VMIVME-7586
PC/AT COMPATIBLE
VMEbus CONTROLLER

PRODUCT MANUAL

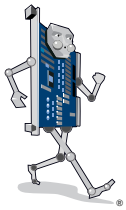
DOCUMENT NO. 500-017586-000 B

Revised July 21, 1997

VME MICROSYSTEMS INTERNATIONAL CORPORATION
12090 SOUTH MEMORIAL PARKWAY
HUNTSVILLE, AL 35803-3308
(205) 880-0444
(800) 322-3616
FAX NO.: (205) 882-0859

COPYRIGHT AND TRADEMARKS

(IOWorks man figure)



(I/O man figure)

UIOC®

WinUIOC®

(VMIC logo)

VME Microsystems International Corporation

VMIVME-7586 Error Notification

Because of a recently discovered design flaw in the VIC64 (Cypress Semiconductor) used to implement the VMEbus interface on the VMIVME-7586, VMIC warns users that a malfunction may occur. The following paragraphs below provide information from Cypress concerning the conditions that must be present to cause this malfunction.

Cypress Erratum:

The following paragraphs came directly from the 6/13/97 Cypress Erratum (Reference No. 95):

Lword and A7-A1 are driven on the VMEbus by a VIC64 slave that is not the addressed slave to the D64 transaction in progress thus corrupting the data of the transaction. The following condition must be met to see this failure:

1. Defining this misbehaving VIC64 as the “failing slave” and the transaction in which the failure is noted the “current transaction.”
2. The failing slave was the addressed slave to the transaction immediately preceding the current transaction. We define this transaction to be the “preceding transaction.”
3. The preceding transaction was a D64 slave read from the failing slave.
4. The current transaction must be a D64 transaction.
5. The preceding transaction exhibits a local “read ahead” cycle which extends (local PAS+ signal is asserted) beyond the address broadcast phase (first VMEbus DS1/0 deassertion after AS* assertion) of the current transaction.

Work around, Hardware:

PAS* must be deasserted before DSB (later deassertion of either DS1/0) of the address broadcast cycle of the current transaction deasserts.

VMIC’s Response:

These conditions may arise if:

1. You have more than one device that may be slave to 64-bit block transfers and at least one has a VIC64 chip.
2. Block transfer master(s) can reassert AS* and execute a BLT 64 Address Broadcast quickly (less than about 100 ns AS* hightime).

The VIC64 as used on the VMIVME-7586 does not cause trouble during Master Mode. In other words, the VMIVME-7586 can safely do master BLT 64 transactions under all conditions. The described malfunction only occurs when the VIC64 is a BLT 64 **slave** (read **from/by** master) *and* the slave transaction is rapidly followed by **another** BLT 64 slave transaction **to** another slave.

RECORD OF REVISIONS

REVISION LETTER	DATE	PAGES INVOLVED	CHANGE NUMBER
A	10/25/96	Release	96-0744
B	07/21/97	Cover and Page iii	97-0539

VMIC
SAFETY SUMMARY

THE FOLLOWING GENERAL SAFETY PRECAUTIONS MUST BE OBSERVED DURING ALL PHASES OF THE OPERATION, SERVICE, AND REPAIR OF THIS PRODUCT. FAILURE TO COMPLY WITH THESE PRECAUTIONS OR WITH SPECIFIC WARNINGS ELSEWHERE IN THIS MANUAL VIOLATES SAFETY STANDARDS OF DESIGN, MANUFACTURE, AND INTENDED USE OF THIS PRODUCT. VME MICROSYSTEMS INTERNATIONAL CORPORATION ASSUMES NO LIABILITY FOR THE CUSTOMER'S FAILURE TO COMPLY WITH THESE REQUIREMENTS.

GROUND THE SYSTEM

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

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

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

KEEP AWAY FROM LIVE CIRCUITS

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

DO NOT SERVICE OR ADJUST ALONE

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

DO NOT SUBSTITUTE PARTS OR MODIFY SYSTEM

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

DANGEROUS PROCEDURE WARNINGS

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

WARNING

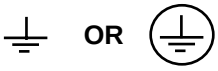
DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS SYSTEM. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

SAFETY SYMBOLS

GENERAL DEFINITIONS OF SAFETY SYMBOLS USED IN THIS MANUAL

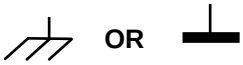
Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the system.

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



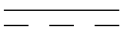
Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.

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.

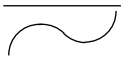


Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.

Alternating current (power line).



Direct current (power line).



Alternating or direct current (power line).

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.

NOTE:

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

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION 1-1

SECTION 1 - INTRODUCTION TO THE VMIVME-7586 1-1

SECTION 2 - ABOUT THIS MANUAL 1-2

SECTION 3 - PRODUCT FAMILY 1-3

SECTION 4 - REFERENCES 1-5

SECTION 5 - PC/AT FEATURES 1-6

SECTION 6 - VMEbus FEATURES 1-8

CHAPTER 2 - INSTALLATION AND SETUP 2-1

SECTION 1 - INTRODUCTION 2-1

SECTION 2 - UNPACKING PROCEDURES 2-2

SECTION 3 - HARDWARE SETUP 2-2

SECTION 4 - LED STATUS DEFINITION 2-5

SECTION 5 - INSTALLATION 2-5

SECTION 6 - FRONT PANEL CONNECTORS 2-6

SECTION 7 - PC/104 EXPANSION SITE 2-7

SECTION 8 - BIOS SETUP 2-8

SECTION 9 - CONFIGURING OPERATING SYSTEMS 2-8

GENERAL RULE REGARDING OPERATING SYSTEMS 2-8

CONFIGURATION EXAMPLES 2-9
Configuring MS-DOS for the VMIVME-7586 2-9
Configuring Windows for the VMIVME-7586 2-10

CHAPTER 3 - PC/AT FUNCTIONS 3-1

SECTION 1 - CPU FUNCTIONAL OVERVIEW 3-1

SECTION 2 - PHYSICAL MEMORY 3-2

SECTION 3 - MEMORY AND I/O PORT MAPS 3-3

MEMORY MAP 3-3

I/O PORT MAP 3-4

SECTION 4 - PC/AT INTERRUPTS 3-8

SECTION 5 - ENHANCED I/O PERIPHERAL PORTS 3-12

SERIAL PORTS 3-13

PARALLEL PORT 3-13

SECTION 6 - VIDEO GRAPHICS ADAPTER 3-14

SECTION 7 - PC/104 EXPANSION SITE 3-15

CHAPTER 4 - VMEbus FUNCTIONS 4-1

SECTION 1 - INTRODUCTION 4-1

SECTION 2 - VMEbus INTERFACE 4-2

VMEBUS INTERFACE OVERVIEW 4-2

VMEBUS INTERFACE HARDWARE 4-4

PROGRAMMING THE VMEBUS INTERFACE 4-6

SECTION 3 - VMEbus MASTER OPERATION 4-7

REAL MODE ACCESS 4-8
PROTECTED MODE ACCESS 4-10
128 MBYTE - PROTECTED MODE ACCESS 4-13

SECTION 4 - VMEbus SLAVE OPERATION 4-16

INTERPROCESSOR COMMUNICATIONS 4-17
DUAL-PORTED DRAM 4-21
SLAVE ACCESS MEMORY MAP 4-24
ADDRESS REMAPPING 4-26
SYSTEM CONSIDERATIONS 4-28

SECTION 5 - SYSTEM CONTROLLER FUNCTIONS 4-31

SECTION 6 - VMEbus INTERRUPT HANDLING 4-32

SOFTWARE INTERRUPTS 4-36
INTERRUPT ON BERR* 4-36
PERIODIC TIMER INTERRUPT 4-36
INTERRUPT PROCESSING 4-37

SECTION 7 - VMEbus INTERRUPTER 4-39

SECTION 8 - VMEbus REQUESTER 4-40

SECTION 9 - READ-MODIFY-WRITE CYCLES 4-40

SECTION 10 - BLOCK TRANSFERS 4-42

DRAM REFRESH CONSIDERATIONS 4-42
MASTER BLT OPERATION 4-43
SLAVE BLT OPERATION 4-45

SECTION 11 - VME64 FUNCTIONS 4-45

MASTER VME64 OPERATION 4-45

SLAVE VME64 OPERATION 4-45

SECTION 12 - BYTE ORDERING 4-46

BYTE SWAPPING 4-46
The Byte-Swapping Problem Defined 4-47
Byte Swapping and the VMEbus 4-48
VMIVME-7586 Byte-Swapping Hardware 4-49
Master/Slave Byte Swapping 4-51

SECTION 13 - VMIVME-7586 REGISTERS 4-52

REGISTER MAPS 4-52

SYSTEM REGISTER DETAILS 4-58
General Purpose Command Register 4-58
Product ID Register 4-61
VIC Base Register 4-61
Extended/Standard Address Register 4-62
On-Board Video Status Register 4-63
128 Mbyte Mode Enable Register 4-63
Rearm Interrupt Register 4-63
Slave Address Mask/Compare Registers 4-63
Size Register 4-63
Remap Register 4-64

INTERRUPT ACKNOWLEDGE REGISTER DETAILS 4-64

VIC REGISTER DETAILS 4-65
VMEbus Interrupter Interrupt Control Register 4-65
VMEbus Interrupt Control Registers 4-66
DMA Status Interrupt Control Register 4-67
Local Interrupt Control Registers 4-68
ICGS Interrupt Control Register 4-70
ICMS Interrupt Control Register 4-70
Error Group Interrupt Control Register 4-71
ICGS Interrupt Vector Base Register 4-73
ICMS Interrupt Vector Base Register 4-73
Local Interrupt Vector Base Register 4-74

Error Group Interrupt Vector Base Register 4-75
Interprocessor Communications Registers 4-76
VMEbus Interrupt Request/Status Register 4-76
VMEbus Interrupt Vector Base Registers 4-76
Transfer Timeout Register 4-77
Local Bus Timing Register 4-80
Block Transfer Definition Register 4-81
Interface Configuration Register 4-82
Arbiter/Requester Configuration Register 4-84
Address Modifier Source Register 4-85
Bus Error Status Register 4-86
DMA Status Register 4-87
Slave Select 0 Control Register 0 4-89
Slave Select 0 Control Register 1 4-91
Slave Select 1 Control Register 0 4-94
Slave Select 1 Control Register 1 4-96
Release Control Register 4-99
Block Transfer Control Register 4-100
Block Transfer Length Registers 4-102
System Reset Register 4-103

INTERPROCESSOR COMMUNICATIONS REGISTERS 4-103
Interprocessor Communications Switch Register 4-104
Interprocessor Communication Registers 4-105
VIC Version Register 4-105
Reset/Halt Status Register 4-105
Mailbox Semaphore Register 4-106
Set/Clear ICGS Switch Registers (Slave-Only) 4-108
Set/Clear ICMS Switch Registers (Slave-Only) 4-108

CHAPTER 5 - MAINTENANCE 5-1

SECTION 1 - MAINTENANCE 5-1

SECTION 2 - MAINTENANCE PRINTS 5-1

APPENDIX A - CONNECTOR PINOUTS A-1

SECTION 1 - INTRODUCTION A-1

SECTION 2 - ETHERNET CONNECTOR PINOUT A-3

SECTION 3 - FLOPPY DRIVE CONNECTOR PINOUT A-4

SECTION 4 - IDE HARD DRIVE CONNECTOR PINOUT A-5

SECTION 5 - KEYBOARD CONNECTOR PINOUT A-6

SECTION 6 - PC/104 CONNECTOR PINOUT A-7

SECTION 7 - PRINTER CONNECTOR PINOUT A-9

SECTION 8 - SERIAL CONNECTOR PINOUT A-10

SECTION 9 - VIDEO CONNECTOR PINOUT A-11

SECTION 10 - VMEbus CONNECTOR PINOUT A-12

APPENDIX B - ETHERNET OPTION B-1

SECTION 1 - INTRODUCTION B-1

SECTION 2 - ETHERNET SOFTWARE COMPATIBILITY B-2

SECTION 3 - ETHERNET DRIVER SOFTWARE B-3

SECTION 4 - ETHERNET DIAGNOSTIC SOFTWARE B-3

SECTION 5 - TECHNICAL DETAILS B-5

APPENDIX C - FLASH MEMORY OPTION C-1

SECTION 1 - INTRODUCTION C-1

SECTION 2 - PREPARING THE FLASH MEMORY C-2

SECTION 3 - COPYING FILES TO FLASH MEMORY C-3

SECTION 4 - USING FLASH MEMORY AS BOOT DEVICE C-4

SECTION 5 - REPROGRAMMING FLASH MEMORY C-4

SECTION 6 - TECHNICAL DETAILS C-4

SECTION 7 - PROGRAMMING C-6

APPENDIX D - BASIC INPUT / OUTPUT SYSTEM D-1

SECTION 1 - INTRODUCTION D-1

SECTION 2 - STANDARD FEATURES D-1

SECTION 3 - QUICK SETUP D-2

SECTION 4 - PROGRAM DESCRIPTION D-5

USER INTERFACE D-6

CONTROL KEY SUMMARY D-7

SECTION 5 - PROGRAM MENUS AND MENU ITEMS D-8

MAIN MENU D-8
System Time D-9
System Date D-9
Diskette A:/B: D-9
Video System D-9
Large Disk Mode D-9
System Memory/ Extended Memory D-9
IDE Adapter 0 Master/IDE Adapter 0 Slave Sub-menus D-10
Autotype Fixed Disk D-10
Type D-11
Cylinders D-13
Heads D-13
Sectors/Track D-13
Write Precomp D-13

MEMORY CACHE SUB-MENU D-14
Internal Cache D-15
External Cache D-15
Cache Shadow Region D-15

Noncacheable Region D-15

MEMORY SHADOW SUB-MENU D-16
System Shadow D-17
Video Shadow D-17
Shadow Memory Regions D-17
C800 - CFFF D-18
D000 - DFFF D-18
E000 - EFFF D-18

BOOT OPTIONS SUB-MENU D-19
Keyboard D-19
Boot Sequence D-20
SETUP Prompt D-20
POST Errors D-20
Floppy Check D-20
Summary Screen D-21

KEYBOARD FEATURES D-21
NumLock D-22
Key click D-22
Keyboard Autorepeat Rate D-23
Keyboard Autorepeat Delay D-23

SECTION 6 - EXITING THE PhoenixBIOS D-23

EXIT MENU D-23
Save Changes & Exit D-24
Discard Changes & Exit D-24
Load Default Values D-24
Load Previous Values D-24
Save Changes D-25

SECTION 7 - STATUS AND ERROR MESSAGES D-26

LIST OF FIGURES

Figure 1-1 VMIVME-7586 Board View 1-3

Figure 1-2 VMIVME-7586 Partial Block Diagram 1-7

Figure 1-3 VMIVME-7586 VMEbus Functions 1-9

Figure 2-1 I/O Port and Jumper Locations 2-3

Figure 2-2 LED Position on the Front Panel 2-5

Figure 2-3 PC/104 Mechanical Connection 2-7

Figure 3-1 Connections Between the PC Interrupt Logic Controller and the VMEbus 3-12

Figure 4-1 VMIVME-7586 VMEbus Functions 4-3

Figure 4-2 VMEbus Interface Block Diagram 4-4

Figure 4-3 VLIC Block Diagram 4-5

Figure 4-4 Real Mode VMEbus Access 4-9

Figure 4-5 Protected Mode VMEbus Access 4-12

Figure 4-6 128 Mbyte Protected Mode VMEbus Access 4-15

Figure 4-7 VMEbus Slave Interface 4-18

Figure 4-8 Slave Compare Operation 4-20

Figure 4-9 Remap of VMEbus Space to 5x86 Space 4-26

Figure 4-10 Slave Remap Circuit Overview 4-27

Figure 4-11 Slave Addressing Detail 4-30

Figure 4-12 Flowchart for Non-NMI Interrupt Processing for IRQ11 .. 4-38

Figure 4-13 Flowchart for NMI Interrupt Processing 4-39

Figure 4-14 Byte Relationships Using the Little-Endian 5x86 4-47

Figure 4-15 Byte Relationships Using the Big-Endian 68040 4-48
Figure 4-16 5x86-to-VMEbus Data Byte Lanes 4-50
Figure A-1 VMIVME-7586 Connector Locations A-2
Figure A-2 Ethernet Connector Pinout A-3
Figure A-3 Floppy Drive Connector Pinout A-4
Figure A-4 IDE Hard Drive Connector Pinout A-5
Figure A-5 PS/2 Keyboard Connector Pinout A-6
Figure A-6 PC/AT Keyboard Connector Pinout A-6
Figure A-7 PC/104 Connector Diagram A-7
Figure A-8 Printer Connector Pinout A-9
Figure A-9 Serial Connector Pinouts A-10
Figure A-10 Video Connector Pinout A-11
Figure A-11 VMEbus Connector Diagram A-12
Figure B-1 Location of the Ethernet Mezzanine B-2
Figure C-1 Flash Mezzanine Jumper Location C-2
Figure D-1 PhoenixBIOS Opening Display D-3
Figure D-2 Elements of the PhoenixBIOS User Interface D-6
Figure D-3 PhoenixBIOS Main Menu D-8
Figure D-4 PhoenixBIOS IDE Adapter Master Sub-menu D-10
Figure D-5 PhoenixBIOS Memory Cache Sub-menu D-14
Figure D-6 PhoenixBIOS Memory Shadow Sub-menu D-17
Figure D-7 PhoenixBIOS Boot Options Sub-menu D-19
Figure D-8 PhoenixBIOS Example Summary Screen D-21
Figure D-9 PhoenixBIOS Keyboard Features Sub-menu D-22
Figure D-10 PhoenixBIOS Exit Menu D-23

LIST OF TABLES

Table 1-1	PC/AT I/O Features	1-8
Table 2-1	VMIVME-7586 Jumper Functions and Settings	2-4
Table 2-2	VMEbus Window Addresses	2-9
Table 3-1	VMIVME-7586 Memory Map	3-3
Table 3-2	VMIVME-7586 I/O Address Map	3-5
Table 3-3	PC/AT Hardware Interrupts	3-8
Table 3-4	PC/AT Interrupt Vector Table	3-9
Table 3-5	Parallel Port Modes	3-13
Table 3-6	Common Supported Graphics Video Resolutions	3-14
Table 4-1	Protected Mode VMEbus Address Modifiers	4-11
Table 4-2	128 Mbyte Master Window Definitions	4-14
Table 4-3	Slave Access Memory Map	4-25
Table 4-4	Interrupt Priorities	4-32
Table 4-5	Interrupt Level Assignments	4-34
Table 4-6	VMEbus Byte Assignment to the Data Lines	4-46
Table 4-7	Byte Swap Modes	4-50
Table 4-8	System Register Map	4-53
Table 4-9	VIC Register Map	4-54
Table 4-10	Slave Access Register Map	4-57
Table A-1	PC/104 Connector Pinout	A-7
Table A-2	VMEbus Connector Pinout	A-12
Table B-1	Boot EPROM Address Selection	B-6
Table D-1	PhoenixBIOS Setup Keystroke Actions	D-7
Table D-2	PhoenixBIOS Fixed Disk Table	D-12
Table D-3	PhoenixBIOS Status and Error Messages	D-26

CHAPTER

1

INTRODUCTION

IN THIS CHAPTER:

SECTION 1 - INTRODUCTION TO THE VMIVME-7586 1-1
SECTION 2 - ABOUT THIS MANUAL 1-2
SECTION 3 - PRODUCT FAMILY 1-3
SECTION 4 - REFERENCES 1-5
SECTION 5 - PC/AT FEATURES 1-6
SECTION 6 - VMEbus FEATURES 1-8

SECTION 1 - INTRODUCTION TO THE VMIVME-7586

The VMIVME-7586 single-board microcomputer is a complete IBM PC/AT 80486 compatible microcomputer with the additional benefits of Eurocard construction and full compatibility with the VMEbus Specification Rev. C.1. The VMIVME-7586 also has an industry-standard PC/104 expansion site.

