable – the default is IRQ 11). SOFT_NMI allows the programmer to produce an NMI under software control.

Nonmaskable Interrupts

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

Note The Memory Parity Error Check feature must be enabled in the **Advanced CMOS Setup** screen for any NMI interrupts to occur. See the BIOS Setup section on page 38 in Chapter Two for details on changing this setting.

Soft NMI

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

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

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

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

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

ACFAIL, SYSFAIL, and BERR NMIs

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

The VMEbus SYSFAIL* signal is used to alert the system of an abnormal condition in the VMEbus chassis. When enabled, the VMEbus SYSFAIL* signal is gated directly to the processor's NMI input and is not edge-latched. Like ACFail above, however, the SysFail bit in the Group Interrupt Mask Register must be cleared upon completion of the ISR to enable future SYSFail interrupts. It is recommended that the SysFail interrupt service routine turn on the front panel LED as one of its first operations and turn it off as one of the last operations. Note also that the VMIVME-7486 can cause the SYSFAIL* signal to activate through the VMEbus Access Control Register.

The BERR (Bus Error) NMI alerts the user that the VMEbus BERR* signal was asserted during a VMEbus transfer, although the transfer that caused the bus error was not necessarily initiated by the VMIVME-7486. The BERR NMI is caused by any low-to-high transition of the VMEbus BERR* signal and should be cleared upon completion of the ISR by setting the BERR bit in the Clear Condition Register.

NMI Processing

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

If the VMIVME-7486 is to handle VMEbus system interrupts using the NMI, however, the default NMI vector needs to be modified to point to the user's own NMI routine. This new ISR should query the Group Interrupt Status Register to determine the cause of the NMI. If the cause is either one of the VMEbus system interrupts described above or the Soft NMI, appropriate response should be taken. The condition should then be cleared and the routine should exit. If the NMI is not caused by a special source, however, the custom ISR should pass control back to the original BIOS ISR to ensure normal PC / AT system error handling.

Maskable Interrupts

The DONE condition or any VMEbus interrupt request can interrupt the processor using IRQ 5, IRQ 11, or IRQ 12, depending upon a system jumper (See Chapter Two, *Installation and Setup* on page 29). Table 4-5 on page 82 shows a typical VMEbus interrupt handler initialization and response using IRQ 12. The corresponding PC / AT interrupt vectors are located in memory at $1CC (IRQ 11) or $1D0 (IRQ 12). Whichever interrupt is used should have its vector modified to point to the user's custom VMEbus / DONE interrupt handler.

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

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

For VMEbus interrupts, reading the IRQ Status-ID register acknowledges the highest priority interrupt pending that is unmasked, from IRQ 7 (highest priority) to IRQ 1 (lowest priority). For example, if both VMEbus IRQ 1 and IRQ 3 are enabled on the VMIVME-7486 through the Interrupt Mask Register and both VMEbus interrupts occur simultaneously, the IRQ Status-ID read acknowledges only VMEbus IRQ 3.