VMIC's VMIVME-7586 has two operating modes: a standard PC/AT compatible mode, and a VMEbus controller mode. Upon powerup it functions as a standard PC/AT. It executes a PC/AT-type power-on self-test, then boots up MS-DOS, Windows or OS/2 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-7586 is discussed in Chapter 3 of this manual.

After booting, the VMIVME-7586 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-7586's VMEbus interface registers according to Chapter 4 of this manual.

The VMIVME-7586 programmer may quickly and easily control all the VMIVME-7586 VMEbus functions simply by linking to a library of VMEbus interrupt and control functions. This library is located in VMIC's VMIVME-9420 IOWorks Access software for Windows NT users and VMIVME-7420 *VMEaccess* software for MS-DOS users.

SECTION 2 - 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 "doubleword."
- Motorola programmers should note that Intel processors have an I/O bus that is completely independent from the memory bus. Every effort has been made in the manual to clarify this by referring to registers and logical entities in I/O space by prefixing I/O addresses as such. Thus, a register at "I/O $140" is not the same as a register at "$140," since the latter is on the memory bus while the former is on the I/O bus.
- Intel programmers should note that addresses are listed in this manual using a linear, "flat-memory" model rather than the old segment:offset model associated with Intel Real Mode programming. Thus, a ROM chip at a segment:offset address of C000:0 will be listed in this manual as being at address $C0000. For reference, here are some quick conversion formulas:

Segment:Offset to Linear Address

Linear Address = (Segment × 16) + Offset

Linear Address to Segment:Offset

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

Offset = *remainder* × 65536

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

Note that there are many possible segment:offset addresses for a single location. The formula above will provide a unique segment:offset address by forcing the segment to an even 64 Kbyte boundary, for example, $C000,

$E000, and so forth. When using this formula, make sure to round the offset calculation properly!

SECTION 3 - PRODUCT FAMILY

Figure 1-1 shows a simplified view of the VMIVME-7586 board. The VMIVME-7586 is one member of VMIC's line of PC/AT compatible VMEbus controllers, all of which combine a standard PC/AT architecture with the ability to control VMEbus slave boards.

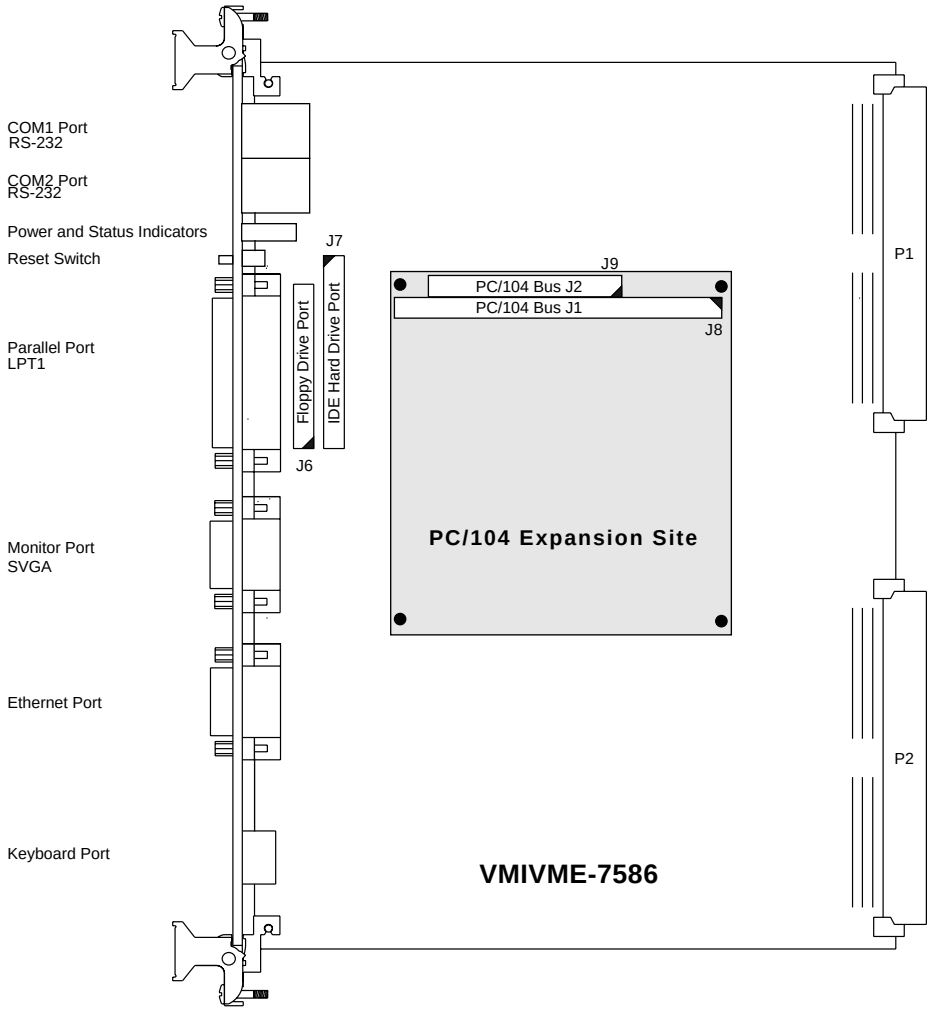


Figure 1-1 VMIVME-7586 Board View

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

The VMIVME-7489 is similar to the VMIVME-7486 PC/AT compatible VMEbus controller, but adds external cache memory, high-speed 16550-compatible serial ports, an enhanced bidirectional parallel port, up to 32 Mbyte 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.

The VMIVME-7586 has combined the best features from the VMIVME-7487 and the VMIVME-7489. It has either 256 Kbyte or 1 Mbyte L2 cache, PC/104 expansion slot, and enhanced I/O as well as multiprocessing capability, dual-ported memory, and a master/slave VMEbus interface. It also supports L1 write-back cache and in slave mode the board provides VMEbus remap to alternate local address capability.

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-7452 is a single-slot module which holds one 3.5 inch floppy drive and up to two 2.5 inch hard drives. The VMIVME-7432 is a PC/104 module with a VMEbus 6U form factor that allows a half-length PC ISA bus expansion board to be used in a VMEbus chassis along with the VMIVME-7489 and -7586. The VMIVME-7432 is also compatible with the VMIVME-7450 and -7452.

VMIC's VMIVME-7420 *VMEaccess* software includes a linkable library of functions to simplify the integration and development of high-performance VMEbus systems using any VMIC PC/AT compatible VMEbus controller utilizing the MS-DOS operating system. The library includes functions for such low-level chores as setting up VMEbus data transfers, processing interrupts and conditions, and handling errors.

For VMEbus controllers using the Windows NT operating system, VMIC recommends the use of the VMIVME-9420 IOWorks Access software package. This package is designed to utilize the full capabilities of Windows NT OS on a VMIC PC/AT compatible VMEbus controller.

SECTION 4 - REFERENCES

For the most up-to-date physical description and specifications for the VMIVME-7586, please refer to VMIC specification number 800-017586-000.

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-7586 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 - VMEbus International Trade Association
7825 East Gelding Drive, No. 104
Scottsdale, AZ 85260

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

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

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

VIC068A VAC068A User's Guide
Cypress Semiconductor
3901 North First St.
San Jose, CA 95134
(408) 943-2600

VIC064 Design Notes
Cypress Semiconductor
3901 North First St.
San Jose, CA 95134
(408) 943-2600

SECTION 5 - PC/AT FEATURES

The VMIVME-7586 performs all the functions of a standard IBM PC/AT motherboard with the following features:

- Single-Slot 6U Size
- High-performance 5x86 processor
  Standard 16 Kbyte internal cache
- The 5x86 processor supports level 1 (L1) write-back cache for near 90 MHz Pentium processor performance
- Up to 32 Mbyte of DRAM
- Optional 256 Kbyte or 1 Mbyte high-speed external level 2 write-back cache
- Super VGA video
  1 Mbyte of video DRAM
  Scan resolutions up to 1280 x 1024, noninterlaced
  Color resolutions up to 16.5 million (TruColor)
- Battery-backed clock/calendar
- Front panel reset switch
- On-board ports for keyboard, IDE hard drive, floppy drive, high-speed serial, and enhanced parallel I/O
- PC/104 expansion site for industry-standard modules
- Front panel "vital sign" indicators
  (power, programmable status, and hard drive activity)

An important feature of the PC104 expansion site is the ability of the user to to disable the onboard VGA Controller and to install an alternative controller, permitting the choice of VGA adapters or LCD panels.

The VMIVME-7586 also supports standard PC/AT I/O features such as those listed in Table 1-1 on page 1-8. Figure 1-2 on page 1-7 also shows a partial block diagram of the VMIVME-7586, emphasizing the I/O features.

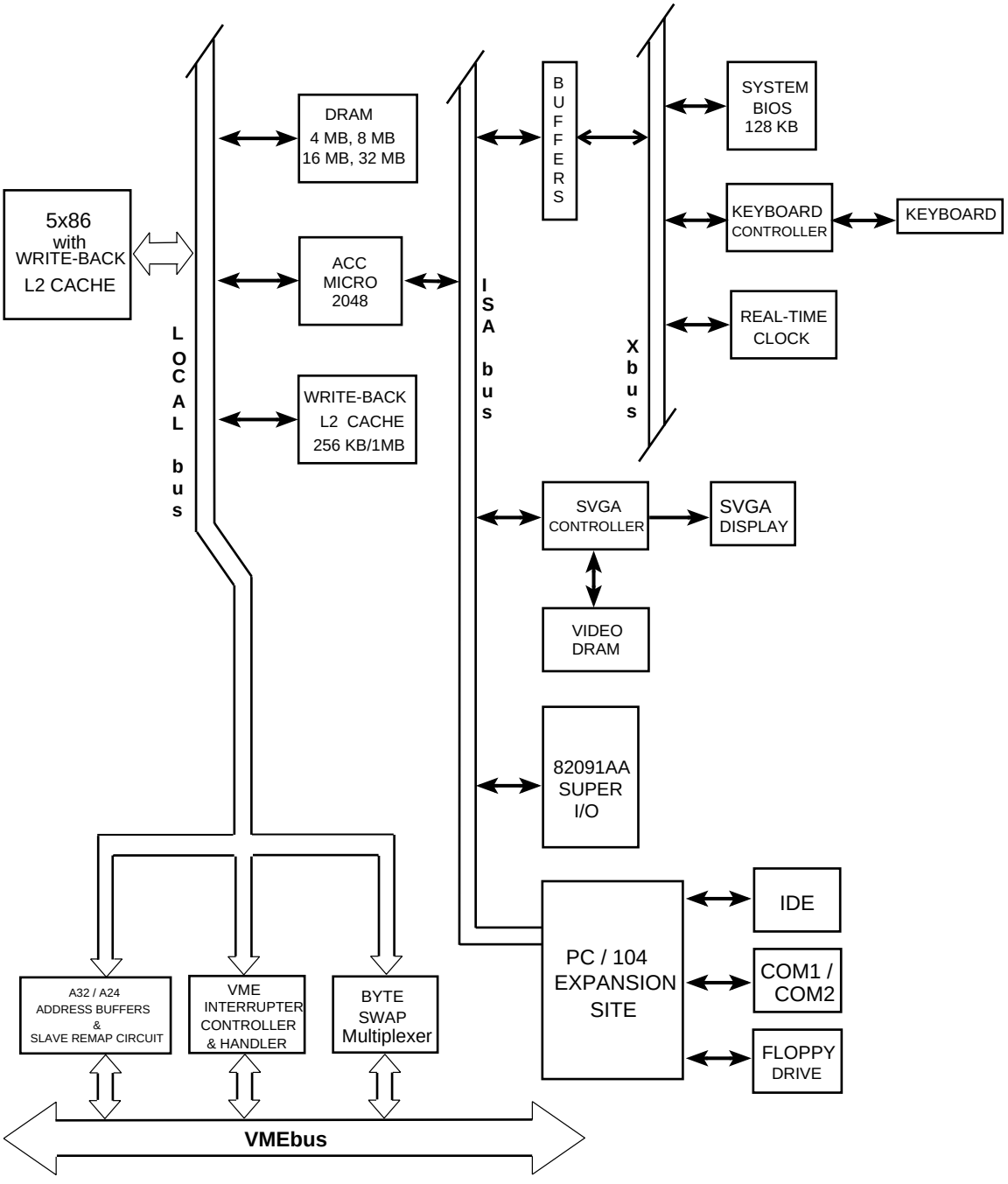


Figure 1-2 VMIVME-7586 Partial Block Diagram

Table 1-1 PC/AT I/O Features

I/O FEATURE	MS-DOS IDENTIFIER	PHYSICAL ACCESS
Two High-Speed Serial Ports (16550-compatible RS-232C)	COM1, COM2	Front Panel RJ45 (male) X 2
One Enhanced Bidirectional Parallel Port (IEEE-1284 ECP/EPP compliant)	LPT1 (or LPT2)	Front Panel DB25S (female)
AT-Style Keyboard Controller with PS/2-Style Adapter	KBD	Front Panel PS/2-Style Mini-DIN Circular (female)
Real-Time Clock/Calendar with Battery	Date, Time	
Super VGA Video Controller with 1 Mbyte DRAM	Display	Front Panel DB15HD High Density (female)
Floppy Disk Controller (two drives maximum)	Drives A, B	34-pin Header Flat Cable type
IDE Fixed Disk Controller (two drives maximum)	Drives C, D	40-pin Header Flat Cable type
Hardware Reset	Cold Boot	Front Panel Pushbutton
IBM/PC Sound	Beep	On-Board Small Speaker
LED Indicators	Power, Status, and Hard Drive Activity	Front Panel

SECTION 6 - VMEbus FEATURES

In addition to its PC/AT functions, the VMIVME-7586 has the following VMEbus features:

- Single-slot, 6U height VMEbus board
- Complete six line Address Modifier (AM-Code) programmability
- 32-bit data interface with separate hardware byte/word swapping for master and slave accesses
- Supports VME64 multiplexed MBLT 64-bit VMEbus block transfers
- User-configured interrupter
- User-configured interrupt handler
- System Controller mode as programmable VMEbus arbiter (PRI, SGL, and RRS modes are supported)
- Full-featured programmable VMEbus requester (ROR, RWD, ROC, and BCAP modes are supported)

- VMEbus BERR* 10 µs bus error timer (jumper enabled)
- Complete VMEbus master access through
  Real-mode memory window or
  Protected-mode linear addressing
- Slave access from the VMEbus to local RAM and interprocessor communications registers
- Slave address remap and window sizing capability
- Selectable 128 Mbyte Master VMEbus window

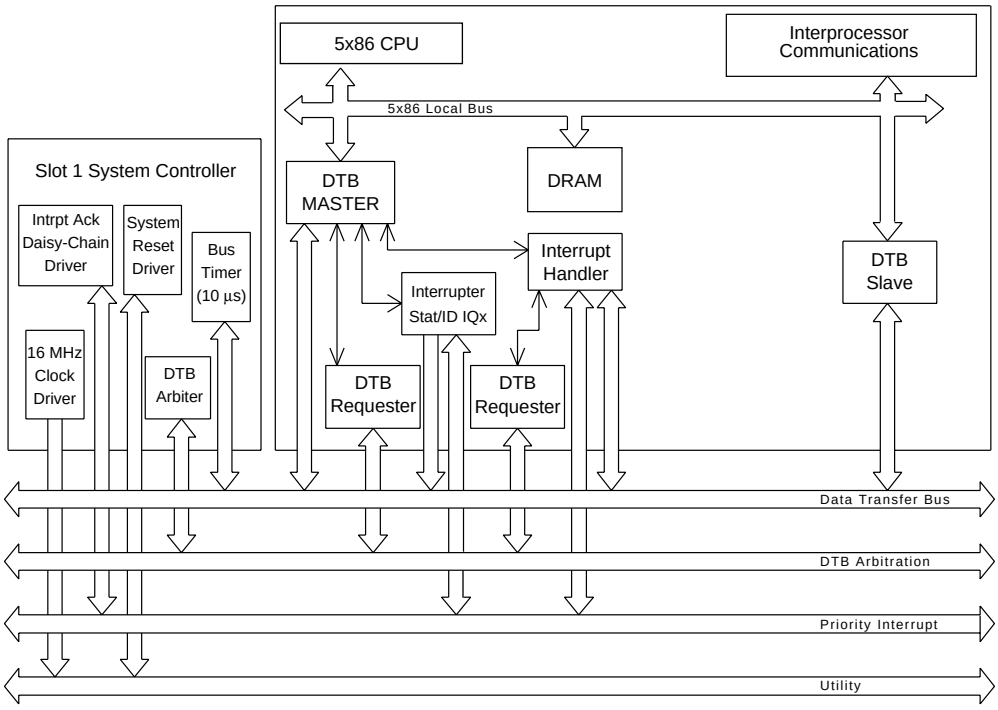


Figure 1-3 VMIVME-7586 VMEbus Functions

Figure 1-3 shows the VMIVME-7586 functions in a typical VMEbus system. The VMIVME-7586 is a versatile single-board solution for VMEbus control with familiar PC/AT operation.

CHAPTER

2

INSTALLATION AND SETUP

IN THIS CHAPTER:

SECTION 1 - INTRODUCTION	**2-1**
SECTION 2 - UNPACKING PROCEDURES	**2-2**
SECTION 3 - HARDWARE SETUP	**2-2**
SECTION 4 - LED STATUS DEFINITION	**2-5**
SECTION 5 - INSTALLATION	**2-5**
SECTION 6 - FRONT PANEL CONNECTORS	**2-6**
SECTION 7 - PC/104 EXPANSION SITE	**2-7**
SECTION 8 - BIOS SETUP	**2-8**
SECTION 9 - CONFIGURING OPERATING SYSTEMS	**2-8**

SECTION 1 - INTRODUCTION

This chapter describes unpacking, inspection, hardware jumper settings, connector definitions, installation, system setup, and operation of the VMIVME-7586.

SECTION 2 - 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).

SECTION 3 - HARDWARE SETUP

The VMIVME-7586 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 2-4 lists each jumper designator, its function, and the factory-installed default configuration

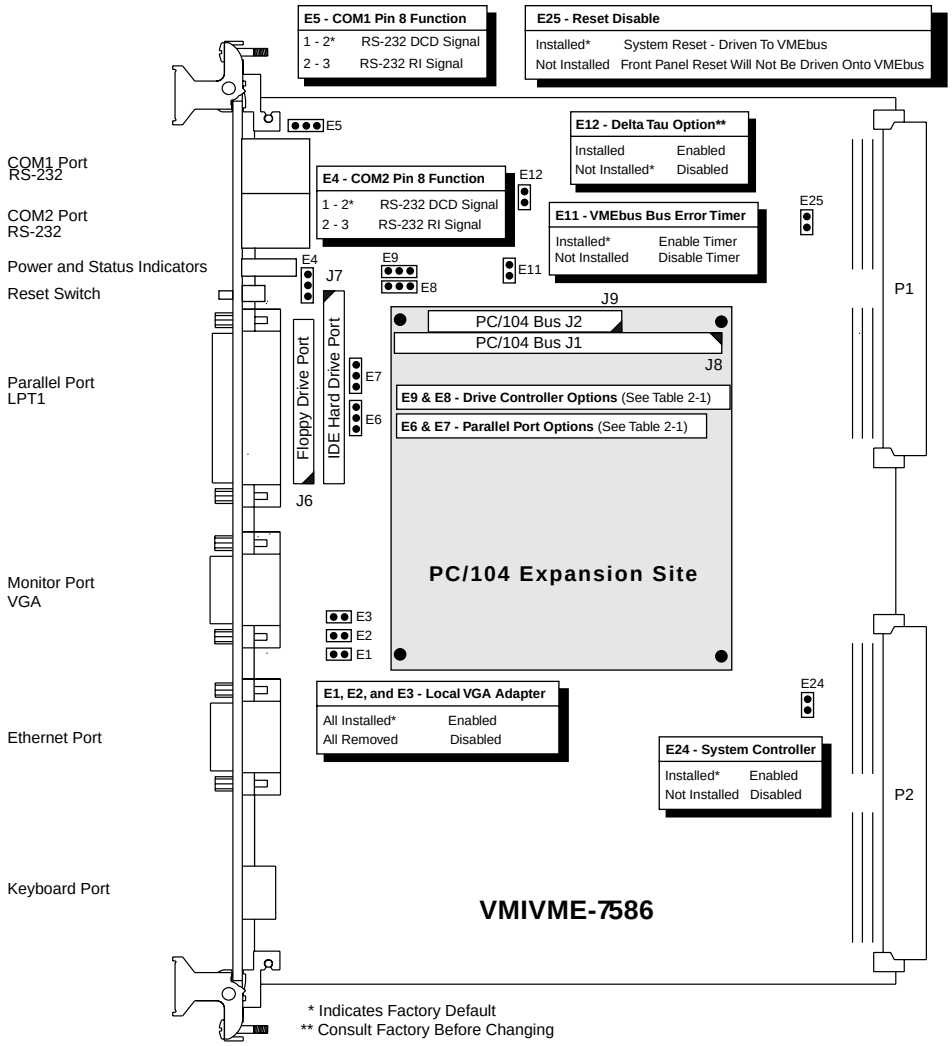


Figure 2-1 I/O Port and Jumper Locations

Table 2-1 VMIVME-7586 Jumper Functions and Settings

JUMPER	FUNCTION	SETTINGS			
E8 & E9	Select Drive Controller Options	**Jumper E8**	**Jumper E9**	**Floppy Disk Controller**	**IDE Controller**
		2-3	2-3	Disabled	Disabled
		2-3	1-2	I/O Address $3F0-$3F7	Disabled
		1-2	2-3	I/O Address $370-$377	I/O Address $170-$177
		1-2 (Default)	1-2 (Default)	I/O Address $3F0-$3F7	I/O Address $1F0-$1F7
E6 & E7	Select Parallel Port Options	**Jumper E7**	**Jumper E6**	**Parallel Port Address**	**Parallel Port Interrupt**
		2-3	2-3	Disabled	Disabled
		2-3 (Default)	1-2 (Default)	I/O Address $378-$37F (Standard LPT1)	IRQ7
		1-2	2-3	I/O Address $278-$27F (Standard LPT2)	IRQ5
		1-2	1-2	I/O Address $3BC-$3BF (Alternate LPT1*)	IRQ7
E1, E2 & E3	Enable/Disable Local VGA Adapter			All Installed = Enabled (Default) All Removed = Disabled	
E24	Enable/Disable System Controller			Installed = Enabled (Default) Removed = Disabled	
E11	Enable/Disable VMEbus Bus Error 10 µs Timeout Timer			Installed = Enabled (Default) Removed = Disabled	
E12**	Enable/Disable Delta Tau Option			Installed = Enables Removed = Disabled (Default)	
E5	1-2 COM1 pin 8* assigned to RS-232 DCD signal 2-3 COM1 pin 8* assigned to RS-232 RI signal			1-2 (DCD)	
E4	1-2 COM1 pin 8* assigned to RS-232 DCD signal 2-3 COM1 pin 8* assigned to RS-232 RI signal			1-2 (DCD)	
E25	Enabled - System Reset Disable - Front Panel Reset			Installed = Enabled (Default) Removed = Disabled	

NOTE:

ANY OTHER JUMPER LOCATIONS ARE RESERVED FOR VMIC USE ONLY AND SHOULD NOT BE ALTERED FROM THE FACTORY DEFAULT SETTINGS.

SECTION 4 - LED STATUS DEFINITION

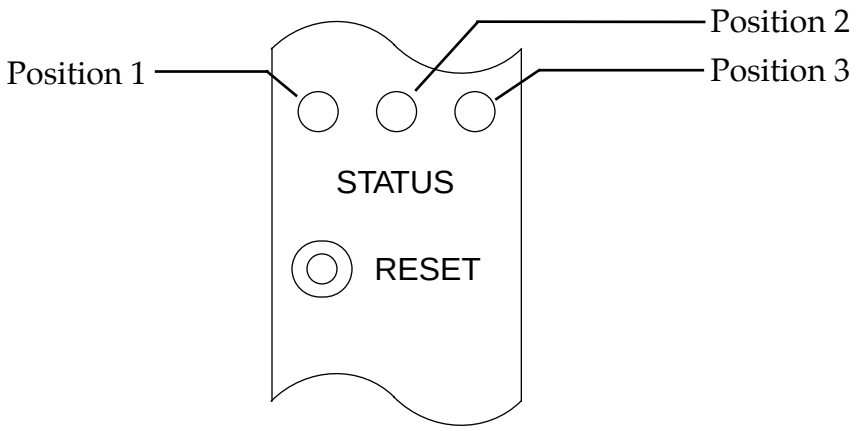


Figure 2-2 LED Position on the Front Panel

Position 1 - Hard Drive Indicator - Indicates when hard drive activity is occurring.

Position 2 - General Purpose/Status - Configured to be used at programmer's discretion. See Table 4-7 on page 4-47 for details.

Position 3 - Power - Indicates when power is applied to the board.

SECTION 5 - INSTALLATION

```
**************
*  CAUTION   *
**************
```

DO NOT INSTALL OR REMOVE BOARD WHILE POWER IS APPLIED.

The VMIVME-7586 conforms to the VMEbus physical specification for a 6U x 4HP dual Eurocard (dual height, single-slot width). It can be plugged directly into any standard chassis accepting this type of board.

Follow these steps to get the VMIVME-7586 up and running:

1. Make sure power to the equipment is off.
2. If a drive module or other expansion module such as the VMIVME-7450 is to be used, connect it to the controller first.

3. Choose a chassis slot. The VMIVME-7586 may be attached to a dual P1/P2 VMEbus backplane or a single P1 backplane. A single P1 connection supplies enough power and allows 16-bit data transfers and 24-bit addressing, but for 32-bit data transfers and 32-bit addressing a dual P1/P2 backplane is required.

   If the VMIVME-7586 is to be the VMEbus system controller, choose the first VMEbus slot and make sure jumper E24 is installed. If some other board is the VMEbus system controller, choose any slot *except* slot one and make sure jumper E24 is removed.

4. Insert the VMIVME-7586 and its attached expansion modules into the chosen VMEbus chassis slot (expansion modules should fill the slots immediately adjacent to the VMIVME-7586). While ensuring that the boards are properly aligned and oriented in the supporting board guides, slide the boards smoothly forward against the mating connector until firmly seated.

5. Connect all needed peripherals to the front panel. Each connector is clearly labelled on the front panel, and detailed pinouts are in Appendix A. Minimally, a keyboard and a monitor are required.

6. Apply power to the system. Several messages are displayed on the screen, including names, versions, and copyright dates for the various BIOS modules on the VMIVME-7586. Among the screen messages should be a VMIC product identifier such as:

   ```
   VME Microsystems International Corp.
   VMIVME-7586
   ```

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

8. If a drive module is present, install the operating system according to the manufacturer's instructions. See Section 9 on page 2-8 for assistance customizing popular operating systems such as MS-DOS and Windows for the VMIVME-7586.

SECTION 6 - FRONT PANEL CONNECTORS

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

a keyboard with a standard PC/AT connector to the VMIVME-7586. See Appendix A for connector pinouts and orientation.

SECTION 7 - PC/104 EXPANSION SITE

Expansion boards that are PC/104 compatible install directly to the VMIVME-7586's PC/104 Expansion Site (see Figure 2-1 on page 2-3). Figure 2-3 illustrates the mechanical connection between the VMIVME-7586 and a PC/104 board. Note that the attachment hardware shown (the nuts, screws, and standoffs) should be included with the PC/104 board and are not included with the VMIVME-7586.

Configuration information for PC/104 accessories should come with the board. The most common problems installing PC/AT accessories usually involve either an I/O addressing conflict or an interrupt conflict. See the I/O Port Map and the discussion regarding PC/AT Interrupts in Chapter 3 to determine port and interrupt configurations compatible with the VMIVME-7586.

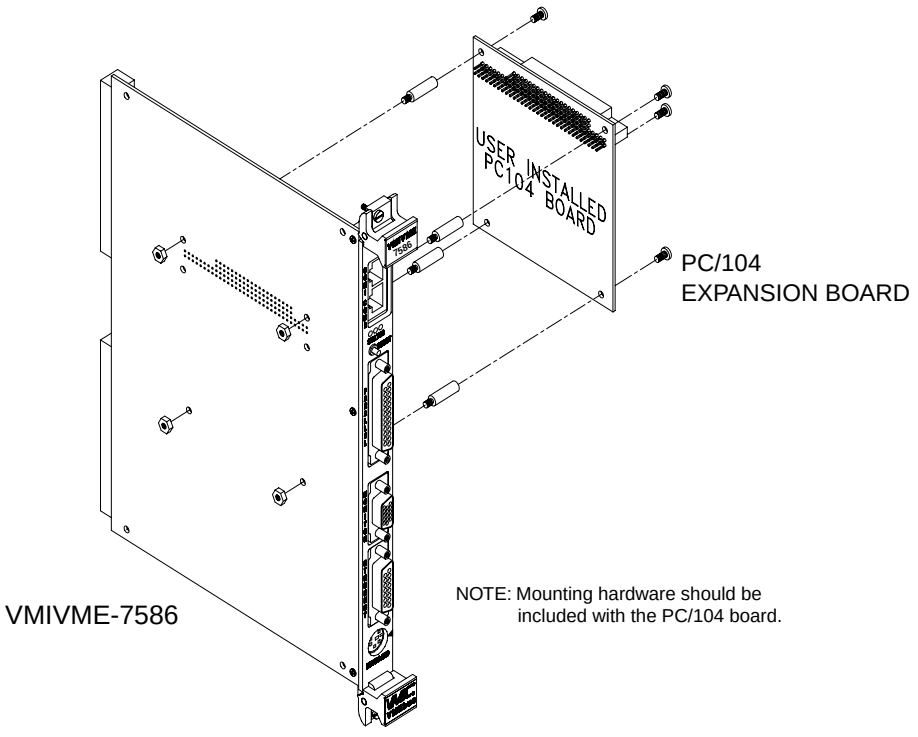


Figure 2-3 PC/104 Mechanical Connection

SECTION 8 - BIOS SETUP

The VMIVME-7586 has an on-board BIOS Setup program that controls many configuration options. These options are saved in a special nonvolatile, battery-backed memory chip and are collectively referred to as the board's "CMOS configuration." The CMOS configuration controls many details concerning the behavior of the hardware from the moment power is applied.

The VMIVME-7586 is shipped from the factory with no hard or floppy drives configured in CMOS. The BIOS Setup program must be run to configure the specific drives attached. It is recommended that the user follow the procedure for Quick BIOS Setup in Appendix D to properly configure the system.

SECTION 9 - CONFIGURING OPERATING SYSTEMS

The VMIVME-7586 is capable of running many popular operating systems, including MS-DOS, Microsoft Windows and Windows NT, and IBM's OS/2. In general, any operating system designed to run on an Intel-based PC should work well on the VMIVME-7586.

The operating systems should generally be installed according to the publisher's instructions, preferably with VMEbus access disabled (see Chapter 4). 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-7586 as a combination PC and VMEbus controller.

GENERAL RULE REGARDING OPERATING SYSTEMS

The VMIVME-7586 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-7586 is to exclude the two VMEbus Windows from the operating system's memory map.

By excluding the VMEbus windows, the operating system is prevented from inadvertently scanning or accessing addresses within the VMEbus

Windows. This prevents strange activity on the VMEbus, since any access within a VMEbus window is translated into a VMEbus access once VMEbus access is enabled.

The complete VMIVME-7586 memory map is presented in Chapter 3, but for reference the address ranges to be excluded for the VMEbus Windows are listed in Table 2-2.

Table 2-2 VMEbus Window Addresses

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

Note that although the Protected Mode VMEbus Window is only 32 Mbyte in size, all addresses within the Protected Mode VMEbus Window *and above* should be excluded up to $FFFD FFFF. This is because the addresses above the Protected Mode VMEbus Window up to the beginning of the reserved addresses at $FFFE 0000 are images of the Protected Mode VMEbus Window and are translated into VMEbus accesses just as if they were inside the actual VMEbus Window.

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

CONFIGURATION EXAMPLES

Configuring MS-DOS for the VMIVME-7586

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 Kbyte of RAM except for the BIOS.

Especially since the release of MS-DOS Version 5.0, it has become popular to supplement DOS with enhanced memory managers in order to take advantage of extended RAM above the traditional 640 Kbyte limit. Since the VMEbus Windows exist in this space above 640 Kbyte, 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-7586. Specify the page frame for EMM386.EXE by adding the parameter **FRAME=D000**. The resulting EMM386.EXE line might then look something like this:

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

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

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

Configuring Windows for the VMIVME-7586

Microsoft Windows will run well on the VMIVME-7586, 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

IN THIS CHAPTER:

SECTION 1 - CPU FUNCTIONAL OVERVIEW	**3-1**
SECTION 2 - PHYSICAL MEMORY	**3-2**
SECTION 3 - MEMORY AND I/O PORT MAPS	**3-3**
SECTION 4 - PC/AT INTERRUPTS	**3-8**
SECTION 5 - ENHANCED I/O PERIPHERAL PORTS	**3-12**
SECTION 6 - VIDEO GRAPHICS ADAPTER	**3-14**
SECTION 7 - PC/104 EXPANSION SITE	**3-15**

SECTION 1 - CPU FUNCTIONAL OVERVIEW

The VMIVME-7586 uses the 5x86 microprocessor as the CPU. The CPU's performance is achieved primarily with a 16 Kbyte write-back cache.

Instructions are executed in the Integer Unit and the Floating Point Unit. The Write-Back Cache Unit stores the most recently used data and the instructions and provides fast access to this information for the Integer and Floating Point Units.

When external memory access is required, the physical address is calculated by the Memory Management Unit and then passed to the Bus interface unit, which provides the interface between the external system board and the processor's internal execution and cache units.

SECTION 2 - PHYSICAL MEMORY

The VMIVME-7586 has one 72-pin SIMM socket, which supports a 1 M x 36, 2 M x 36, 4 M x 36, or 8 M x 36 factory-installed SIMM module for a total of 4, 8, 16, or 32 Mbyte of DRAM. The memory module is rated at 70 ns or faster. Memory can be accessed as bytes, words, or longwords. Note that this memory is not user-upgradable.

All DRAM on the VMIVME-7586 is dual-ported to the VMEbus. This means the memory is not only addressable by the local processor, but is also addressable through the VMEbus slave interface by another VMEbus master.

****
* CAUTION *
****

Caution must be used when sharing memory between the local processor and the VMEbus to prevent a VMEbus master from overwriting the local processor's operating system.

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

The VMIVME-7586 includes both the system and video BIOS in a single 128 K x 8 EPROM. The system portion of this ROM (at $F0000) is automatically shadowed, but for higher performance it is recommended that shadow RAM for the video BIOS be enabled (see Appendix D).

The VMIVME-7586 may also contain some external cache memory. If present, the cache consists of either 256 Kbyte or 1 Mbyte of direct-mapped level two (L2) fast cache RAM. External cache enhances system performance by reducing memory access time, since cache access is approximately five to ten times faster than DRAM access. Either size external cache uses a high-speed write-back design and may be disabled from the BIOS Setup program (see Appendix D).

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

SECTION 3 - MEMORY AND I/O PORT MAPS

MEMORY MAP

The memory map for the VMIVME-7586 is shown in Table 3-1. All systems share this same memory map, although a VMIVME-7586 with less than the full 32 Mbyte of DRAM does not fill the entire space reserved for On-Board Extended Memory.

Table 3-1 VMIVME-7586 Memory Map

MODE	MEMORY ADDRESS RANGE	SIZE	DESCRIPTION
PROTECTED MODE	$FFFF 0000 - $FFFF FFFF	64 Kbyte	ROM BIOS Image
	$8000 0000 - $FFFE FFFF	2047+ Mbyte	Reserved
	$4300 0000 - $7FFF FFFF ($4000 0000 - $7FFF FFFF)*	1024 Mbyte	Protected Mode VMEbus Windows*
	$0200 0000 - $42FF FFFF ($0200 0000 - $39FF FFFF)*	1040 Mbyte	Reserved
	$0010 0000 - $01FF FFFF	31 Mbyte	Reserved for On-Board Extended Memory (not filled on all systems)
REAL MODE	$F0000 - $FFFFF	64 Kbyte	ROM BIOS
	$E0000 - $EFFFF	64 Kbyte	Real Mode VMEbus Window
	$D0000 - $DFFFF	64 Kbyte	Reserved for Expansion BIOS (alternately may be used as a LIM/EMS page frame)
	$C8000 - $CFFFF	32 Kbyte	Reserved for Expansion BIOS
	$C0000 - $C7FFF	32 Kbyte	Video ROM
	$A0000 - $BFFFF	128 Kbyte	Video RAM
	$00000 - $9FFFF	640 Kbyte	User RAM/DOS RAM

* There are actually two modes of protected mode access. In the first mode, there are 16 Protected Mode VMEbus Windows of 16 Mbyte each in this space. Each window corresponds to a different type of VMEbus access (Extended, Standard, Supervisory, Nonprivileged, and so on). In the second, mode there is one 128 Mbyte of user-defined space with ten other user and supervisory spaces defined. See Chapter 4 for Protected Mode VMEbus addressing details.

I/O PORT MAP

The 5x86 CPU has special input/output instructions that access I/O peripherals residing in I/O addressing space (which is separate and distinct from memory addressing space). When the CPU decodes and executes an I/O instruction, it produces a 16-bit I/O address on lines A00-A15 and identifies the I/O cycle with the M/IO control line. Thus, the CPU has an independent 64 Kbyte I/O address space which is accessible as bytes, words, or longwords. Locations in I/O address space are often referred to as *ports*.

Standard PC/AT hardware circuitry decodes only ten 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-7586 incorporates all standard I/O peripherals of the PC/AT architecture such as keyboard, DMA, interrupt controllers, timers and real-time clock, as well as parallel and serial I/O ports, video registers, and floppy and hard drive task registers. The BIOS initializes and configures all these registers properly, so beware of adjusting these I/O ports directly. The assigned and user-available I/O addresses are summarized in the I/O Address Map, Table 3-2 on page 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 2 for details concerning hardware jumpers.