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

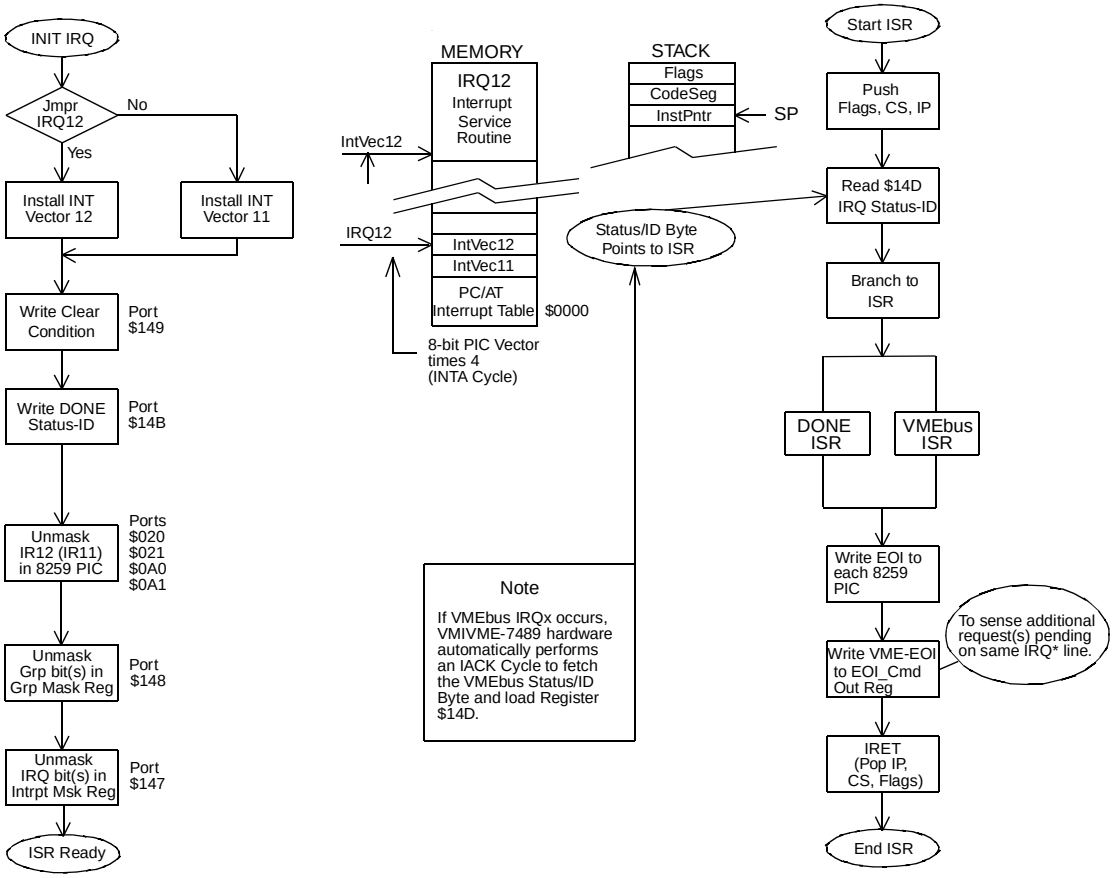


Figure 4-5 Interrupt Response Flow Chart

Condition Polling

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

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

In some systems, it may also be advantageous to poll VMEbus interrupt signals that are masked off. Thus, a VMIVME-7486 might monitor another VMEbus module's activity without actually performing a VMEbus access. The state of the VMEbus IRQ1*-IRQ7* signals can be read through the IRQ* Status Register. If a VMEbus IRQ* signal is asserted while it is masked off at the Interrupt Mask Register, the condition can only be cleared by accessing the VMEbus interrupter that activated the signal, since reading the IRQ Status-ID Register will *not* clear the condition in this case. Only the VMEbus IRQ* signals enabled to cause VMIVME-7486 interrupts are affected by reading the IRQ Status-ID Register.

Interrupter

The VMIVME-7486 can interrupt another interrupt handler on the VMEbus on one of IRQ7*-IRQ3* lines. The line selection is programmed by writing an IPL2-IPL0 3-bit binary encoded value to the Interrupt Level Select Register. Thereafter, when the CPU writes to the Interrupter Status-ID Register, a VMEbus interrupt is generated on the selected IRQ* line. The interrupt handler acknowledges by reading the Status / ID byte just written.

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

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

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

Requester

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

The ROR bit (bit 4) in the VMEbus Access Control Register defines how the VMIVME-7486 will release the VMEbus. Its power up default is clear, indicating RWD (Release-When-Done) operation. The bus will be released after every transfer. When the ROR bit in the VMEbus Access Control Register is set, the VMIVME-7486 enters Release-On-Request mode and will not release the bus and de-assert BBUSY* until another master has requested the bus. It does this by monitoring all four Bus Request lines BR3*-BR0* to determine if another master is requesting the bus.

The ROR mode is the preferred option when the VMIVME-7486 is the main CPU in the VMEbus. ROR mode decreases arbitration overhead for each transfer, resulting in improved performance.

System Controller

For VMEbus slot-1 applications the VMIVME-7486 may be jumpered to perform system controller functions including:

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

The bus error timer generates a BERR* signal when a nonexistent or nonresponding device is addressed on the VMEbus. The maximum response time for the on-board bus timer is factory set at about 10 µs. The bus error timer function may be disabled by a jumper (see Chapter Two, *Installation and Setup* on page 29 for hardware details). For systems in which the VMIVME-7486 bus timer is disabled, the bus timeout must be provided by another device in the VMEbus chassis. In that case, the maximum bus timeout value for reliable VMIVME-7486 operation is 8 ms. This value is dictated by the DRAM refresh time for the VMIVME-7486's memory. Timeouts beyond the 8 ms limit risk memory corruption due to lack of refresh.