Table 3-2 VMIVME-7586 I/O Address Map

I/O ADDRESS RANGE	SIZE IN BYTES	HARDWARE DEVICE	PC/AT FUNCTION	ASSOCIATED JUMPERS †
$000 - $00F	16	ACC Micro 2040 Chip	DMA Controller 1 (Intel 8237A Compatible)	
$010 - $01F	16		Reserved	
$020 - $021	2	ACC Micro 2040 Chip	Master Interrupt Controller (Intel 8259A Compatible)	
$022 - $03F	30		Reserved	
$040 - $043	4	ACC Micro 2040 Chip	Programmable Timer (Intel 8254 Compatible)	
$044 - $05F	30		Reserved	
$060 - $064	5	ACC Micro 2040 Chip	Keyboard, Speaker, Eqpt. Config (Intel 8042 Compatible)	
$065 - $06F	11		Reserved	
$070 - $071	2	ACC Micro 2040 Chip	Real-Time Clock, NMI Mask	
$072 - $07F	14		Reserved	
$080 - $08F	16	ACC Micro 2040 Chip	DMA Page Registers	
$090 - $091	2		Reserved	
$092	1	ACC Micro 2040 Chip	Alt. Gate A20 / Fast Reset Register	
$093	1		Reserved	
$094	1	Super VGA Chip	POS102 Access Control Register	E3, E4 & E5
$095 - $09F	11		Reserved	
$0A0 - $0A1	2	ACC Micro 2040 Chip	Slave Interrupt Controller (Intel 8259A Compatible)	
$0A2 - $0BF	30		Reserved	
$0C0 - $0DF	32	ACC Micro 2040 Chip	DMA Controller 2 (Intel 8237A Compatible)	
$0E0 - $0EF	16		Reserved	
$0F0 - $0FF	16	ACC Micro 2040 Chip	Math Coprocessor / 2040 Config	
$100 - $101	2		Reserved	
$102	1	Super VGA Chip	POS102 Register	E3, E4 & E5
$103 - $13F	61		Reserved	

Table 3-2 VMIVME-7586 I/O Address Map (Continued)

I/O ADDRESS RANGE	SIZE IN BYTES	HARDWARE DEVICE	PC/AT FUNCTION	ASSOCIATED JUMPERS †
$140 - $14F	16	**Custom VMIC Hardware**	**VMEbus Interface Registers (see Chapter 4 for details)**	E24 & E11
$150 - $153	4	**Custom VMIC Hardware**	**Slave Size and Remap Registers**	
$154 - $16F	32		Reserved	
$170 - $177	8	Super I/O Chip	Secondary Hard Disk Controller	E9 & E8
$178 - $1EF	120		**User I/O**	
$1F0 - $1F7	8	Super I/O Chip	Primary Hard Disk Controller	E9 & E8 (default)
$1F8 - $277	128		**User I/O**	
$278 - $27F	8	Super I/O Chip	LPT2 Enhanced Parallel I/O (EPP/ECP Compatible)	E6 & E7
$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	E9 & E8
$378 - $37F	8	Super I/O Chip	LPT1 Enhanced Parallel I/O (EPP/ECP Compatible)	E6 & E7 (default)
$380 - $3AF	48		Reserved	
$3B0 - $3BB	16	Monochrome Adapter***** or Super VGA Chip	Monochrome Adapter***** or VGA Monochrome Video	E1, E2 & E3‡
$3BC - $3BF	4	Super I/O Chip	Alternate LPT1 Enhanced Parallel I/O (ECP Compatible)	E6 & E7
$3C0 - $3CF	16	Super VGA Chip	VGA Adapter	E1, E2 & E3
$3D0 - $3DF	16	CGA Adapter***** or Super VGA Chip	CGA Adapter***** or VGA Color Video	E1, E2 & E3‡
$3E0 - $3E7	8		Reserved	
$3E8 - $3EE	7	UART*****	COM3 Serial I/O*****	
$3F0 - $3F7	8	Super I/O Chip	Primary Floppy Disk Controller	E9 & E8 (default)

Table 3-2 VMIVME-7586 I/O Address Map (Continued)

I/O ADDRESS RANGE	SIZE IN BYTES	HARDWARE DEVICE	PC/AT FUNCTION	ASSOCIATED JUMPERS †
$3F8 - $3FE	7	Super I/O Chip	COM1 High-Speed Serial I/O (16550A Compatible)	
$3FF	1		Reserved	
ADDRESSES BEYOND $3FF ARE NOT USABLE BY PC/104 EXPANSION BOARDS				
$400 - $4FF	256	VIC Chip **§**	VMEbus Interface controller (default)**§**	
$500 - $501	2		Reserved	
$502 - $508	7	Custom VMIC Hardware**§**	Interrupt ID Registers (default)**§**	
$509 - $677			Reserved	
$678 - $67A	3	Super I/O Chip	LPT2 ECP Registers	E6 & E7
$67B - $777	253		Reserved	
$778 - $77A	3	Super I/O Chip	LPT1 ECP Registers	E6 & E7 (default)
$77B - $7BB	65		Reserved	
$7BC - $7BE	3	Super I/O Chip	Alternate LPT1 ECP Registers	E6 & E7
$7BF - $46E7	16169		Reserved	
$46E8	1	Super VGA Chip	VGA Adapter Sleep Register	E1, E2 & E3
$46E9 - $FFFF	47383		Reserved	

† The Associated Jumpers can affect whether or not the listed hardware is addressed at the given location. Hardware jumper settings are discussed in Chapter 2.

***** While these I/O ports are reserved for the listed functions, they are not implemented on the VMIVME-7586. 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 Appendix D for details concerning BIOS Setup).

§ The locations of the VIC chip and Interrupt ID Registers within I/O space are determined by the VIC Base Register (see Chapter 4 for programming details). The default VIC Base Register value positions the VIC at I/O $0400 and the Interrupt ID Registers at I/O $502.

SECTION 4 - PC/AT INTERRUPTS

In addition to an I/O port address, an I/O device often has a separate hardware interrupt line assignment. Assigned to each interrupt line is a corresponding interrupt vector in the 256-vector interrupt table at $00000 to $003FF in memory. The 16 maskable interrupts and the single nonmaskable interrupt (NMI) are listed in Table 3-3 along with their functions. Table 3-4 on page 3-9 details the vectors in the interrupt vector table.

Table 3-3 PC/AT Hardware Interrupts

IRQ	AT FUNCTION	COMMENTS
NMI	Parity Errors (Must be enabled in BIOS Setup)	**Used by VMIVME-7586 VMEbus Interface**
0	System Clock/Calendar	Set by BIOS Setup
1	System Clock/Calendar	Set by BIOS Setup
2	Duplexed to IRQ9	
3	COM2 / COM4	
4	COM1 / COM3	
5	LPT2	**Used by VMIVME-7586 VMEbus Interface** (alternate choice, set by jumpers E6 & E7)
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-7586 VMEbus Interface**
12	Not Assigned	**Used by VMIVME-7586 VMEbus Interface**
13	Math Coprocessor	
14	AT Hard Drive	
15	Not Assigned	

Table 3-4 PC/AT Interrupt Vector Table

INTERRUPT #		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
00	0		Divide Error	Same as Real Mode
01	1		Debug Single Step	Same as Real Mode
02	2	NMI	Memory Parity Error, **VMEbus Interrupts**	Same as Real Mode (Must be enabled in BIOS Setup)
03	3		Debug Breakpoint	Same as Real Mode
04	4		ALU Overflow	Same as Real Mode
05	5		Print Screen	Array Bounds Check
06	6			Invalid OpCode
07	7			Device Not Available
08	8	IRQ0	Timer Tick	Double Exception Detected
09	9	IRQ1	Keyboard Input	Coprocessor Segment Overrun
0A	10	IRQ2	BIOS Reserved	Invalid Task State Segment
0B	11	IRQ3	COM2 Serial I/O	Segment Not Present
0C	12	IRQ4	COM1 Serial I/O	Stack Segment Overrun
0D	13	IRQ5	Fixed Disk Controller or VMEbus Interrupts (alt)	General Protection Violation
0E	14	IRQ6	Floppy Disk Controller	Page Fault
0F	15	IRQ7	LPT1 Parallel I/O	Unassigned
10	16		BIOS Video I/O	Coprocessor Error
11	17		Eqpt Configuration Check	Same as Real Mode
12	18		Memory Size Check	Same as Real Mode
13	19		XT Floppy/Hard Drive	Same as Real Mode
14	20		BIOS Comm I/O	Same as Real Mode
15	21		BIOS Cassette Tape I/O	Same as Real Mode
16	22		BIOS Keyboard I/O	Same as Real Mode
17	23		BIOS Printer I/O	Same as Real Mode
18	24		ROM BASIC Entry Point	Same as Real Mode
19	25		Bootstrap Loader	Same as Real Mode
1A	26	IRQ8	Real-Time Clock	Same as Real Mode
1B	27		Control/Break Handler	Same as Real Mode
1C	28		Timer Control	Same as Real Mode

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

INTERRUPT #		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
1D	29		Video Parameter Table Pntr	Same as Real Mode
1E	30		Floppy Parm Table Pntr	Same as Real Mode
1F	31		Video Graphics Table Pntr	Same as Real Mode
20	32		DOS Terminate Program	Same as Real Mode
21	33		DOS Function Entry Point	Same as Real Mode
22	34		DOS Terminate Handler	Same as Real Mode
23	35		DOS Control/Break Handler	Same as Real Mode
24	36		DOS Critical Error Handler	Same as Real Mode
25	37		DOS Absolute Disk Read	Same as Real Mode
26	38		DOS Absolute Disk Write	Same as Real Mode
27	39		DOS Program Terminate, Stay Resident	Same as Real Mode
28	40		DOS Keyboard Idle Loop	Same as Real Mode
29	41		DOS CON Dev. Raw Output	Same as Real Mode
2A	42		DOS 3.x+ Network Comm	Same as Real Mode
2B	43		DOS Internal Use	Same as Real Mode
2C	44		DOS Internal Use	Same as Real Mode
2D	45		DOS Internal Use	Same as Real Mode
2E	46		DOS Internal Use	Same as Real Mode
2F	47		DOS Print Spooler Driver	Same as Real Mode
30-60	48-96		Reserved by DOS	Same as Real Mode
61-66	97-102		User Available	Same as Real Mode
67-71	103-113		Reserved by DOS	Same as Real Mode
72	114	IRQ10		Same as Real Mode
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 an 8-bit vector to each PIC 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-7586 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-7586's PC/104 expansion port.

Figure 3-1 depicts the VMIVME-7586 interrupt logic pertaining to VMEbus operations. Note that the NMI interrupt generated by the VMIVME-7586 must be enabled in the BIOS Setup (see Appendix D).

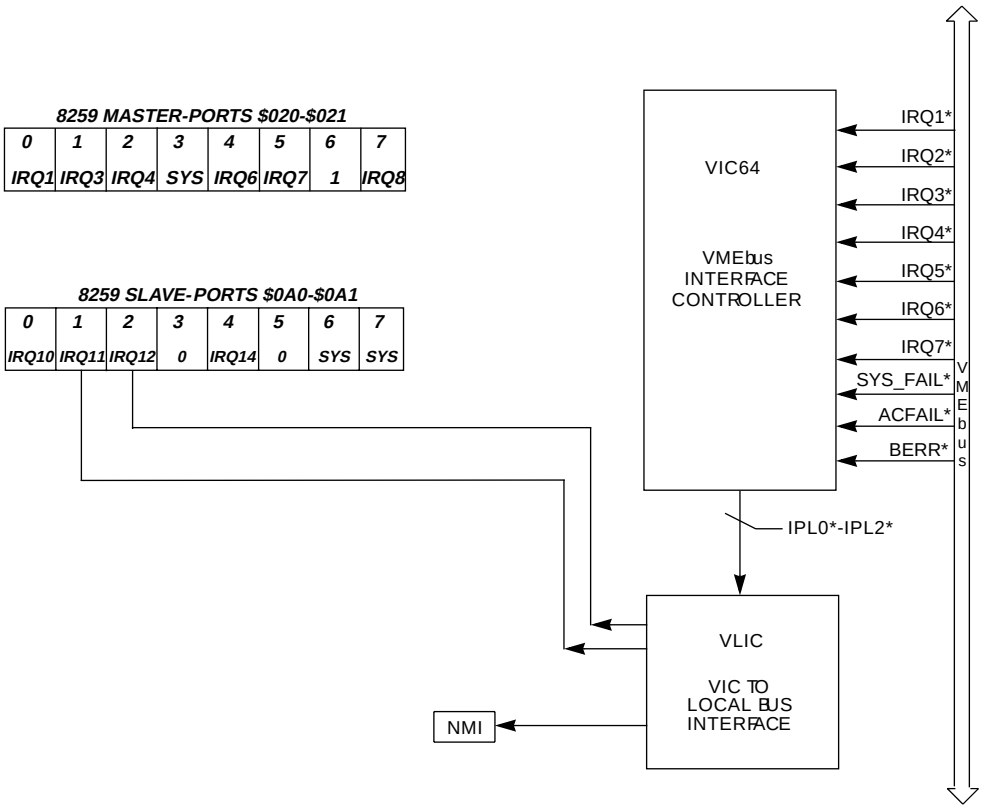


Figure 3-1 Connections Between the PC Interrupt Logic Controller and the VMEbus

SECTION 5 - ENHANCED I/O PERIPHERAL PORTS

The VMIVME-7586 incorporates the Intel 82091AA Advanced Integrated Peripheral chip, also known as the AIP or Super I/O chip. This chip provides the VMIVME-7586 with a floppy drive controller, IDE hard drive interface, two serial ports, and one parallel port. While the floppy and hard drive interfaces are standard, the AIP has two high-performance advantages over ordinary ISA bus peripherals: high-speed 16550 compatible serial ports, and a fast multifunctional, bidirectional parallel port.

SERIAL PORTS

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

In order to maintain compatibility with older, non-FIFO-aware communications programs, the VMIVME-7586's BIOS does not enable the serial port FIFO buffers. Most contemporary communications programs and drivers automatically detect a 16550 compatible port and enable the buffers themselves.

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

PARALLEL PORT

The VMIVME-7586's enhanced parallel port may be configured for any one of four modes as defined by IEEE Standard 1284 shown in Table 3-5.

Table 3-5 Parallel Port Modes

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

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

In the ISA compatible and PS/2 compatible modes, software controls the parallel port handshake signals during data transfers. The EPP protocol increases throughput by generating an automatic handshake in hardware.

The ECP protocol not only implements a hardware handshake, but adds a 16 byte FIFO with DMA capability to reduce processor overhead.

The VMIVME-7586's AIP chip supports all parallel port modes in Table 3-5 on page 3-13, except the EPP mode is not supported if the alternate LPT1 I/O address of $3BC-$3BF is used (see Chapter 2 for hardware setup details). The BIOS initializes the port to ISA-compatible mode in order to maximize compatibility with older applications, but newer software will recognize the advanced port and reconfigure it accordingly.

SECTION 6 - VIDEO GRAPHICS ADAPTER

The monitor port on the VMIVME-7586 is controlled by a Cirrus Logic chip with 1 Mbyte video DRAM. The video controller chip is hardware and BIOS compatible with the IBM EGA and VGA standards and also supports VESA high-resolution and extended video modes. Table 3-6 shows the popular graphics video modes supported by the Cirrus Logic video chip.

Table 3-6 Common Supported Graphics Video Resolutions

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

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

The VMIVME-7586's processor has 16-bit access to video memory with no wait states. Video I/O registers are accessed at the maximum ISA bus rate. In addition, the Cirrus Logic video controller supports up to a 64 x 64 pixel hardware cursor.

Floppy disks supplied with the VMIVME-7586 contain video drivers for Windows and other popular programs and operating systems. Follow the instructions on the floppy disk label to install the drivers using an interactive installation program.

NOTE:

WHEN AN ALTERNATE PC/104 VIDEO ADAPTER IS REQUIRED, REMOVAL OF JUMPERS E3, E4, AND E5 IS NECESSARY. THE REMOVAL OF THESE JUMPERS ENSURES THAT THE ON-BOARD VIDEO CONTROLLER IS DISABLED AND THAT THE AUXILIARY VIDEO BIOS WILL BE RECOGNIZED.

SECTION 7 - PC/104 EXPANSION SITE

The VMIVME-7586 supports industry-standard expansion modules through PC/104: the IEEE P996.1 Standard for Compact Embedded-PC Modules. PC/104 modules are available from third-party vendors for display devices, PCMCIA adapters, Ethernet and SCSI interfaces, and other I/O functions.

VMIC's VMIVME-7432 PC/104 to ISA Adapter is designed specifically to mate with the VMIVME-7586, allowing it to use a standard half-length ISA bus board in the same chassis. This expansion allows low-cost or specialty ISA bus boards to be used in the same chassis with VMEbus equipment.

In order to allow the VMIVME-7450 or VMIVME-7452 hard and floppy drive module to be attached along with the VMIVME-7432, the VMIVME-7432 duplicates the hard drive and floppy drive controller functions of the VMIVME-7586. The on-board VMIVME-7586 hard and floppy controller should, therefore, be disabled when a VMIVME-7432 is installed.

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

CHAPTER

4

VMEbus FUNCTIONS

IN THIS CHAPTER:

SECTION 1 - INTRODUCTION	**4-1**
SECTION 2 - VMEbus INTERFACE	**4-2**
SECTION 3 - VMEbus MASTER OPERATION	**4-7**
SECTION 4 - VMEbus SLAVE OPERATION	**4-16**
SECTION 5 - SYSTEM CONTROLLER FUNCTIONS	**4-31**
SECTION 6 - VMEbus INTERRUPT HANDLING	**4-32**
SECTION 7 - VMEbus INTERRUPTER	**4-39**
SECTION 8 - VMEbus REQUESTER	**4-40**
SECTION 9 - READ-MODIFY-WRITE CYCLES	**4-40**
SECTION 10 - BLOCK TRANSFERS	**4-42**
SECTION 11 - VME64 FUNCTIONS	**4-45**
SECTION 12 - BYTE ORDERING	**4-46**
SECTION 13 - VMIVME-7586 REGISTERS	**4-52**

SECTION 1 - INTRODUCTION

This chapter serves as both a programmer's instruction manual and a reference guide to the VMEbus interface of the VMIVME-7586. The chapter begins with sections concerning general VMEbus interface issues and then delves into detailed operations based on function. Finally, I/O maps and register-level descriptions of the VMIVME-7586's VMEbus interface are provided for reference.

The register details are provided last for easy reference for the programmer familiar with the VMIVME-7586, but the novice will need to study the reference section as well. The first-time VMIVME-7586 user is strongly urged to read through the functional sections with a copy of the register and bit maps (beginning on page 4-53) close at hand.

SECTION 2 - VMEbus INTERFACE

This section discusses how the VMIVME-7586 interfaces with the VMEbus. At powerup or after a system reset (but not after a soft reset, or **<Ctrl+Alt+Del>** reset), the VMIVME-7586 is "isolated" from the VMEbus. The user must first set the VME Enable bit in the General Purpose Command Register before access to the VMEbus is possible so that the VMIVME-7586 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-7586, the user may proceed with VMEbus activities from a known state and in an orderly manner.

VMEbus INTERFACE OVERVIEW

At powerup, the 5x86 processor enters the Real Mode. MS-DOS is typically loaded and run in this mode. It is important to understand that only 1 Mbyte of memory is available in Real Mode. Windows, and OS/2 on the other hand, run in Protected Mode and can utilize all available extended memory.

The VMIVME-7586 can access the VMEbus in Real Mode or Protected Mode. As a VMEbus interrupt processor the VMIVME-7586 can also perform IACK* cycles in either Real Mode or Protected Mode. Figure 4-1 on page 4-3 shows the functional interconnections between the VMIVME-7586 and the VMEbus. Note that the VMEbus is connected to the 5x86 local bus.

In Real Mode, the VMIVME-7586 uses a paging scheme to access any part of the 4 Gbyte of VMEbus addressing space 64 Kbyte at a time through the 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 are controlled by the Extended/Standard Access Register (for bits A16-A31). Think of the Extended/Standard Access Register as the value that controls which 64 Kbyte block of VMEbus space is available within the Real Mode VMEbus Window.

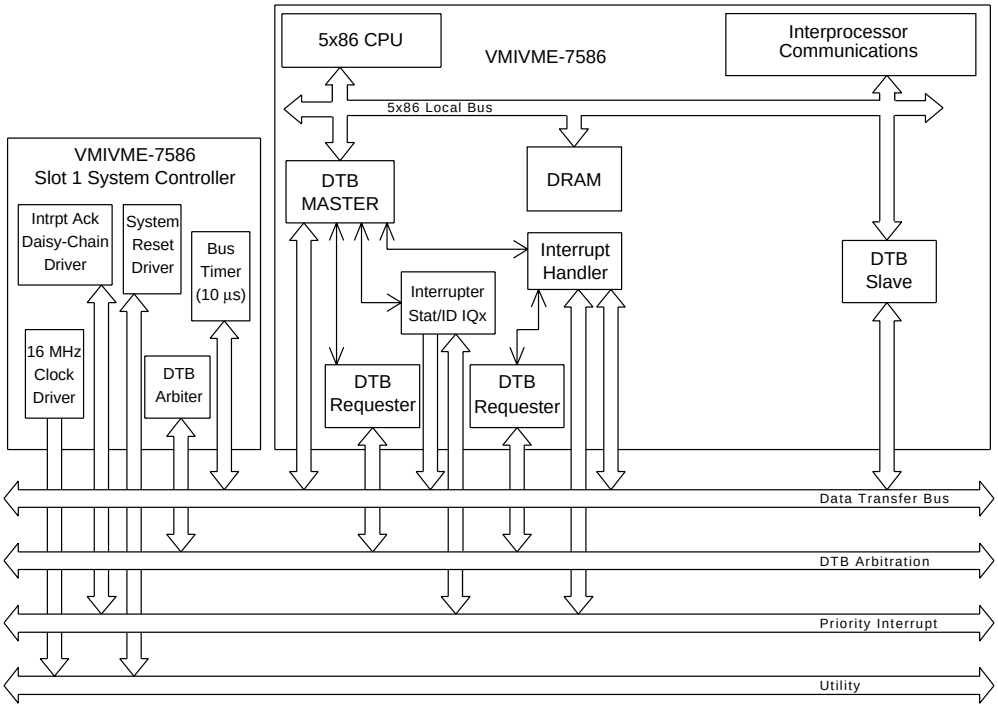


Figure 4-1 VMIVME-7586 VMEbus Functions

In Protected Mode, the VMIVME-7586 can access the VMEbus through the Protected Mode VMEbus Window, which occupies 1024 Mbyte beginning at $4300 0000. This 1024 Mbyte space is divided into 16 equal spaces of 16 Mbyte, each corresponding to the VMEbus but with different VMEbus Address Modifiers. By supplying the extended VMEbus address bits A24-A31 in the Extended/Standard Address Register, all 4 Gbyte of VMEbus memory may be addressed in Protected Mode, 16 Mbyte at a time.

An alternative method for protected mode access maybe implemented by enabling 128 Mbyte access. This mode, beginning at address $4000 0000, provides the user with 128 Mbyte of user-defined space and also supports automatically generated address modifiers for short, standard, and extended accesses.

VMEbus INTERFACE HARDWARE

The VMIVME-7586 VMEbus hardware interface consists of a Cypress VIC64 VMEbus Interface Controller (hereafter referred to as the VIC or the VIC64), its companion VIC-to-Local bus Interface Circuitry (referred to as the VLIC), and three Cypress CY7C964.

The VIC is a high-performance industry-proven single integrated circuit. The VIC was developed through joint efforts of Cypress Semiconductor and the VMEbus Technology Consortium under the auspices of the VMEbus International Trade Association (VITA).

Figure 4-2 shows the interconnection between the 5x86 local bus, the VLIC block (VIC-to-Local bus Interface Circuitry), the VIC64 and companion CY7C964s, and the VMEbus.

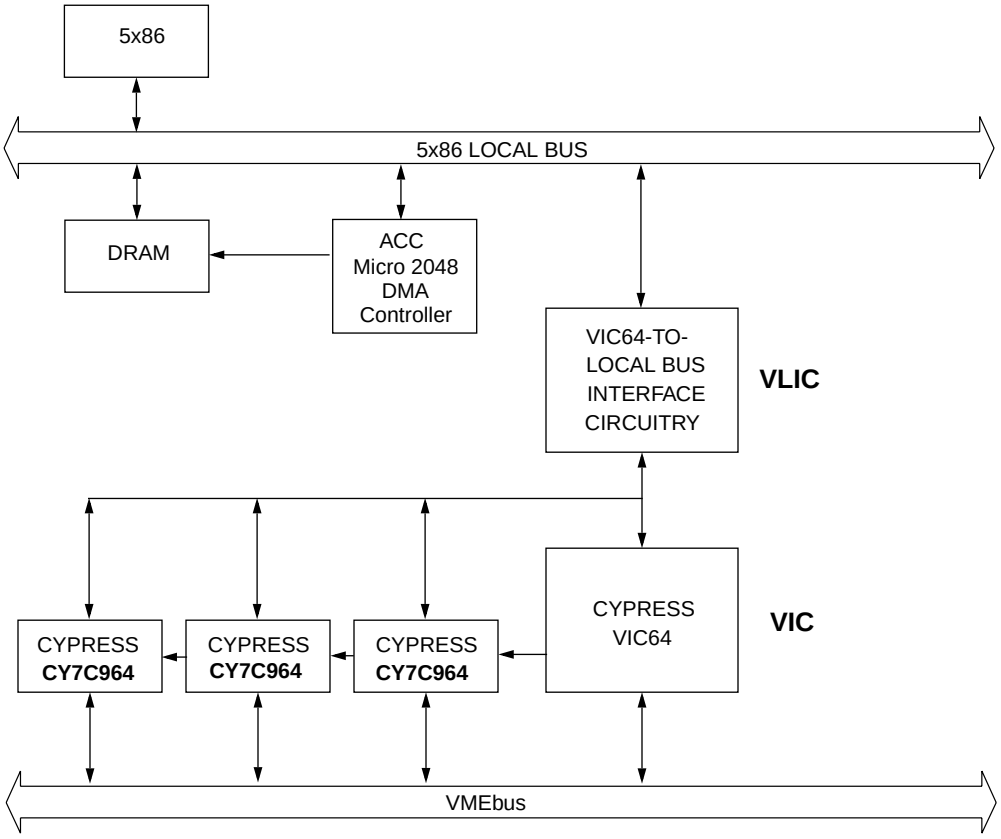


Figure 4-2 VMEbus Interface Block Diagram

The CPU interface of the VIC was designed to be part of a Motorola 68030 system. An interface is required to connect the 5x86 bus to the 68030-like CPU interface of the VIC. This connection is accomplished by the VLIC using the functional modules illustrated in Figure 4-3. This interface allows the VIC to access the local bus (DRAM), and allows the 5x86 to access VIC registers and the VMEbus using the VIC.

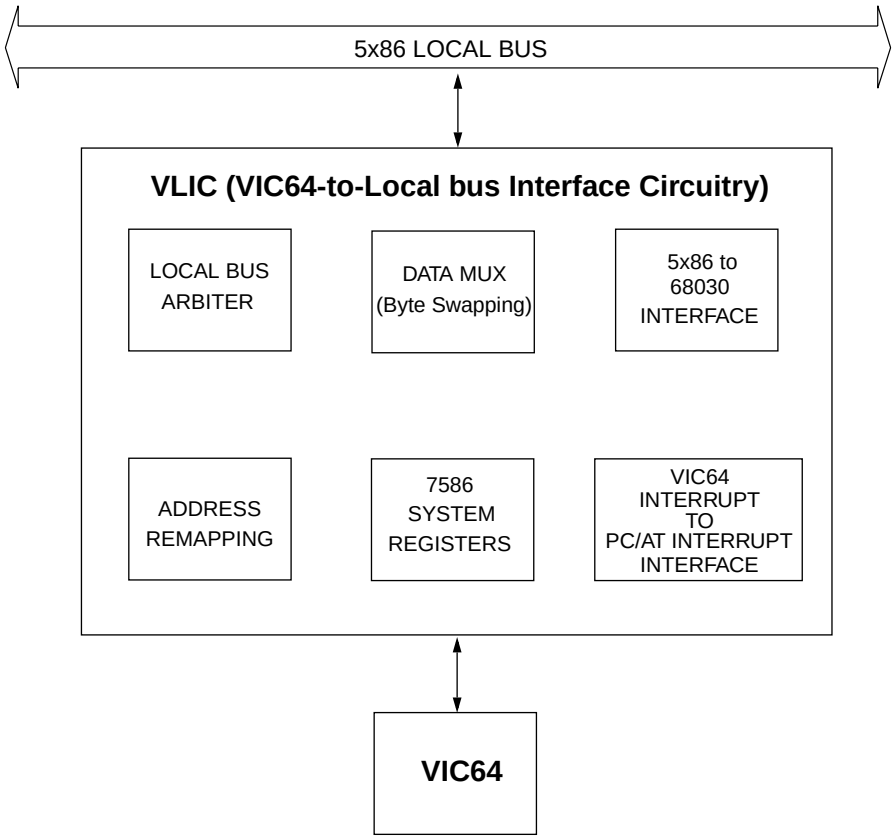


Figure 4-3 VLIC Block Diagram

The local bus arbiter arbitrates the 5x86 bus between the VIC and the ACC Micro 2048. The VIC requests the local bus when a VMEbus slave access to the dual-ported DRAM is requested. The 2048 requests the local bus when a DRAM refresh is required or when the 2048 DMA controller needs the bus.

The data multiplexer function is used to implement byte swapping. Byte swapping is implemented in both directions (both VMIVME-7586 acting as VMEbus master and VMIVME-7586 being accessed as a VMEbus slave).

The VLIC contains system registers that allow the VMIVME-7586 VMEbus interface to be programmed. These are discussed in detail in Section 13.

The VLIC also contains hardware to translate the 68030-like interrupt structure of the VIC to the PC/AT interrupt structure.

Slave remapping functionality is also configured in the VLIC circuit block.

PROGRAMMING THE VMEbus INTERFACE

The VMIVME-7586 VMEbus interface is illustrated in Figure 4-2 on page 4-4 as consisting of the VIC64, three CY7C964s, and miscellaneous circuitry described collectively as the VIC64-to-Local bus Interface Circuitry (VLIC). The interface is configured by programming the VIC64, the three CY7C964 chips, and a set of System Registers. These System Registers are shown in Table 4-8 on page 4-53.

Additionally, two jumpers configure the VMIVME-7586 as system controller and enable the bus-timeout timer. See Chapter 3 for hardware jumper details. System controller functions are described later in this chapter.

All VMEbus interface registers are in I/O space and consequently are accessed using 5x86 I/O instructions. The VMIVME-7586 contains 12 System Registers, three Interrupt Acknowledge Registers, and 58 VIC Registers. Of these, only the System Registers have fixed I/O addresses. The base location for the Interrupt Acknowledge and VIC Registers is programmed through one of the System Registers. See Section 13 for a complete register map and detailed descriptions.

The VIC contains 58 byte-wide registers. The base address of these registers is programmable. The VIC Base Register is used to program bits A15-A11 of the base address (see the detailed description of the VIC Base Register on page 4-61). The three interrupt acknowledge registers use this same base address. This register is cleared at powerup, providing a default base address of I/O $400. Setting the VIC Enable bit enables the base address and allows the VIC and the three Interrupt Acknowledge Registers to be accessed. Attempts to access the VIC registers or Interrupt Acknowledge Registers without the VIC Enable bit set initiates an access to the ISA bus. This access terminates properly but retrieves undefined information if the access was a read.

In addition to setting the VIC Enable bit, address bit A10 must be set when accessing the VIC Registers or Interrupt Acknowledge Registers. This prevents conflicts that may occur between the VIC register addresses and the AT I/O resources located between I/O address $000 and $3FF.

To access the VIC registers, program the VIC Base Register, then perform a byte I/O access to the desired VIC register according to the VIC Register Map on page 4-53. Please note that the VIC only allows byte accesses.

For example, the VIC System Reset Register (SRR) has an offset address of $E3. If the A11-A15 bit field in the VIC Base Register contained the value $00000, the I/O address used to access this register would be I/O $4E3.

Similarly, the addresses of the three Interrupt Acknowledge Registers also depend on the contents of the VIC Base Register. Like the VIC registers, A10 of the I/O address must be set. Address bit A8 must also be set to access the Interrupt Acknowledge Registers. Section 13 details the location and function of the Interrupt Acknowledge Registers.

SECTION 3 - VMEbus MASTER OPERATION

The VMIVME-7586 VMEbus interface allows the 5x86 to access the VMEbus while in Real Mode or Protected Mode. In Real Mode, the VMIVME-7586 accesses the VMEbus using a paging scheme to address any part of the 4 Gbyte VMEbus through a 64 Kbyte window at $E0000. Prior to accessing this segment, address modifier information must be programmed into the VIC Address Modifier Source Register. In addition, the Extended/Standard Address Register must also be programmed.

When in Protected Mode, the VMIVME-7586 accesses the VMEbus using a paging scheme similar to that described for Real Mode; however, the window is 1024 Mbyte. Within the 1024 Mbyte window are sixteen 16 Mbyte regions that produce the desired address modifier codes automatically. User-defined address modifier codes are also available by addressing the VMEbus through a user area in conjunction with the VIC Address Modifier Source Register.

An alternative method for protected mode access may be implemented by enabling 128 Mbyte access. This mode, beginning at address $4000 0000, provides the user with 128 Mbyte of user-defined space and also supports automatically generated address modifiers for short, standard and extended accesses.

REAL MODE ACCESS

When in Real Mode, accessing the 64 Kbyte of memory located at physical address $E0000 causes an access to VMEbus (provided the VME Enable bit is set in the General Purpose Command Register). The desired address modifier code is contained in the VIC Address Modifier Source Register. The upper address bits are contained in the Extended/Standard Address Register (VMEbus address bits A31-A16). The lower address bits (VMEbus address bits A15-A0) are driven directly by the CPU.

See Figure 4-4 on page 4-9 for a graphic representation of Real Mode VMEbus access.

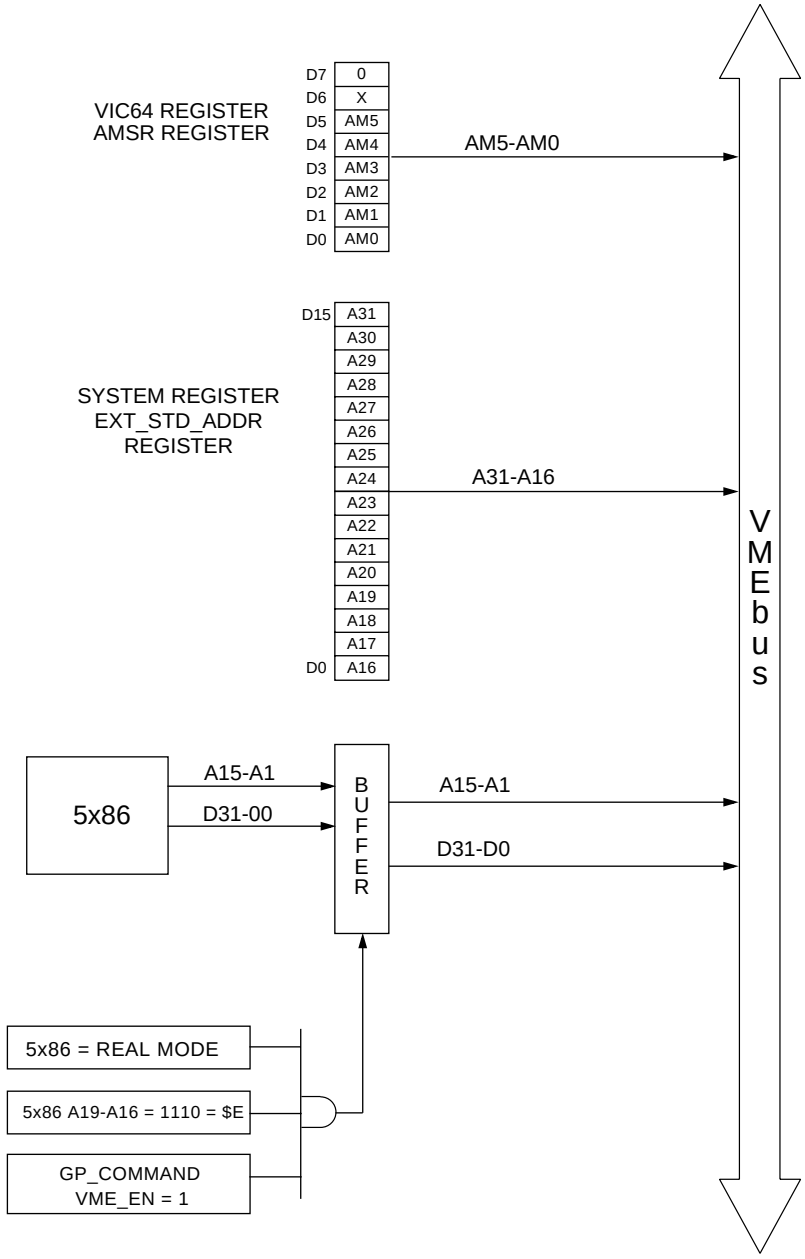


Figure 4-4 Real Mode VMEbus Access

After powerup, accessing VMEbus in Real Mode requires at least the following steps:

1. Initialize the VIC Base Register. Program the VIC base address and enable access to the VIC by setting the VIC Enable bit.

2. Initialize the VIC Address Modifier Source Register. Initialize bits 5-0 with the desired VMEbus address modifier code. Set the SysFail Mask bit in the VIC Mailbox Semaphore Register ICR7. Note that other VIC registers may also require programming depending on arbiter and requester options.

3. Initialize the General Purpose Command Register by setting the VME Enable bit. Other bits may be set depending on application.

4. Initialize the Extended/Standard Address Register. This sets the upper VMEbus address bits as shown in Figure 4-4 on page 4-9, effectively controlling which 64 Kbyte segment of VMEbus addressing space appears in the Real Mode VMEbus Window.

5. Perform a memory access between local addresses $E0000 and $EFFFF. A byte, word, or longword access generates a VMEbus access.

PROTECTED MODE ACCESS

In Protected Mode, a VMEbus access is performed when the 5x86 accesses memory with A30=1, A25=1, and A24=1. The upper VMEbus address bits (A31-A24) are driven from the Extended/Standard Address Register. The lower VMEbus address bits (A23-A1) are driven directly by the 5x86.

The 5x86 address bits A29-A26 are used to decode between sixteen unique 16 Mbyte memory regions. Each region corresponds to the same VMEbus address, but with different address modifier codes. The address size and type for each region are shown in Table 4-1 on page 4-11. See Figure 4-5 on page 4-12 for a pictorial representation of Protected Mode access.

Table 4-1 Protected Mode VMEbus Address Modifiers. **Note: I/O Port $143 must be set to $00**.

A31	A30	A29 (ASIZ1)	A28 (ASIZ0)	A27 (FC2)	A26 (FC1)	A25	A24	ADDRESS MODIFIER AND MODE
0	1	1	1	1	1	1	1	A24 Supervisory-program
0	1	1	1	1	0	1	1	A24 Supervisory-data
0	1	1	1	0	1	1	1	A24 Nonprivileged-program
0	1	1	1	0	0	1	1	A24 Nonprivileged-data
0	1	1	0	1	1	1	1	A16 Supervisory-program
0	1	1	0	1	0	1	1	A16 Supervisory-data
0	1	1	0	0	1	1	1	A16 Nonprivileged-program
0	1	1	0	0	0	1	1	A16 Nonprivileged-data
0	1	0	1	1	1	1	1	A32 Supervisory-program
0	1	0	1	1	0	1	1	A32 Supervisory-data
0	1	0	1	0	1	1	1	A32 Nonprivileged-program
0	1	0	1	0	0	1	1	A32 Nonprivileged-data
0	1	0	0	1	1	1	1	User-defined (VIC AMSR 5-0)*
0	1	0	0	1	1	1	1	User-defined (VIC AMSR 5-3 + 6)*
0	1	0	0	1	0	1	1	User-defined (VIC AMSR 5-0)*
0	1	0	0	1	0	1	1	User-defined (VIC AMSR 5-3 + 5)*
0	1	0	0	0	1	1	1	User-defined (VIC AMSR 5-0)*
0	1	0	0	0	1	1	1	User-defined (VIC AMSR 5-3 + 2)*
0	1	0	0	0	0	1	1	User-defined (VIC AMSR 5-0)*
0	1	0	0	0	0	1	1	User-defined (VIC AMSR 5-3 + 1)*

* These address modifiers depend upon the value of the AM2-0 Option bit (bit 7) in the VIC Address Modifier Source Register (AMSR). Shaded fields define the address modifier when the AM2-0 Option bit is set. The bit is clear by default. See the description of the Address Modifier Source Register, page 4-85 for additional information.