Note that the VMIVME-7486 generates a VMEbus SYSRESET* signal only if it is configured as a system controller, and then only after power up or when the front panel reset switch is pressed. SYSRESET* is *not* generated from the "soft boot" caused by a **<**CTRL+ALT+DEL**>** keyboard reset.

See Chapter Two, *Installation and Setup* on page 29 for information on jumpering the VMIVME-7486 as a system controller.

System Registers

Register Map

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

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

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

Table 4-3 VMIVME-7486 VMEbus Register Map

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

Table 4-3 VMIVME-7486 VMEbus Register Map (Continued)

Register Name	Access	I/O Adr	D7	D6	D5	D4	D3	D2	D1	D0
Interrupter Status-ID	W	$14F	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

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

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

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

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

Register Details

VMEbus Access Control Register

The VMEbus Access Control Register is a read / write byte register at I / O address $140. The five active bits in this register control certain global aspects of VMEbus activity. All active bits in this register are cleared on power up or hard reset.

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

VMEbus Access Control Register: Status LED bit

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

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

VMEbus Access Control Register: ROR bit

When the VMIVME-7486 requests bus mastership, the on-board requester initiates a VMEbus request and eventually gets a Bus Grant. The requester then signals the VMIVME-7486 to take control of the VMEbus.

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

When the data transfer task is finished, the requester has two ways to release control of the bus. Bit 4 specifies whether to release control of the bus immediately upon completion (Release When Done, or RWD) or to maintain control of the bus until another master requests control (Release On Request, or ROR). RWD mode is the default for maximum compatibility with other masters, but if the VMIVME-7486 is the only master, or is usually the master, then ROR mode can speed up repeated VMEbus accesses since less overhead is involved every transfer. See Section in this chapter for a more detailed discussion of the VMIVME-7486 requester function.

VMEbus Access Control Register: SysFail bit

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

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

Since other controllers on the VMEbus may not function normally while the SYSFAIL* line is active, VMIC recommends that the SysFail bit be set immediately after power up or reset initialization routines, even if no VMEbus activity is planned. This is especially important if the VMIVME-7486 is not the slot-1 controller, since an active SYSFAIL* line may prevent the controller from performing its normal functions.

Note

Note The VMIVME-7486 activates the VMEbus SYSFAIL* signal upon power up or reset. It is important to de-activate SYSFAIL* as soon as possible to allow normal VMEbus activity.

VMEbus Access Control Register: Big Endian bit

The 80486 processor uses Intel's Little Endian method for transferring a multiple-byte group between its internal registers and external memory. The VMEbus, however, expects Motorola's Big Endian method for transferring a multiple-byte group to the same locations. The VMIVME-7486 has hardware on-board to support either transfer method, and the byte-swapping hardware is controlled by bit 2 of this register.

D2	Big Endian Bit Function
0	Use Little Endian multiple-byte transfers (default)
1	Use Big Endian multiple-byte transfers

Any byte swapping is transparent both to the on-board CPU and to VMEbus devices. See the discussion on byte swapping in Section 2 for more details.

VMEbus Access Control Register: VME Enable bit

Bit 1 controls VMEbus access. While this bit is clear – the default state after power up or reset – the VMIVME-7486 cannot access the VMEbus. While VMEbus access is disabled, the VMIVME-7486 VMEbus interface registers may still be accessed, but no data transfers to or from the VMEbus will occur. Likewise, no VMEbus signal can cause an interrupt on the VMIVME-7486.

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

The VMIVME-7486 is never completely isolated from the VMEbus, however, regardless of the VME Enable bit. The SysFail bit in this same register always functions. Also, the VMIVME-7486 will still generate a VMEbus SYSRESET* when the front panel reset button is pressed, as long as the VMIVME-7486 is configured as the system controller. Likewise, if the VMIVME-7486 is not the system controller, it will still respond to a VMEbus SYSRESET* regardless of the VME Enable bit status.