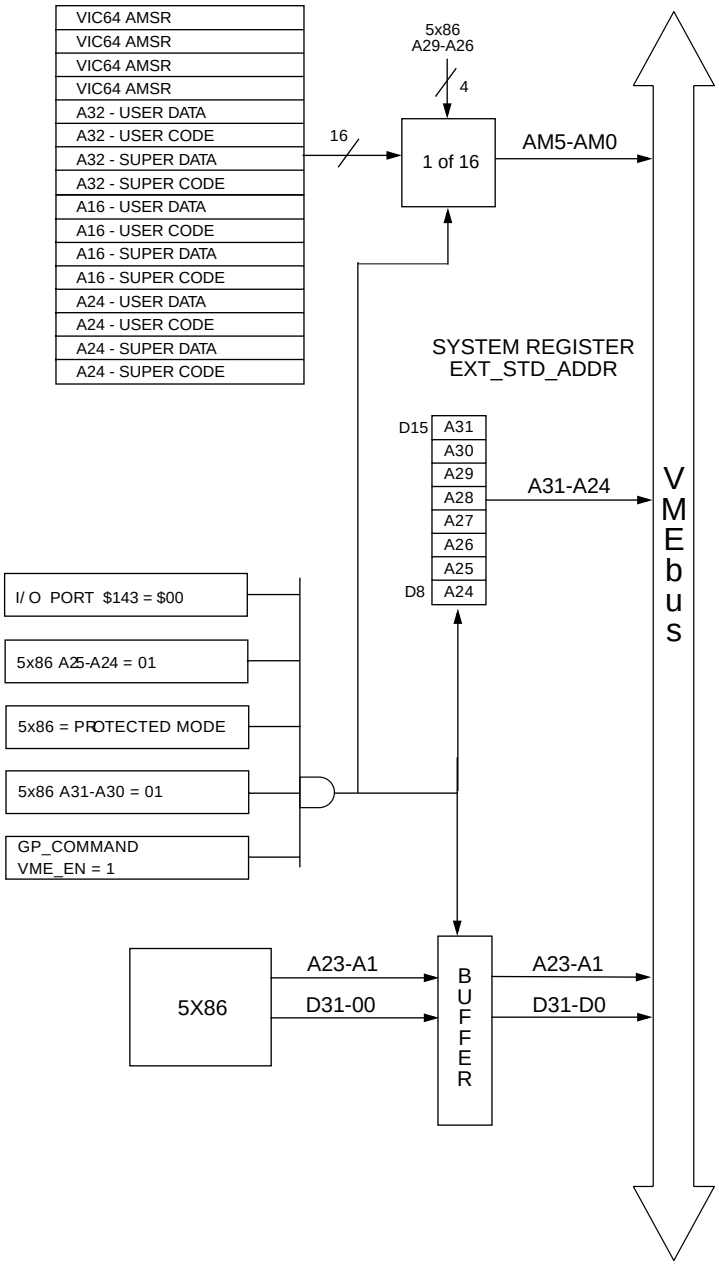


Figure 4-5 Protected Mode VMEbus Access

After powerup, accessing VMEbus in Protected Mode requires at least the following steps:

1. Initialize the VIC Base Register. Program the VIC base address and enable access to the VIC by setting the VIC Enable bit.

2. Initialize the VIC Address Modifier Source Register. Clear bit 7 and initialize bits 5-0 with the desired VMEbus address modifier code. Note that the Address Modifier Source Register is *only* used when 5x86 address bits A29 and A28 are clear. Set the SysFail Mask bit in the VIC Mailbox Semaphore Register. Other VIC registers may be programmed depending on arbiter/requester options.

3. Initialize the General Purpose Command Register by setting the VME Enable bit. Other bits may be set depending on application.

4. Initialize the Extended/Standard Address Register. This sets the upper VMEbus address bits as shown in Figure 4-5 on page 4-12, effectively controlling which 16 Mbyte segment of VMEbus addressing space appears in the Protected Mode VMEbus Window. For Protected Mode operation, bits 0-7 of the register are not used.

5. Perform memory access at an address between $4300 0000 and $7FFF FFFF, depending upon what address modifier is desired (see Table 4-1 on page 4-11). A byte, word, or longword access generates a VMEbus access.

NOTE:

5X86 ADDRESS BITS A25-A24 MUST BE SET TO 1 DURING VMEbus PROTECTED MODE ACCESS.

128 Mbyte - PROTECTED MODE ACCESS

An alternative method for protected mode access may be implemented by enabling 128 Mbyte access by writing a $01 to I/O port $143. In this mode only A27-A31 will come from the Extended/Standard Address Register. A0-A26 will then be generated by the user. The mode will provide the user with 128 Mbyte of user-defined space, and also supports automatically generated address modifiers for short, standard, and extended accesses.

The 5x86 address bits A29-A26 are used to decode between eleven unique memory regions. Please note that A26 is used both to drive the VMEbus and as a decode for the access type required. The address size and type for each region are shown in Table 4-2. See Figure 4-6 on page 4-15 for a pictorial representation of 128 Mbyte Protected Mode access.

Table 4-2 128 Mbyte Master Window Definitions

A31	A30	A29 (ASIZ1)	A28 (ASIZ0)	A27 (FC2)	A26 (FC1)	ADDRESS MODIFIER AND MODE	ADDRESS RANGE	SIZE IN Mbytes
0	1	0	0	X	X	user-defined	$4000 0000 - 47FF FFFF	128
0	1	0	1	0	0	A32 User Data / User Block	$5000 0000 - 53FF FFFF	32
0	1	0	1	0	1	A32 User Pgm / User Block	$5400 0000 - 57FF FFFF	32
0	1	0	1	1	0	A32 Super Data / Super Block	$5800 0000 - 5BFF FFFF	32
0	1	0	1	1	1	A32 Super Pgm / Super Block	$5C00 0000 - 5FFF FFFF	32
0	1	1	0	0	X	A16 User Access	$6000 0000 - 67FF FFFF	64 (wrapping occurs)
0	1	1	0	1	X	A16 Super Access	$6800 0000 - 6FFF FFFF	64 (wrapping occurs)
0	1	1	1	0	0	A24 User Data / User Block	$7000 0000 - 73FF FFFF	32 (wrapping occurs)
0	1	1	1	0	1	A24 User Pgm / User Block	$7400 0000 - 77FF FFFF	32 (wrapping occurs)
0	1	1	1	1	0	A24 Super Data / Super Block	$7800 0000 - 7BFF FFFF	32 (wrapping occurs)
0	1	1	1	1	1	A24 Super Pgm / Super Block	$7C00 0000 - 7FFF FFFF	32 (wrapping occurs)

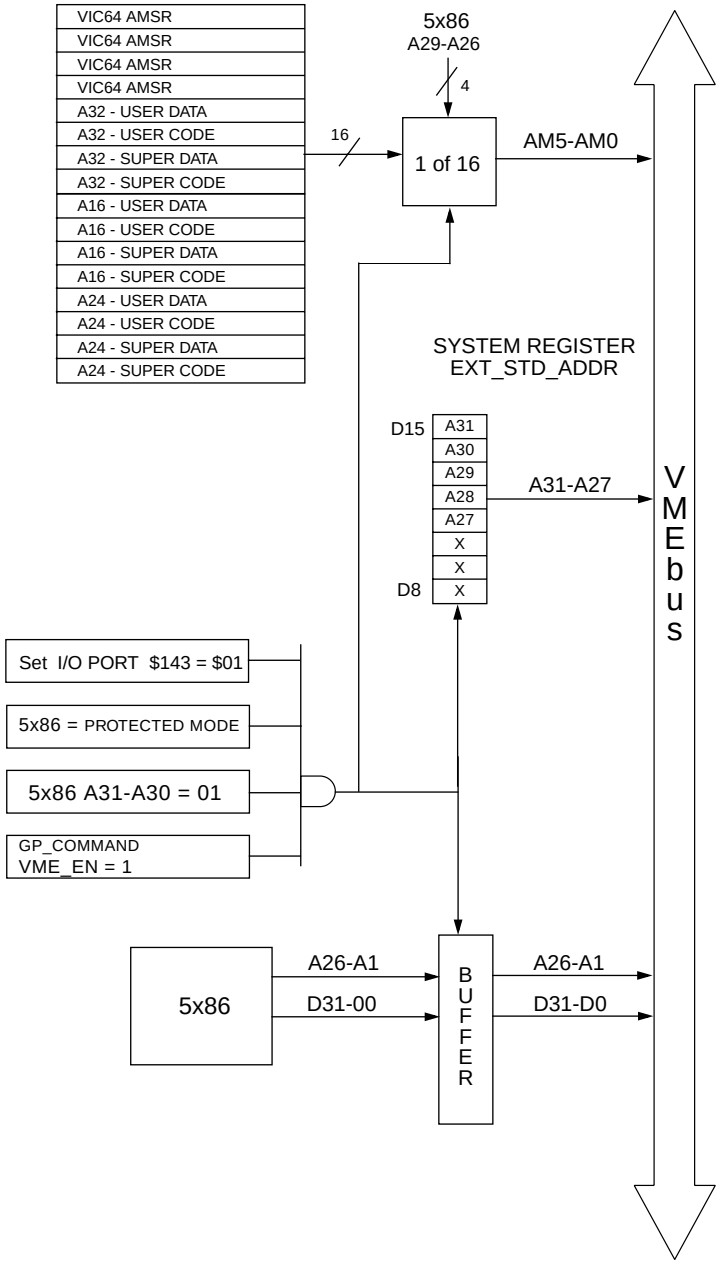


Figure 4-6 128 Mbyte Protected Mode VMEbus Access

After powerup, accessing VMEbus in Protected Mode for 128 Mbyte access requires at least the following steps:

1. Initialize the VIC Base Register. Program the VIC base address and enable access to the VIC by setting the VIC Enable bit.

2. Initialize the VIC Address Modifier Source Register. Clear bit 7 and initialize bits 5-0 with the desired VMEbus address modifier code. Note that the Address Modifier Source Register is *only* used when 5x86 address bits A29 and A28 are clear. Set the SysFail Mask bit in the VIC Mailbox Semaphore Register. Other VIC registers may be programmed depending on arbiter/requester options.

3. Initialize the General Purpose Command Register by setting the VME Enable bit. Other bits may be set depending on the application.

4. Initialize the Extended/Standard Address Register. This sets the upper VMEbus address bits as shown in Figure 4-5 on page 4-12, effectively controlling which segment of VMEbus addressing space appears in the Protected Mode VMEbus Window. For Protected Mode operation, bits 0-7 of the register are not used.

5. Perform an I/O write to port $143 of $01.

6. Perform memory access at an address between $4000 0000 and $7FFF FFFF, depending upon what address modifier is desired (see Table 4-2 on page 4-14). A byte, word, or longword access generates a VMEbus access.

SECTION 4 - VMEbus SLAVE OPERATION

The VMIVME-7586 has two resources that are accessible by VMEbus slave accesses: VIC interprocessor communication facilities and dual-port DRAM.

INTERPROCESSOR COMMUNICATIONS

The VIC contains interprocessor communication facilities which are accessible in the VMEbus Short I/O (A16) address space (see Table 4-8 on page 4-53 for a complete slave register map). The facilities are:

- Four Global Mailbox Switches (ICGS0-3)
- Four Module Mailbox Switches (ICMS0-3)
- Five Mailbox Registers (VIC registers ICR0-ICR4)
- One VIC Version/Revision Register (VIC register ICR5)
- One Reset/Halt Status Register (VIC register ICR6)
- One Mailbox Semaphore Register (VIC register ICR7)

Note that the terms “Interprocess Communication Registers (ICRs)” and “Mailbox Registers” are generally interchangeable. The switches mentioned are bits that can be set or cleared from the VMEbus. Depending on the VIC programming, the switches can be used to cause interrupts at the VMIVME-7586’s CPU. The Interprocessor Communications Switch Register, available only to the local processor, controls these switches locally. VMEbus masters control these switches from separate Set/Clear Switch Registers described below.

Like all VIC registers, these are 8-bit registers. The Mailbox Registers (ICR0-ICR5) are accessible by the VMIVME-7586’s 5x86 and by VMEbus masters and provide a “mailbox” resource. The Mailbox Semaphore Register (ICR7) provides semaphore bits associated with each of the five mailbox registers. The other ICRs allow a VMEbus master to gain status information about the VMIVME-7586. See page 4-105 for a complete description of all Interprocessor Communications Registers.

In addition to the ICRs, 16 registers addressable only from the VMEbus allow setting or clearing the global and module switches. The Set/Clear Switch Registers each have a specific function, setting or clearing a particular switch. The data written to these registers does not matter; the act of writing to the register performs the associated function.

The above VIC resources are accessible using VMEbus A16 Short I/O accesses (as well as from the VMIVME-7586 itself, except for the Set/Clear Switch Registers). Unlike all other interprocessor communications resources, the registers that set or clear global (ICGS) switches are only accessible from privileged VMEbus accesses. All other resources can be accessed using either privileged or nonprivileged address modifiers. The base Short I/O address is programmed by the local Slave A16 Address Compare Register. See Figure 4-7 on page 4-18 for a block diagram of the VMIVME-7586 slave VMEbus interface.

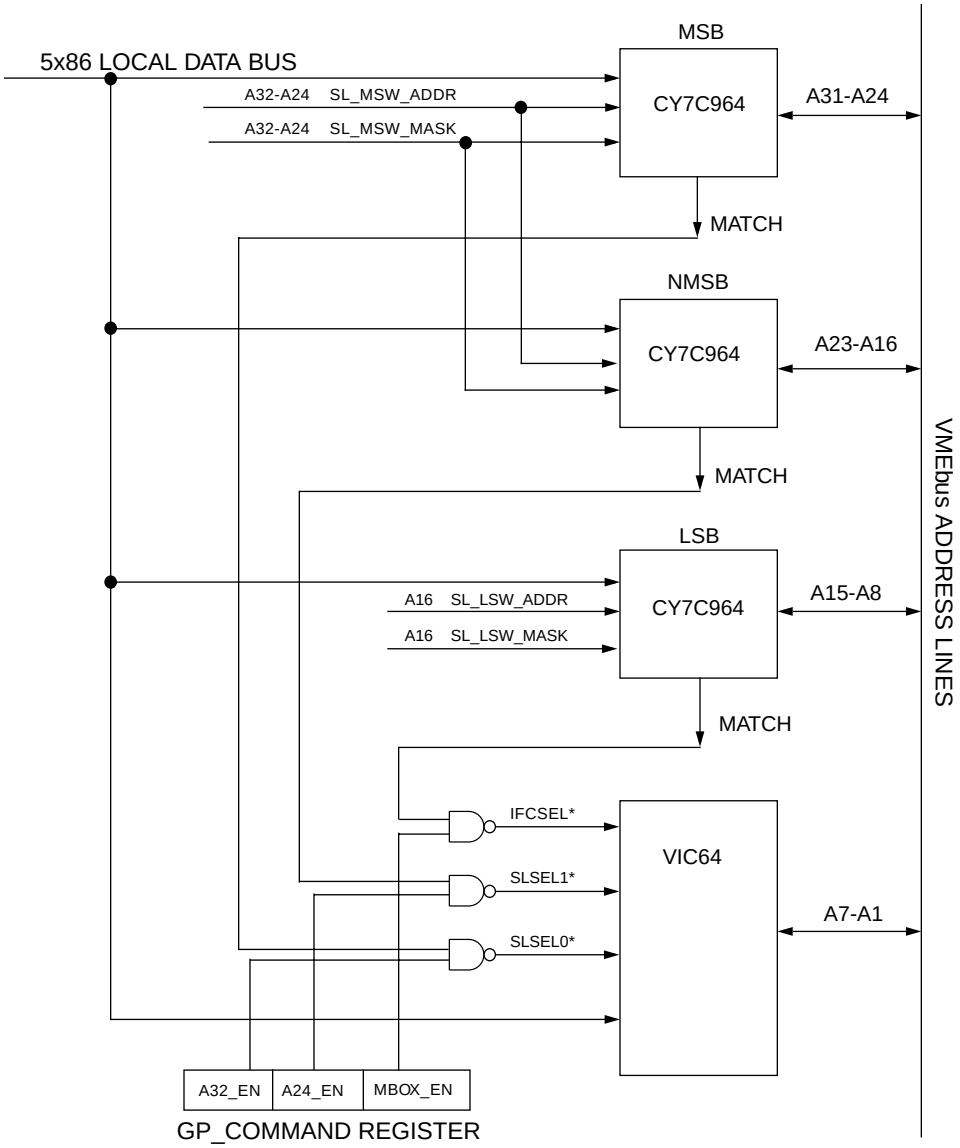


Figure 4-7 VMEbus Slave Interface

Each CY7C964 contains a Slave Address Compare register and a Slave Address Mask register. The compare register is used to program a base address to be used by VMEbus to access the resource. The mask register is used to define an address range that is used to access the resource.

As Figure 4-7 on page 4-18 shows, the LSB CY7C964 has a MATCH* output which is connected to the VIC ICFSEL* input. The MATCH* signal is activated when the VMEbus address matches the address defined by the LSB CY7C964 address compare and mask registers. When the LSB MATCH* signal is activated, the VIC further decodes VMEbus address bits A7-A1 to determine which interprocessor communication resource is being accessed. See the VIC064 User's Guide from Cypress Semiconductor for a detailed description.

The LSB CY7C964 registers correspond to the Slave A16 Address Compare and Slave A16 Address Mask system registers.

The upper register bits of the Slave A16 Address Compare and Mask registers correspond to VMEbus address bits A8-A15 (see the register description on page 4-63). Note that register bits D7-D0 are not used since the VIC decodes address bits A7-A1.

Initialize the Slave A16 Address Compare register to correspond to VMEbus address bits A15-A8. Use the Slave A16 Address Mask register to create an address range with the base address given by the Slave A16 Address Compare. VMEbus address bits defined in the Slave A16 Address Compare register become "don't cares" when the corresponding bit in the Slave A16 Address Mask register is programmed with a 1. See Figure 4-8 on page 4-20 for a graphic description of both registers.

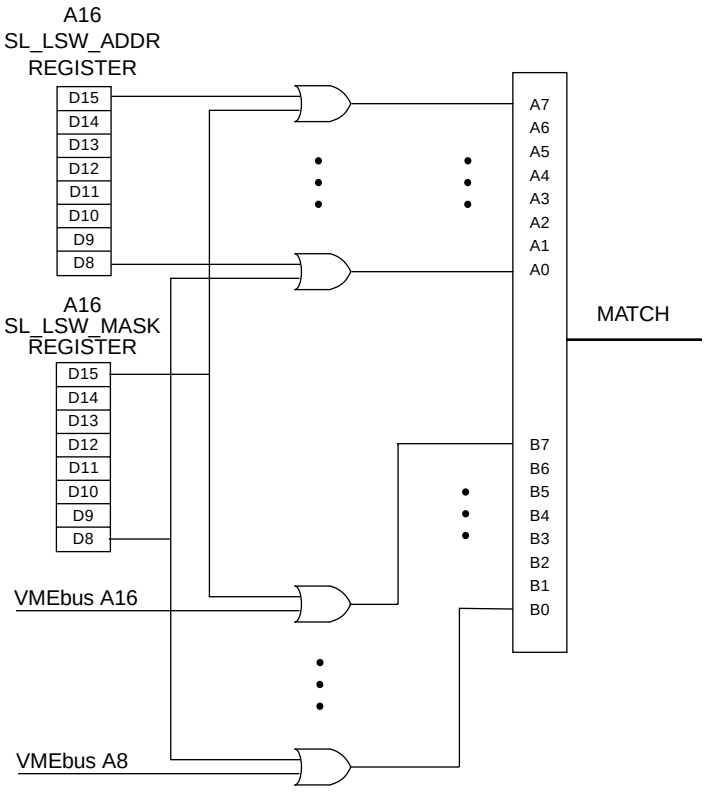


Figure 4-8 Slave Compare Operation

The General Purpose Command system register contains a Mailbox Enable bit that is used to ensure that slave accesses to the VIC are blocked until the Slave A16 Address Compare and Slave A16 Address Mask registers are properly initialized. Accesses to the VIC intercommunication resources prior to the Mailbox Enable bit being set will result in a VMEbus BERR* (provided that a bus-timeout module is enabled somewhere in the system).

Example: Program VIC ICRs to reside at VMEbus address $A500

1. Write 0xA500 data to Slave A16 Address Compare register.

2. Write 0x0 to Slave A16 Address Mask register.

3. Set Mailbox Enable bit in General Purpose Command system register.

Example: Program VIC ICRs to be accessible by VMEbus addresses between $4000 and $7F00.

1. Write 0x4000 data to Slave A16 Address Compare register.

2. Write 0x3F00 data to Slave A16 Address Mask register.

3. Set Mailbox Enable bit in the General Purpose Command register.

After powerup, the following procedure should be followed to allow a slave access to the VIC interprocessor communication facilities:

1. Initialize VIC Base Register. Program VIC base address and enable access to VIC by setting the VIC Enable bit.

2. Initialize VIC registers. Set the SysFail Mask bit in the VIC Mailbox Semaphore Register.

(Note that the following steps may not be needed depending on how the interprocessor communication facilities are to be used; that is, if ICF resources are not used to cause interrupts into the VMIVME-7586.)

3. Initialize the Slave A16 Address Compare and Slave A16 Address Mask Registers. Program the desired address and range. Note that the Slave A16 Address Compare must *always* be programmed prior to programming the Slave A16 Address Mask register. This is required since an access to the Slave A16 Address Compare register automatically clears the Slave A16 Address Mask register.

4. Initialize the General Purpose Command Register. Other bits may be set depending on the application.

DUAL-PORTED DRAM

All of the VMIVME-7586's DRAM (1 - 32 Mbyte, depending on the memory option ordered) is accessible from the VMEbus. More accurately, all VMIVME-7586 physical memory is accessible from the VMEbus. This includes all of the DRAM, but also includes all physical resources within the first megabyte of 5x86 address space (640 Kbyte of RAM, VGA RAM and BIOS, and the system BIOS). I/O ports are inaccessible through the slave interface. Being able to access the lower 640 Kbyte of RAM has some

utility; however, extreme caution should be used when accessing this region to prevent overwriting the operating system.

The VMIVME-7586 slave interface allows the dual-port DRAM slave address to be programmed in software. Also, incoming accesses may be remapped throughout the 32 Mbyte space. In addition, the VMEbus can access the DRAM using either A32 Extended addressing or A24 Standard addressing.

The A32 and A24 addresses are programmed using registers, which are contained in the MSB and NMSB Cypress CY7C964 ICs. See Figure 4-7 on page 4-18.

Each CY7C964 contains an address compare register and a mask register. The compare register is used to program a base address for the VMEbus to access the DRAM. The mask register is used to define an address range to access the DRAM.

As Figure 4-7 on page 4-18 shows, the MSB and NMSB CY7C964s have MATCH outputs which are connected to the VIC SLSEL0* and SLSEL1* inputs, respectively. The MATCH signals are activated when the VMEbus address matches the address defined by the MSB or NMSB CY7C964 address compare and mask registers.

The VIC contains slave select control registers (SS0CR0, SS0CR1, SS1CR0, SS0CR1), which allow the user to define VMEbus address modifier codes to be associated with the SLSEL0* and SLSEL1* inputs. When the VIC detects an active slave select input, it then checks to see if VMEbus signals AM5-AM0 match the address modifiers defined by the slave select control registers. If the AM5-AM0 signals match with the control register definitions, a slave access to the dual-port DRAM is performed.

The MSB CY7C964 is connected to VMEbus address bits A31-A24. Consequently, the VIC SLSEL0* input is reserved for VMEbus A32 addressing. Therefore, the VIC SS0CR0 register *must* be programmed for A32 addressing.

NOTE:

THE NMSB CY7C964 IS CONNECTED TO VMEbus ADDRESS BITS A23-A16. CONSEQUENTLY, THE VIC SLSEL1* INPUT IS RESERVED FOR VMEbus A24 ADDRESSING. THEREFORE, THE VIC SS1CR0 REGISTER *MUST* BE PROGRAMMED FOR A24 ADDRESSING.

The MSB and NMSB CY7C964 registers correspond to the Slave A32/A24 Address Compare and Slave A32/A24 Address Mask system registers.

Register bits of the Slave A32/A24 Address Compare and Slave A32/A24 Address Mask Registers correspond to VMEbus address bits A31-A16.

Note that D15-D8 of the Slave A32/A24 Address Compare and Slave A32/A24 Address Mask Registers correspond to registers contained in the MSB CY7C964. Bits D7-D0 of the registers correspond to registers contained in the NMSB CY7C964.

Initialize the Slave A32/A24 Address Compare register to set up A32 and/or A24 base addresses for accessing dual-port DRAM. Use the Slave A32/A24 Address Mask register to create an address range within each VMEbus address space (A32 and/or A24). VMEbus address bits defined in the Slave A32/A24 Address Compare register become "don't cares" when the corresponding bit in the Slave A32/A24 Address Mask register are set.

Example: Program the VMIVME-7586 slave interface to map 32 Mbyte of on-board DRAM to A32 VMEbus address $1200 0000.

1. Write 0x12XX data to the Slave A32/A24 Address Compare register.
2. Write 0x00XX to the Slave A32/A24 Address Mask register.
3. Initialize VIC SS0CR0 for A32 accesses.
4. Set the Slave A32 Enable bit in the General Purpose Command system register.

DRAM will now be accessible using VMEbus addresses $1200 0000 through $12FF FFFF.

Example: Program the VMIVME-7586 slave interface to map 1 Mbyte of on-board DRAM to A24 VMEbus address $10 0000.

1. Write 0xXX10 data to Slave A32/A24 Address Compare register.
2. Write 0xXX0F to Slave A32/A24 Address Mask register.
3. Initialize VIC SS1CR0 for A24 accesses.
4. Set A24_EN bit in General Purpose Command system register.

DRAM will now be accessible using VMEbus addresses $10 0000 through $1F FFFF.

The dual-port DRAM can be configured to be addressable using A32 accesses, A24 accesses, or both (by setting both the Slave A24 and Slave A32 Enable bits in the General Purpose Command Register). This may be useful for systems requiring the dual-port DRAM to be shared between A32 and A24 masters.

After powerup, the following procedure should be followed to allow a slave access to the dual-port DRAM:

1. Initialize the VIC Base system register. Program VIC base address and enable access to VIC by setting the VIC Enable bit.

2. Initialize VIC registers. Initialize the VIC SS0CR0 for A32 access and/or initialize SS1CR0 for A24 access. Set the SysFail Mask bit in the VIC Mailbox Semaphore Register.

3. Initialize the Slave A32/A24 Address Compare and Slave A32/A24 Address Mask Registers. Program the desired address and range.

   NOTE:

   THE SLAVE A32/A24 ADDRESS COMPARE MUST ALWAYS BE PROGRAMMED PRIOR TO THE SLAVE A32/A24 ADDRESS MASK REGISTER. THIS IS REQUIRED SINCE AN ACCESS TO THE SLAVE A32/A24 ADDRESS COMPARE REGISTER AUTOMATICALLY CLEARS THE SLAVE A32/A24 ADDRESS MASK REGISTER.

4. Initialize the General Purpose Command Register. Set the Slave A32 Enable and/or the Slave A32 Enable bits. Other bits may be set depending on the application.

SLAVE ACCESS MEMORY MAP

All of the VMIVME-7586 on-board physical memory may be accessed using VMEbus slave accesses. The VMIVME-7586 local addresses which map to the VMEbus cannot be accessed since they are off-board resources and would require the VMIVME-7586 to be both VMEbus master and VMEbus slave simultaneously. Table 4-3 on page 4-25 describes accessible and nonaccessible memory regions.

Table 4-3 Slave Access Memory Map

Local Address	Resource
$0200 0000 - $FFFF FFFF	NOT ACCESSIBLE *
$0010 0000 - $01FF FFFF	Extended memory DRAM (31 Mbyte maximum)**
$F0000 - $FFFFF	System BIOS
$E0000 - $EFFFF	NOT ACCESSIBLE (will produce BERR*)***
$D0000 - $DFFFF	Reserved for OS memory management (EMS)
$C8000 - $CFFFF	Reserved for Flash or Ethernet mezzanine options
$C0000 - $C7FFF	VGA BIOS
$A0000 - $B0000	VGA display DRAM
$00000 - $9FFFF	640 Kbyte of DRAM

* This region of the VMIVME-7586 local address space cannot be accessed from the VMEbus. A31-A25 of the VMIVME-7586 local address bus are forced to 0 when a VMEbus slave access occurs.

** The amount of extended memory DRAM is VMIVME-7586 option dependent.

*** Access to this region will not return a VMEbus DTACK*. A BERR* will be produced if a VMEbus timeout module is enabled in the system.

In general, applications that perform slave accesses to VMIVME-7586 memory will only use the 640 Kbyte of DRAM located at the VMIVME-7586 address $00000 and the extended memory DRAM, which begins at the VMIVME-7586 address $100000. However, the VMIVME-7586 slave interface does allow access to other local VMIVME-7586 resources as shown in Table 4-3. All of these resources should be used with caution to prevent overwriting the OS.

ADDRESS REMAPPING

One powerful feature of the VMIVME-7586 is the address window remapping and size control for slave VME accesses. Remapping gives the

programmer the capability of working around the dynamics of PC based compilers where memory usage may be unpredictable.

The code developed on a secondary VMEbus board could, for instance, use a 64 Kbyte range starting at 1 Mbyte in the VMEbus address space for shared array storage in the VMIVME-7586. In this particular case, if a high memory manager was used such as HIGHMEM.SYS, then system drivers maybe loaded in the VMIVME-7586's address space starting at 1 Mbyte which is in conflict with the VMEbus space. By programming the registers that control the address remapping and sizing, the user can shift the address of the VMEbus access from 1 MByte to, for example, 8 MByte on the 5x86 local bus thereby avoiding the memory conflict. This example is illustrated in Figure 4-9.

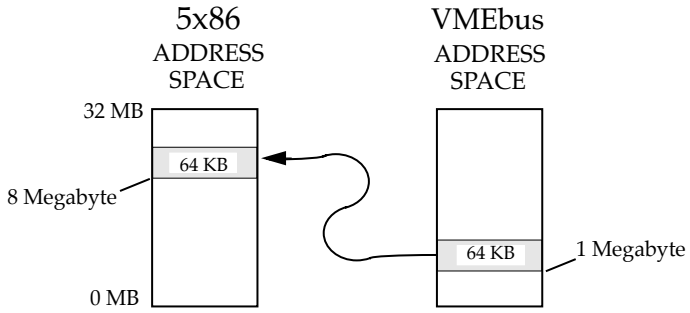


Figure 4-9 Remap of VMEbus Space to 5x86 Space

The sizing functionality provides the programmer with the ability to limit the space the VMEbus can access. If, for instance, the size is set to 64 Kbyte and the remap address is set to 8 Mbyte as previously described, then accesses will only transpire in the 64 Kbyte region starting at 8 Mbyte. Any accesses outside the 64 Kbyte window will wrap around back into the 64 Kbyte region starting at 8 Mbyte.

Figure 4-10 on page 4-27 depicts the slave remap circuit overview. The VMEbus addresses A15 through A0 are buffered to the 5x86 local address bus as shown at the top of the figure. The remapping and sizing of the lower 16 bits of the address bus are not supported; therefore, the minimum remapping resolution is 64 Kbyte. The lower half of the figure shows a simplified representation of the remapping and sizing circuit. The only bits that are affected are A24 through A16 on the VME address bus and the 5x86 local address bus. (Note, that the upper address lines, A31 through A24, are compared in the VIC circuitry to identify a valid access.) The

lower part of the figure depicts the VME address bus being gated to the 5x86 local address bus in the case that the corresponding window size register bit is set to '1'. If the corresponding window size register bit is set to '0' then the corresponding remap address register bit is gated to the 5x86 local bus.

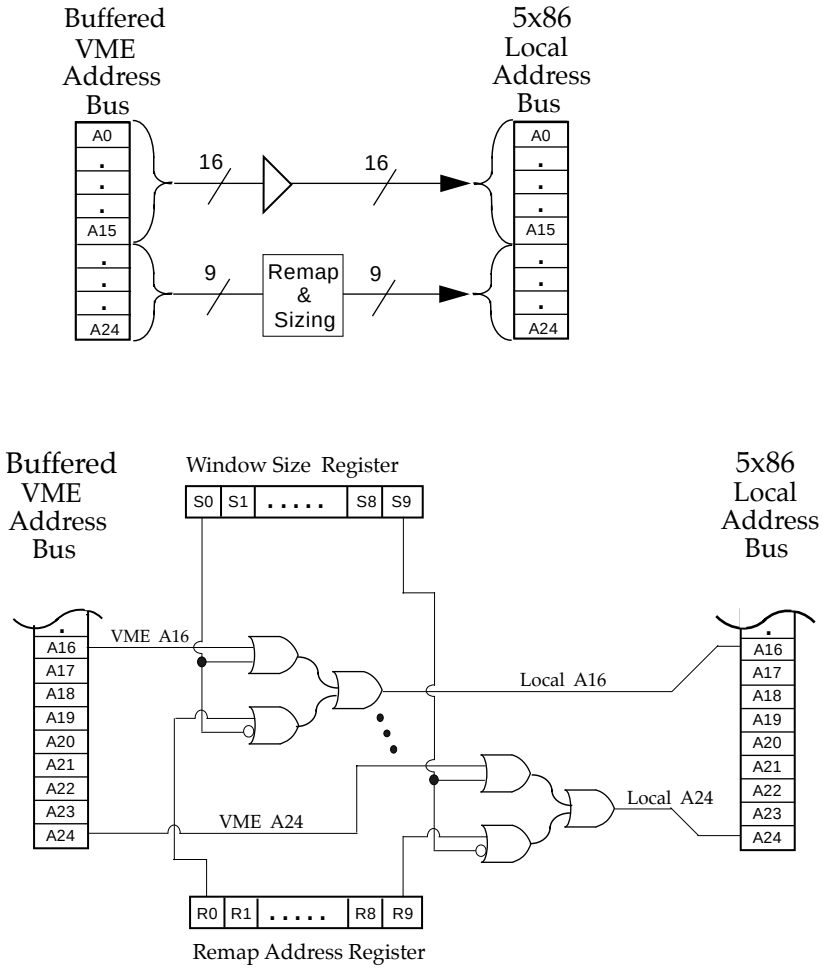


Figure 4-10 Slave Remap Circuit Overview

NOTE:

ADDRESSES MAY ONLY BE REMAPPED ON 64 KBYTE BOUNDARIES DUE TO THE HARDWARE IMPLEMENTATION AS DEPICTED IN FIGURE 4-10.

Some of the implications of the system design include: first, to disable size/remap function store $1FF into the window size register. This will gate the VME address bus directly to the 5x86 local bus. Second, the window size is a minimum of 64 Kbyte, incrementing to 128 Kbyte, 256 Kbyte, 512 Kbyte up to 32 Mbyte. And lastly, the remap and size registers are 9 bits. To write or read these registers a 16 bit I/O access must be performed. The upper 7 bits should be discarded by masking.

The remapping and sizing are bypassed during a master BLT cycle. In configuring a master BLT cycle the programmer must program the VIC interface with the local address. Therefore, remapping must be disabled and is done so automatically. There is no need for the programmer to disable slave remapping and sizing for the master BLT cycle. In a slave BLT cycle the secondary VMEbus unit will store the appropriate address into the VMIVME 7586's VIC interface. This address will be remapped accordingly.

NOTE:

IF THE REGIONS MARKED "NOT ACCESSIBLE" ARE ACCESSED, THE VMEBUS TRANSACTION WILL NOT BE TERMINATED WITH DTACK*. A BERR* WILL RESULT IF A BUS TIMEOUT MODULE IS ENABLED IN THE SYSTEM (EITHER THE VMIVME-7586 BUS TIMEOUT MODULE, OR ANOTHER BUS TIMEOUT MODULE ELSEWHERE).

SYSTEM CONSIDERATIONS

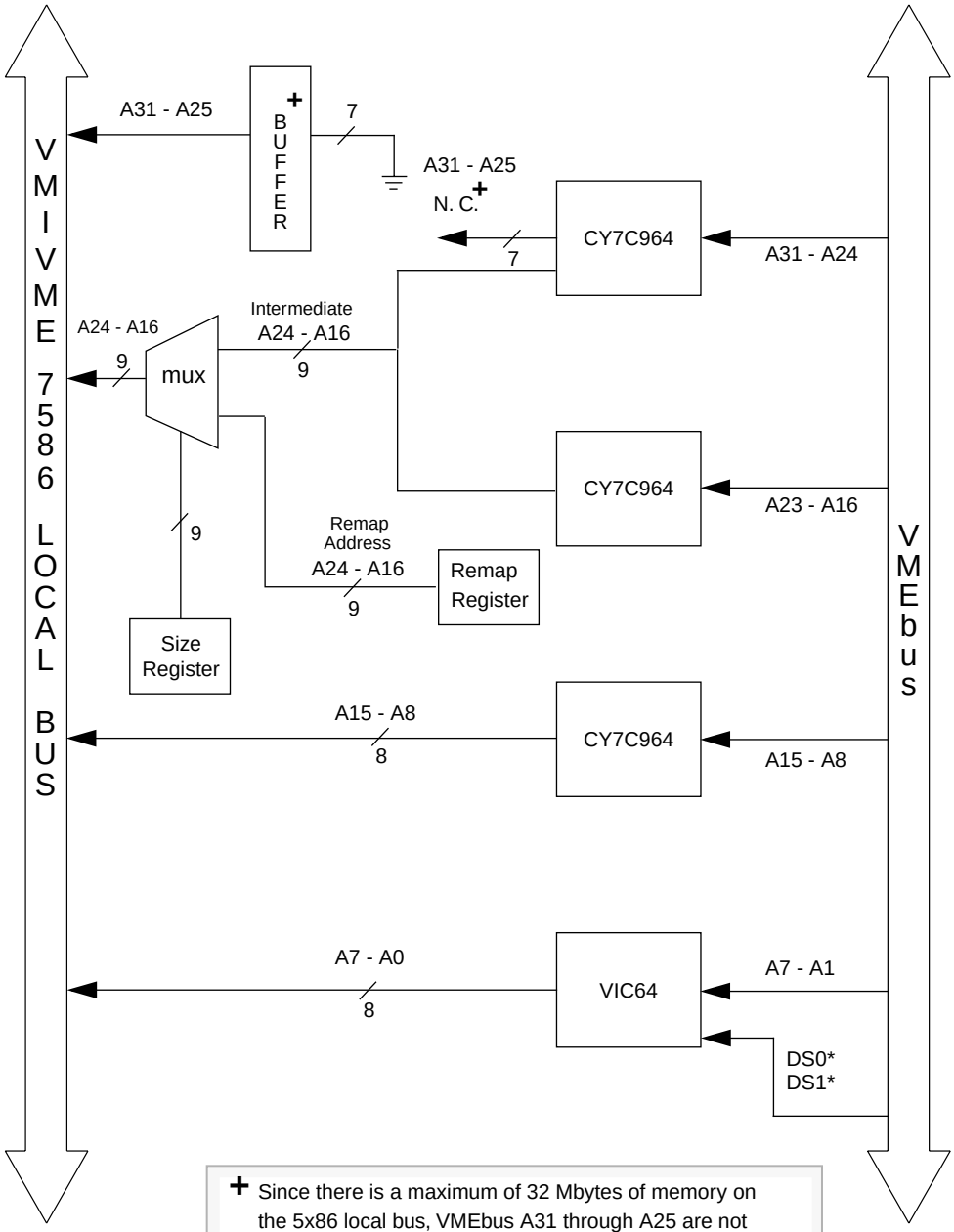
Some systems may require that a VMEbus master size the VMIVME-7586 on-board extended memory. In this case, it is important to know that VMEbus accesses to the VMIVME-7586 local address range $10 0000 – $FF FFFF will return a DTACK even if no RAM is installed there. The data read will be undefined.

When configured for A32 accesses, the VMIVME-7586 consumes at least 16 Mbyte of VMEbus A32 space (regardless of the state of the Slave A32/A24 Address Mask system register). Since the VMIVME-7586 only contains approximately 32 Mbyte of dual-ported DRAM (depending on the memory option), there is no reason to ever set bits 15-8 of the Slave A32/A24 Address Mask Register. In general, bits 15-8 of the Slave A32/A24 Address Mask Register should be cleared for A32 slave operation.

When configured for A24 accesses, the VMIVME-7586 consumes at least 64 Kbyte of VMEbus A24 space (if the Slave A32/A24 Address Mask system register is cleared). Setting bit 0 of the Slave A32/A24 Address Mask register increases the memory allocation to 128 Kbyte. Setting the Slave A32/A24 Address Mask Register to all ones allocates the entire 16 Mbyte of VMEbus A24 space to the VMIVME-7586.

It may be desirable to program the VMIVME-7586 to respond to user-defined address modifiers. This is accomplished by programming the VIC Address Modifier Source Register with the desired address modifier. The VIC slave select registers SS0CR0 and/or SS1CR0 can then be programmed to use the Address Modifier Source Register. See Chapter 6 in the VIC manual. Keep in mind though, that the VIC Address Modifier Source Register may not be available for this type of slave operation if the VMIVME-7586 is using this register for master operations (use of the Address Modifier Source Register is required for real-mode applications to access the VMEbus).