VMEbus Standard Access Register

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

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

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

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

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

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

VMEbus Extended/AM Access Register

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

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

This register contains both the extended VMEbus address lines A24-A31 as well as the VMEbus AM Code. The Address Modifier bits AM0-AM5 here correspond directly to the VMEbus AM0-AM5 bits. The AM Code determines the type of transfer the VMIVME-7486 will perform: Extended, Standard, Short, or User Defined. The AM code must *always* be initialized to a value corresponding to the desired VMEbus access mode. The VMEbus Specification has a complete list of Address Modifier codes, but Table 4-4 below lists the most common and useful codes.

Table 4-4 Valid VMEbus AM Codes

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

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

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

Interrupt Mask Register

The Interrupt Mask Register is a read/write byte register at I/O address $147. The seven active bits in this register can prevent individual VMEbus interrupt lines from interrupting the VMIVME-7486. All active bits in this register are cleared on power up or hard reset.

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

This register is used to individually mask any of the seven VMEbus interrupts to the interrupt handler on the VMIVME-7486. A clear bit indicates that the VMIVME-7486 will not respond to that particular VMEbus interrupt.

Interrupt acknowledgment is prioritized such that if more than one IRQ* line is enabled, the VMIVME-7486 acknowledges the higher priority interrupt first, with IRQ7* having the highest priority and IRQ1* the lowest.

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

Group Interrupt Mask Register

The Group Interrupt Mask Register is a read / write byte register at I / O address $148. The five active bits in this register disable individual VMEbus system interrupts from interrupting the VMIVME-7486. All active bits in this register are cleared on power up or hard reset.

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

When clear, each active bit in this register prevents its associated interrupt source from interrupting the VMIVME-7486 as shown below:

SysFail	When set, this bit enables the VMEbus SYSFAIL* signal to reach the NMI input to the 80486 CPU.
ACFAIL	When set, this bit enables the gating of the VMEbus ACFAIL* line onto the NMI input to the 80486 CPU.
BERR	When set, this bit enables the VMEbus BERR bus error interrupt to reach the NMI input to the 80486 CPU as a result of a BERR* cycle or bus timeout.
VMEbus	When set, this bit enables IRQ1*-IRQ7* interrupts from the VMEbus (although they may also be controlled individually by the Interrupt Mask Register described previously).
DONE	When set, this bit enables the DONE interrupt to reach the NMI input to the 80486 CPU when the on-board interrupter has its interrupt acknowledged.

Clear Condition Register

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

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

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

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

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

DONE Status-ID Register

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

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

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

Group Interrupt Status Register

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

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

The Group Interrupt Status Register allows the processor to determine the source of the pending interrupts and also allows the associated VMEbus conditions to be polled. A set bit indicates the corresponding condition is present, although the actual interrupt may be masked off. See *Interrupt Handling* on page 79 of this chapter for details concerning interrupt handling and this register.

Soft NMI Select Register

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

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

IRQ Status-ID Register

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

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

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

EOI Command Register

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

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

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

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

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

Interrupt Level Select Register

The Interrupt Level Select Register is a read / write byte register at I / O address $14E. The four active bits in this register control the level of the VMEbus interrupts generated by the VMIVME-7486. The value of all bits in this register are indeterminate after power up or hard reset.

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

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

Table 4-5 IRQ Level IPL Values

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

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

IRQ* Status Register

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

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

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

Interrupter Status-ID Register

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

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

This register is used by the on-board interrupter and supplies the status-ID byte to a VMEbus interrupt handler during an IACK* cycle. The VMIVME-7486 Interrupter is a ROAK (Release On Acknowledge) interrupter, so the VMIVME-7486 releases the IRQ* line when the handler acknowledges its interrupt by reading this value.

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

APPENDIX

A

Ethernet Option

Contents

Introduction

The VMIVME-7486 PC / AT compatible VMEbus controller board may be ordered with the Ethernet Mezzanine option. The Ethernet Mezzanine provides for local area network access by the VMIVME-7486 controller by means of a standard Ethernet AUI port on its front panel. A customer-supplied AUI adapter connects to the front panel to provide the final interface to the physical network, which may 10BASE5 (Thick Coax), 10BASE2 (Thin Coax), or 10BASE-T (Twisted Pair). The Ethernet Mezzanine provides an EPROM socket that provides the ability to boot directly from the network with the proper software.

Figure A-1 on page 101 shows the factory installed Ethernet Mezzanine on the VMIVME-7486 controller board. Note that the Ethernet Mezzanine AUI connector on the front panel of a VMIVME-7486 replaces the COM1 serial connector, making COM1 unavailable on controllers with the Ethernet Mezzanine installed. Software may still detect a COM1 port, since the UART is still present, but without an external connector the COM1 port is useless.

Ethernet Mezzanine Software Compatibility

The Ethernet Mezzanine is based on National Semiconductor's DP83905 AT/LANTIC VLSI chip. This device is software compatible with Novell's "NE2000" standard. Any software that can be configured to support an NE2000 compatible card should execute correctly on the VMIVME-7486 with the Ethernet Mezzanine installed.

Ethernet Mezzanine Driver Software

Driver software must be provided by the customer's own Network Software Supplier for use with the Ethernet Mezzanine. The following popular drivers should work well:

1. Novell Netware 2.15 (IPX)
2. Novell Netware 3.11 (ODI)
3. Microsoft Lan Manager 2.0 (NDIS)
4. Microsoft Lan Manager Boot ROM
5. FTP Packet Driver
6. SCO Unix 386 V5
7. NVMIVME-7489

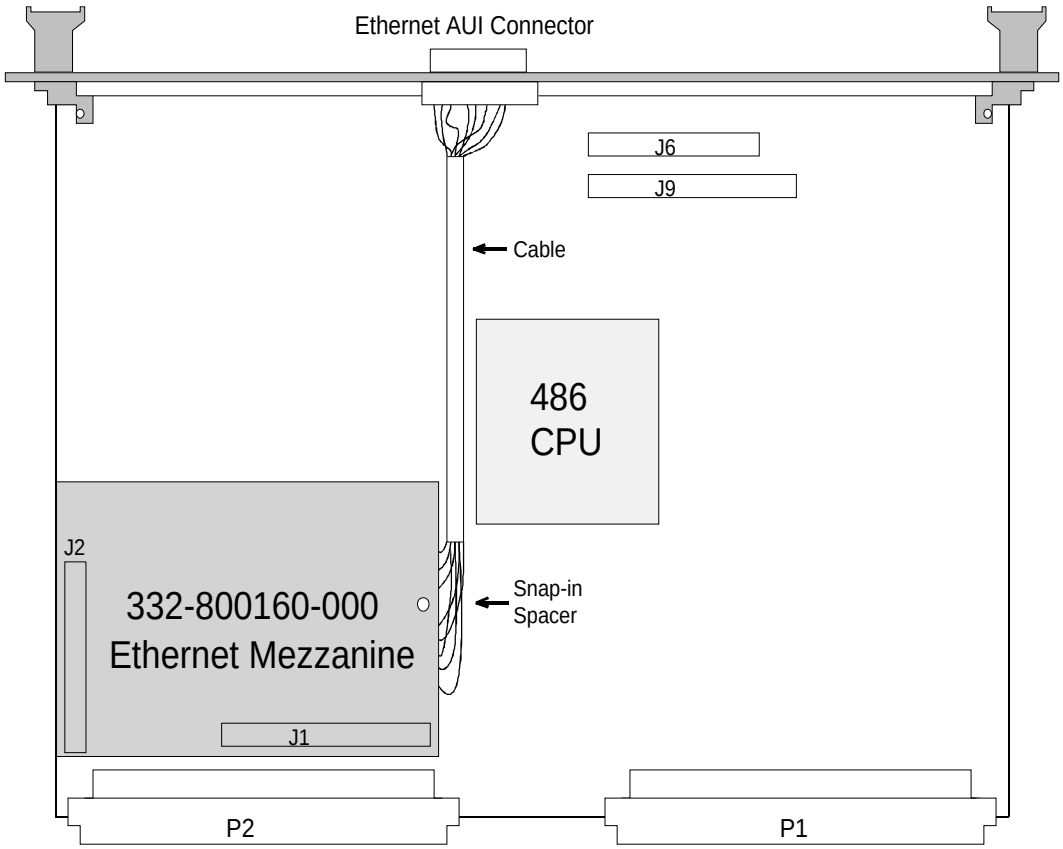


Figure A-1 Location of the Ethernet Mezzanine

Ethernet Mezzanine Diagnostic Software

The Ethernet Mezzanine option is provided with hardware diagnostic software in the EDIAGS directory of the associated Ethernet Mezzanine option diskette. This diskette contains the **NIC Inspector** and **ATLES** programs provided with the permission of National Semiconductor Corporation.

- The **NIC Inspector** is a DOS program with a windows-like user interface. This is the recommended program for verifying correct operation of the Ethernet Mezzanine. This software requires an AUI to physical media adapter and either a real physical network or just a properly terminated short cable to allow the execution of basic self tests. The **NIC Inspector** does not allow the user to change the Ethernet Mezzanine settings, such as I/O address, Boot Prom size and address, IRQ selection, etc. It is recommended that these factory default settings not be changed under normal circumstances. If they must be changed, another provided program (**ATLES**) can do this. Test the Ethernet Inspector with the **NIC Inspector** as follows:

  1. From the EDIAGS directory type `INSPECT` to begin execution.
  2. Use the cursor keys to select an I/O address of 320. (the factory default)
  3. Press the <TAB> key and then use the cursor keys to select an interrupt level of 5. There may be a couple of warning messages informing that level 4 is in use. If so, just ignore them and press the <ENTER> key to continue. The only valid choice for the Ethernet Mezzanine interrupt level is 5.
  4. Press and release the <ALT> key to select the menu. Use the cursor keys to highlight the **Open** command, then press <ENTER> to begin testing the Ethernet Mezzanine.

- ATLES is also a DOS program that provides diagnostic capabilities as well as the ability to make changes in the Ethernet Mezzanine settings and store them into the small nonvolatile EEPROM at U5. Some users may wish to install a Boot EPROM in the 28-pin socket at U1. This previously programmed Boot EPROM will then allow a cold boot directly from the network if the network server is loaded with the proper software. ATLES provides the ability to select size and address parameters and to enable or disable the Boot EPROM. The recommended device for use as a Boot EPROM is a 27C256.

1. From the EDIAGS directory enter ATLES to begin execution.
2. Press the <F1> key twice to configure and initialize the Ethernet Mezzanine.
3. Press <F1> as required to select an I/O port address of 0x320. Press <ENTER>.
4. Highlight Exit (using the down arrow key) and press <ENTER>.
5. Press the <F5> key to begin diagnostic tests.

Technical Details

The Ethernet Mezzanine is an ISA bus peripheral responding to thirty-two bytes in I/O space at $0320 through $033F (the factory default I/O address). If a user installed Boot EPROM is installed in the socket at U1 and enabled, the Ethernet Mezzanine will also occupy up to 32 Kbytes in ISA memory space at an address selected by the user.

The **ATLES** program allows writing to the EEPROM from which the Atlantic IC obtains its operating parameters at power-up (not to be confused with the Boot EPROM). This device provides non-volatile storage. Valid selections for the memory occupied by the Boot EPROM on VMIVME-7486 products are shown below :

NO BOOT ROM => 0 Kbytes (Factory Default)

EPROM Size = 8 Kbytes or 16 Kbytes

Address = C000 => DO NOT SELECT !!! (conflicts with VGA area)

Address = C400 => DO NOT SELECT !!! (conflicts with VGA area)

Address = C800 => 8 Kbytes from C8000 to C9FFF or 16 Kbytes to CDFFF

Address = CC00 => 8 Kbytes from CC000 to CDFFF

Address = D000 => DO NOT SELECT !!! (conflicts with VMEbus Window)

Address = D400 => DO NOT SELECT !!! (conflicts with VMEbus Window)

Address = D800 => DO NOT SELECT !!! (conflicts with VMEbus Window)

Address = DC00 => DO NOT SELECT !!! (conflicts with VMEbus Window)

EPROM Size = 32 Kbytes

Address = C000 => DO NOT SELECT !!! (conflict with VGA area)

Address = C800 => 32 Kbytes from C8000 to CFFFF

Address = D000 => DO NOT SELECT !!! (conflicts with VMEbus Window)

Address = D800 => DO NOT SELECT !!! (conflicts with VMEbus Window)

CAUTION Use extreme caution when attempting to make changes to Boot EPROM options. It is possible to select an address range that conflicts with the system VGA area or the Real Mode VMEbus Window. Conflicting addresses can cause *serious* problems for the following reasons:

- An addressing conflict with the VGA area will prevent a normal system boot.
- However, changing the option back to a valid selection requires a normal system boot.
- Therefore, it is possible to select a *fatal* option.

DO NOT SELECT A BOOT EPROM ADDRESS CONFLICTING WITH THE VGA AREA !!!

APPENDIX

B

Flash Memory Option

Contents

Introduction

The VMIVME-7486 CPU board may be ordered with the 2 Mbyte Flash Memory Mezzanine option. The 2 Mbyte Flash Memory Mezzanine allows the user to configure the VMIVME-7486 CPU as a diskless VMEbus master running software stored in the flash memory. The 2 Mbyte Flash Memory Mezzanine emulates a 1.44 Mbyte floppy diskette using VMIC-proprietary Flash BIOS. The user may transfer the contents of a 1.44 Mbyte floppy diskette (easily formatted with DOS system files) containing the user program to the Flash Memory Mezzanine, along with control and batch information. The VMIVME-7486 CPU is then configured to boot from the Flash Memory Mezzanine and perform all 486 PC / AT functions in a single VME 6U slot.

Preparing the Flash Memory Mezzanine

The VMIVME-7486 with 2 Mbyte Flash Memory Option has an additional strapping option located on the Flash Mezzanine. This horizontal jumper is located on the side of the Flash Mezzanine near the 486 CPU as shown in Figure B-1.

1. Installing the jumper shunt connects +12V to the Vpp input of all flash memory devices except the Flash BIOS..

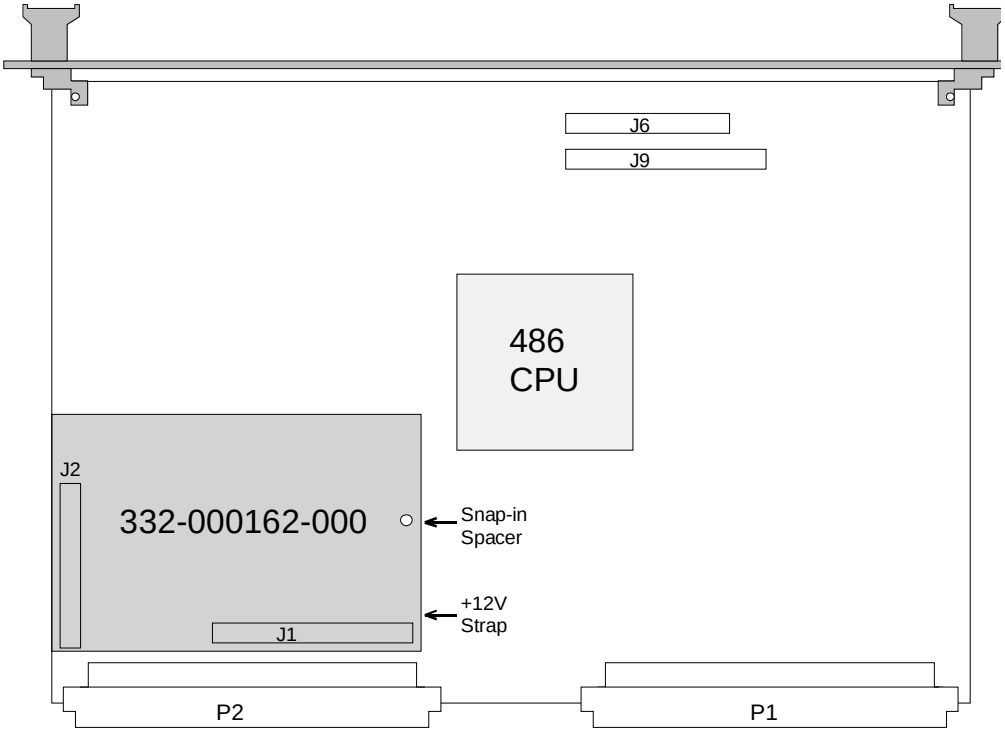


Figure B-1 Location of +12V Strap

2. Once the +12V jumper shunt is installed, attach a 3.5" floppy drive (and also possibly a hard drive) to the VMIVME-7486 in the VME chassis.
3. Apply power to the chassis and prepare to hit the <DEL> key during the AMI BIOS memory test in order to run the BIOS Setup program.
4. When the BIOS Setup program opening screen appears, select **STANDARD SETUP**. Using the cursor keys to move and <PG UP> / <PG DN> keys to toggle settings, enable drive B: as a 1.44 Mbyte floppy drive.
5. Press <ESC> and then select **WRITE SETUP TO CMOS**. Answer <Y>, then press <ESC> after the setup has been saved to CMOS.

6. Observe DOS boot up. The flash BIOS should print out the following message:

```
FLASH BIOS V1.0 (C) VMIC 1994
```

Copying Files to Flash Memory Mezzanine

1. Format a bootable floppy disk (that is, ensure the system files IO.SYS and MSDOS.SYS are installed on the floppy using the format command: **FORMAT A: /S**).
2. Copy CONFIG.SYS, AUTOEXEC.BAT, and other user files onto the newly-formatted 1.44 Mbyte floppy disk.
3. The VMIVME-7486 with 2 Mbyte Flash Memory option is shipped with a 3.5" diskette containing the programming utility **PFLASH.EXE**. Insert the VMIC diskette in the drive and execute the programming utility **PFLASH.EXE**. The **PFLASH** program will prompt for the user-formatted floppy to be inserted again.
4. Upon successful completion of the flash programming software, execute the **DIR B:** command. The flash mezzanine “B:" directory contents and volume number should now match that of the floppy in A:.

Configuring the VMIVME-7486 to Boot from the Flash Memory Mezzanine

1. To configure the VMIVME-7486 as a diskless VMEbus master, turn power off to the VME chassis and remove the floppy and hard drive cables.
2. Apply power to the chassis and prepare to hit the <DEL> key during the AMI BIOS memory test in order to run the BIOS setup program.
3. When the setup program opening screen appears, select **STANDARD SETUP**.
4. Using the cursor keys to move and <PG UP> / <PG DN> keys to toggle settings; disable floppy drive B: and hard drive C:.
5. Press <ESC> to exit this menu, then select **ADVANCED SETUP**.
6. Disable the **SEEK FLOPPY DRIVE AT BOOT** option.
7. Configure **SYSTEM BOOT UP SEQUENCE** for A:, C:.
8. If the VMIVME-7486 is to be configured as a diskless VMEbus CPU without keyboard, disable the **WAIT FOR <F1> IF ANY ERROR** option.
9. Press <ESC> and then select **WRITE SETUP TO CMOS**.
10. Answer <Y>, then hit <ESC> after the setup has been saved to CMOS.
11. Once the +12V jumper shunt is removed, the Flash Memory Mezzanine is write-protected.

Re-Programming the Flash Memory Mezzanine

The Flash Memory Mezzanine may be programmed more than 10,000 times when the PFLASH programming utility is employed exclusively. Simply follow the previous three procedures *Preparing the Flash Memory Mezzanine* on page 106, *Copying Files to Flash Memory Mezzanine* on page 108, and *Configuring the VMIVME-7486 to Boot from the Flash Memory Mezzanine* on page 109. Refer to *Programming* on page 112 for information about programming the Flash Memory Mezzanine without using the **PFLASH** utilily.

Technical Details

The 2 Mbyte Flash Memory Mezzanine is an ISA bus peripheral responding to eight bytes in I/O space at I/O $0300 through I/O $0307, and 32 Kbytes in memory space between $0C8000 through $0CFFFF. The Flash BIOS is configured from the factory to be read-only at memory range $0C8000-$0CFFFF with byte-wide data path. All I/O accesses at addresses $0300, $0302, and $0304 are 16-bit wide even though the field at $0302 contains only four valid data bits. All I/O bus accesses to the 2 Mbyte Flash Memory Mezzanine are zero wait-state.

The 2 Mbyte Flash Memory Mezzanine bulk non-volatile storage is I/O-mapped utilizing an address pointer and a bi-directional data port. The write-only address pointer has the following format:

Lower Address - $0300

16-bit Lower Address Value															
A16															A1

Upper Address Value - $0302

4-bit Upper Address Value				
	A20	A19	A18	A17

Data Port - $0304

16-bit Flash Data Value															
D15															D0

Programming

The user is instructed to refer to the following publications by Intel or AMD for 28F020 programming algorithm information:

1994 Flash Memory: Volume I
Intel Corp.
Intel Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641 USA

1994/1995 Flash Memory Products Data Book
Advanced Micro Devices AMD
901 Thompson Place
P.O Box 3453
Sunnyvale, CA 94088-3453 USA

Maintenance

Maintenance

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

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

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

Maintenance Prints

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