During a VMIVME-7586 slave access, VMEbus address bits connect to the VMIVME-7586 local address bits as shown in Figure 4-11. Note that local address bits A31-A25 are always driven to 0 during a slave access.



+ Since there is a maximum of 32 Mbytes of memory on the 5x86 local bus, VMEbus A31 through A25 are not passed onto the local bus. The local address lines A31 through A25 are pulled low during a slave access to avoid accessing memory outside the 32Mbyte range.

The CY7C964 uses the VMEbus address lines A31through A24 to generate a valid board decode which is not depicted in this figure.

Figure 4-11 Slave Addressing Detail

SECTION 5 - SYSTEM CONTROLLER FUNCTIONS

For VMEbus slot-1 applications, the VMIVME-7586 may be jumpered to perform system controller functions including:

- 16 MHz System Clock
- SYSRESET* Driver
- IACK Daisy-Chain Driver
- Programmable Arbiter (PRI, SGL, and RRS modes supported)
- Bus Error Timer

The VMIVME-7586 generates VMEbus SYSCLK and SYSRESET* signals and provides the IACK driver only if it is configured as a system controller. See Chapter 2 for information on jumpering the VMIVME-7586 as a system controller.

After powerup or a hard reset, the VIC automatically asserts SYSRESET* for the required minimum of 200 ms. SYSRESET* is *not* generated from the "soft boot" caused by a **<Ctrl+Alt+Del>** keyboard reset.

Enabling the system controller function also enables the VIC VMEbus arbiter. The VIC Release Control Register allows the arbiter to be configured for PRI, RRS, or BCLR operation. Single-level arbitration is obtained by programming the RCR for PRI and setting all requesters to level 3. In the RRS scheme, arbitration priority is assigned on a rotating basis; when the bus is granted to a requester on one bus request line, then the highest priority for the next arbitration rotates to the next bus request line. In systems containing many contending VMEbus masters, the use of RRS arbitration and fair requests is strongly recommended to prevent excessive bus latency to some VMEbus masters.

The VIC contains a programmable bus timeout timer. This timer can only be enabled when the VMIVME-7586 is configured as the system controller. The timeout value is programmed by the VIC TTR.

The VMIVME-7586 has a hardware bus timeout circuit that is independent of the VIC. This function is provided since the bus timeout in the VIC can only be enabled when the VMIVME-7586 is the system controller. In addition, the hardware timer offers a timeout value (10 µs) that is not offered by the VIC. This circuit is enabled by installing a jumper (see Chapter 2 for hardware configuration details).

The two closest choices of timeout offered by the VIC are 4 and 16 µs. Most applications require a timeout of longer than 4 and 16 µs borders on the DRAM refresh requirement.

Either bus error timer generates a BERR* signal when a nonexistent or nonresponding device is addressed on the VMEbus. For systems in which both VMIVME-7586 bus timers are 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-7586 operation is 10 µs. This value is dictated by the DRAM refresh time for the VMIVME-7586's memory. Timeouts beyond the 10 µs limit risk memory corruption due to lack of refresh.

It is recommended that the hardware bus timeout timer mentioned be used instead of the VIC bus timeout timer. This is desirable since the VMIVME-7586 local DRAM controller requires the local bus every 15.6 µs to refresh the DRAM. Therefore, to properly signal a bus error for master operations caused by the VMIVME-7586, a timeout of less than 15.6 µs is required in order to guarantee DRAM refresh timing.

SECTION 6 - VMEbus INTERRUPT HANDLING

All VMEbus related interrupts are generated by the VIC. The VIC contains 29 interrupt sources which are grouped into 19 priority categories. When multiple interrupts are pending, the VIC issues the interrupts using a fixed priority scheme. Note that not all of the VIC interrupt sources are used by the VMIVME-7586. The use of the interrupt source and the source priority is given below. Priority 1 is the highest priority source.

Table 4-4 Interrupt Priorities

Priority	Source	VMIVME-7586 Use
1	LIRQ7	NOT USED
2	Error Group	ACFAIL*, Write post fail, Arbitration timeout, SYSFAIL*
3	LIRQ6	VMIVME-7586 initiated BERR
4	LIRQ5	Reserved by VMIC
5	LIRQ4	Software Interrupt
6	LIRQ3	NOT USED
7	LIRQ2	Periodic Timer Interrupt

Table 4-4 Interrupt Priorities (Continued)

Priority	Source	VMIVME-7586 Use
8	LIRQ1	NOT USED
9	ICMS Group	ICMS Group
10	ICGS Group	ICGS Group
11	IRQ7	VMEbus IRQ7*
12	IRQ6	VMEbus IRQ6*
13	IRQ5	VMEbus IRQ5*
14	IRQ4	VMEbus IRQ4*
15	IRQ3	VMEbus IRQ3*
16	IRQ2	VMEbus IRQ2*
17	IRQ1	VMEbus IRQ1*
18	DMA complete	Interrupter DMA Complete
19	VMEbus Interrupter	Interrupt on VMEbus Interrupt Acknowledge

The VIC allows each interrupt source to be programmed for an interrupt level (7-1). The VIC does not award priority based on the programmed interrupt level, but awards priority based on a fixed priority as shown above. Allowing the interrupt level to be programmed simply allows the user to prioritize VIC interrupts relative to other on-board interrupt sources.

The VIC issues an interrupt level equal to the level programmed for the highest priority interrupt pending. For example, if the VMEbus IRQ2* source is programmed to interrupt on level 4, the VIC will issue a level 2 interrupt when both interrupt sources are pending at the same time.

All VIC interrupt sources have interrupt vectors or IDs associated with them. This vector or ID is supplied by the VIC during the interrupt acknowledge cycle. The VIC determines which interrupt level is being acknowledged and then issues the ID for highest priority pending interrupt that is programmed with that level.

The VMIVME-7586 allows the user to choose one of three PC/AT interrupt channels to be used for interfacing with the VIC interrupts: IRQ11, IRQ12, or the NMI. The user programs each source to interrupt the 5x86 on one of

these PC/AT interrupts by using interrupt levels. The interrupt levels correspond to the PC/AT interrupt channels according to Table 4-5.

Table 4-5 Interrupt Level Assignments

Level	PC/AT Interrupt
4	NMI
2	IRQ11
1	IRQ12

VIC interrupts are acknowledged by reading the ID register associated with the PC/AT interrupt source. Reading the ID register causes an interrupt acknowledge cycle to be initiated for the associated level. The returned 8-bit ID is then used to determine which of the 29 interrupt sources caused the interrupt.

Three system registers are used to acknowledge VIC interrupts: NMI Interrupt ID, IRQ11 Interrupt ID, and IRQ12 Interrupt ID. Each register is 8 bits wide and will return an interrupt vector for the highest pending interrupt that was programmed with the corresponding level. In the event that the highest pending interrupt is a VMEbus interrupt (IRQ7*-IRQ1*), the register read will initiate a VMEbus Interrupt Acknowledge cycle. The returned ID value will be the ID read during the interrupt acknowledge cycle.

WARNING

ALL ACTIVE VIC INTERRUPT SOURCES MUST BE PROGRAMMED TO INTERRUPT ON EITHER LEVEL 4, LEVEL 2, OR LEVEL 1. PROGRAMMING THE INTERRUPT LEVEL TO ANY OTHER VALUE WILL RESULT IN UNDEFINED BEHAVIOR.

The VIC is designed to be part of a Motorola 68000-type system. Consequently, the VIC will only DTACK and perform an interrupt acknowledge cycle if there is an interrupt pending that matches the level that is being acknowledged. The VIC to local bus interface (VLIC) relies on getting a DTACK from the VIC for synchronization with the 5x86 local bus. Failure to receive a VIC DTACK will lock the 5x86 bus.

WARNING

THE NMI INTERRUPT ID, IRQ11 INTERRUPT ID, AND IRQ12 INTERRUPT ID REGISTERS SHOULD ONLY BE READ WHEN THE CORRESPONDING VIC INTERRUPT LEVEL IS PENDING. THIS IS GUARANTEED BY:

1. **Programming the VIC to produce interrupts using only levels 4, 2, or 1.**
2. **Only reading the ID registers within the corresponding interrupt service routine (NMI ISR, IRQ11 ISR, or IRQ12 ISR).**

The VMIVME-7586 IRQ11 and IRQ12 interrupts are dedicated for use with the VIC and therefore are not shared by any other on-board resource. In contrast, the VMIVME-7586 NMI interrupt is shared by the VIC and other on-board circuitry.

The PC/AT architecture defines two possible sources for the NMI: on-board parity errors and off-board parity errors. The off-board parity error was designed to allow ISA boards which contained memory to flag the motherboard processor in the event of a parity error. This is accomplished using the IOCHCK* signal (I/O channel check). Since the VMIVME-7586 does not use parity checking for off-board resources, the IOCHCK* signal is used to provide another on-board NMI source: the VIC.

User software can determine if the NMI was caused by an on-board memory parity error or by the IOCHCK* (VIC) by reading 2 bits defined in an I/O address at $61. Bit 7 is used for on-board parity status, bit 6 is used for the VIC. An NMI is caused if either or both bits are set. It is recommended that priority be given to the on-board parity error if both bits are set at the same time.

WARNING

ONLY READ THE NMI_ID REGISTER FROM WITHIN THE NMI ISR. CHECK TO SEE THAT BIT 6 OF THE I/O PORT AT $61 IS SET PRIOR TO READING THE NMI_ID REGISTER.

SOFTWARE INTERRUPTS

The VMIVME-7586 5x86 CPU can initiate a software-controlled interrupt on either the IRQ11, IRQ12, or NMI interrupts. This is accomplished by using the local interrupt LIRQ4 channel of the VIC.

The VIC LIRQ4 input is connected to GND. A software-generated interrupt can be produced by programming the VIC LICR4 register. Initialize the register for level-sensitive inputs, inputs active low, autovectoring enabled. Initialize the IPL field to level 4, 2, or 1 as desired.

The interrupt is initiated by clearing bit 7. The interrupt is cleared by reading the ID register within the appropriate interrupt service routine.

INTERRUPT ON BERR*

The VMIVME-7586 allows the 5x86 to receive an interrupt if the VMIVME-7586 causes a VMEbus BERR*. This is particularly important feature given that the 5x86 CPU does not directly support bus error conditions.

Consequently, when the 5x86 accesses the VMEbus, the cycle will terminate when either a DTACK* or BERR* is activated. If a VMEbus access is terminated by BERR*, user software will not know unless the event triggers an interrupt. It is recommended that the NMI interrupt be used to signify the BERR* cycle termination.

The LIRQ6 input triggers an interrupt when the VMEbus BERR* is activated as a result of a VMIVME-7586 initiated transfer. Bus errors caused by some other device on the VMEbus will not cause an interrupt on the VMIVME-7586.

The VIC register LICR6 is used to program the LIRQ6 input as an interrupt. To use the channel as described, program the LICR6 register for edge sensitive, falling edge with autovectors enabled. Program the VIC LIVBR register with the desired interrupt vector information.

PERIODIC TIMER INTERRUPT

The VMIVME-7586 provides a timer that can be used for producing periodic interrupts. This timer is independent of timers that are part of the PC/AT architecture. The timer is implemented using the VIC.

The timer is programmed using the VIC SS0CR0 register. The timer can be programmed for 50, 100, or 1000 Hz operation. The timer can be programmed to produce an interrupt using the LIRQ2 channel of the VIC.

INTERRUPT PROCESSING

The VIC produces interrupts that are level sensitive as required by the Motorola 68000 interrupt architecture. In contrast, the PC/AT architecture defines non-NMI interrupts to be edge-sensitive. To ensure that the PC/AT hardware properly recognizes VIC interrupts, VIC level interrupts are converted to edges by performing a write access to the Rearm Interrupt Register. This write access is only required within the IRQ11 and IRQ12 interrupt service routines.

Other hardware read/write cycles are necessary to enable and rearm interrupts as required by the PC/AT hardware. These operations are shown for the IRQ11 and IRQ12 interrupts in Figure 4-12 on page 4-38. Operation for the NMI interrupt is shown in Figure 4-13 on page 4-39.

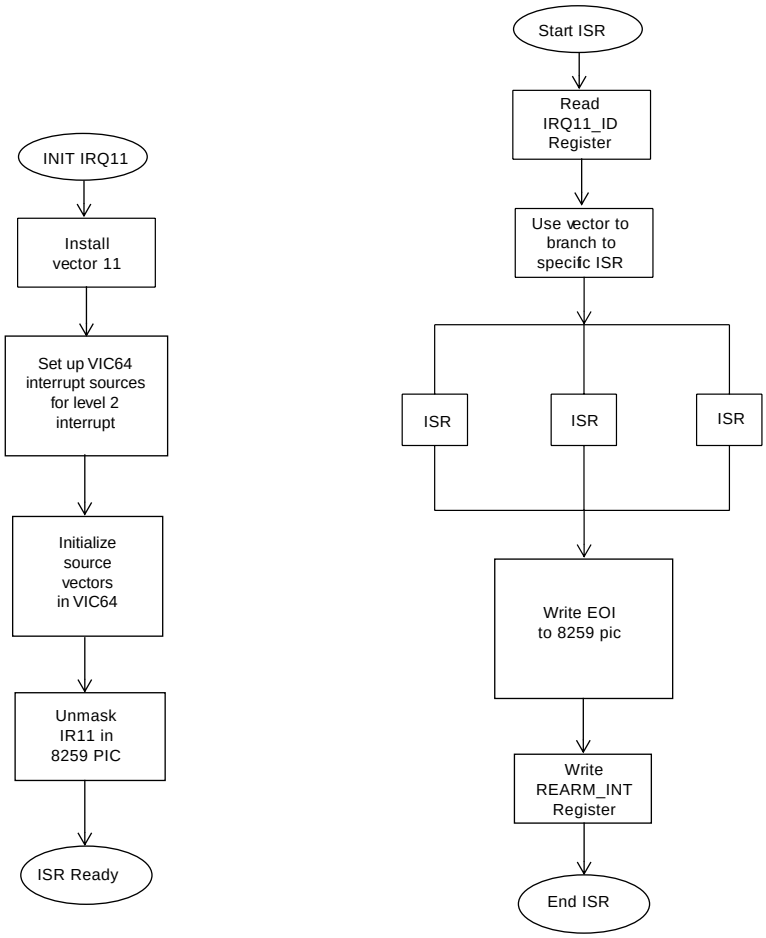


Figure 4-12 Flowchart for Non-NMI Interrupt Processing for IRQ11

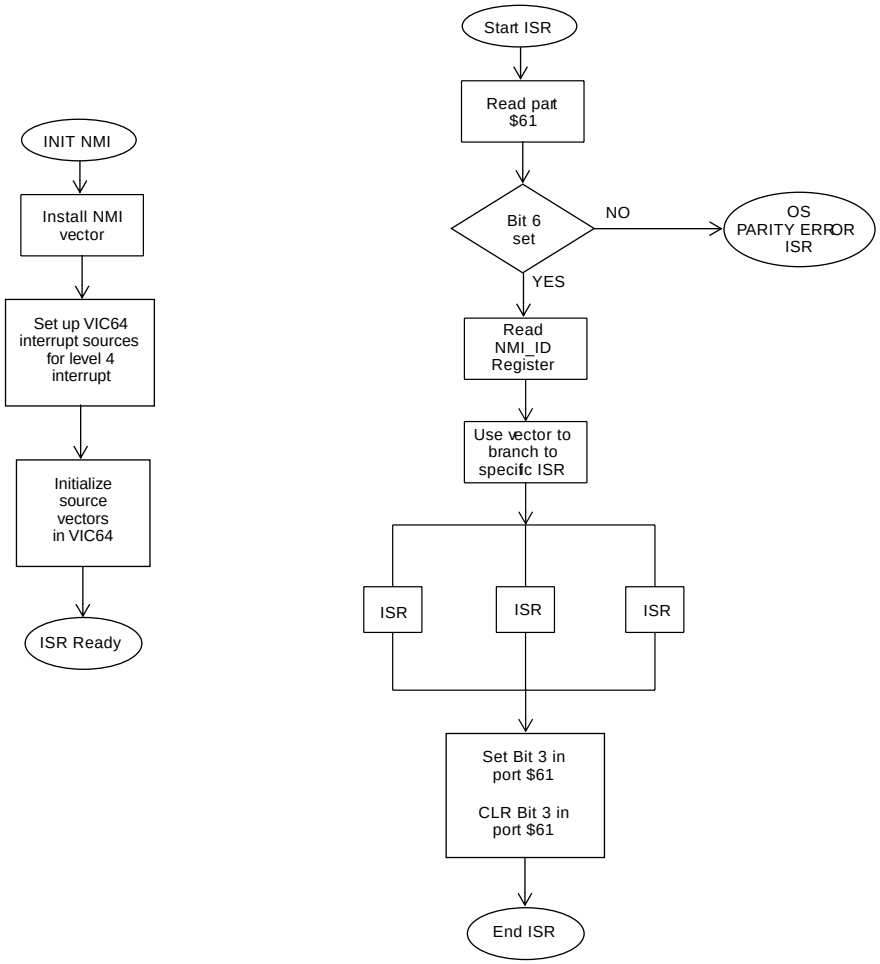


Figure 4-13 Flowchart for NMI Interrupt Processing

SECTION 7 - VMEbus INTERRUPTER

The VMIVME-7586 allows the user to activate VMEbus interrupt signals IRQ7*-IRQ1*. This is accomplished by writing to the VIC VIRSR. The VIC contains a vector register associated with each IRQ signal. When the issued IRQ is acknowledged by an off-board interrupt handler, the VIC supplies the vector associated with that interrupt provided the VMIVME-7586 IACKIN* signal is active.

The VMIVME-7586 can be programmed to interrupt the CPU when a VMIVME-7586 issued VMEbus interrupt has been acknowledged. This is

accomplished by programming the VIC VIICR and the EGIVBR. The VIICR is used to program the desired interrupt level. The EGIVBR is used to program the interrupt vector that will be supplied to the 5x86 when the appropriate interrupt acknowledge register is read.

When an interrupt is received that indicates acknowledgment of the VMIVME-7586 issued VMEbus interrupt, the VIC VIRSR is read to identify which interrupt has been acknowledged.

SECTION 8 - VMEbus REQUESTER

Both the master and interrupt handler functions require the use of the VMEbus. Both request this access using the VMIVME-7586 on-board requester which is implemented by the VIC.

The VIC allows the user to program which VMEbus bus request signal (BR3*-BR0*) will be used to request the bus. This is programmed in the VIC ARCR. The default setting of the register after system reset is BR3*.

The requester release mode is also programmable. Using registers in the VIC, the user can program the request mode as:

ROR Release-on-Request

RWD Release-When-Done

ROC Release-on-Bus-Clear

BCAP Bus-Capture-and-Hold

The desired mode is programmed in the VIC RCR. The default mode of this register is the ROR mode.

SECTION 9 - READ-MODIFY-WRITE CYCLES

The VMIVME-7586 allows both master and slave VMEbus read-modify-write (RMW) cycles to be performed.

Slave VMEbus RMW cycles require no special programming of the VMIVME-7586. The VIC will keep the local bus on behalf of the VMEbus slave access until the read and write portion of the cycle is completed. The 5x86 is not allowed to interrupt the local bus read-modify-write cycle.

Use of Master VMEbus RMW cycles require that the VIC be programmed to support the RMW operation. This is programmed by using the VIC ICR. Bit 6 of this register is set if VMEbus RMW cycles are desired.

A Master VMEbus RMW cycle is accomplished by instructing the 5x86 to access the VMEbus using a "LOCKED" instruction. The 5x86 instruction set provides "test and set" type instructions. When used with the "lock" prefix, these instructions explicitly "lock" the local bus by activating the 5x86 LOCK* bus signal.

The 5x86 LOCK* signal is connected to the RMC* input of the VIC. When the VIC detects a master VMEbus request by the VMIVME-7586 and the activation of the RMC* input, a VMEbus RMW cycle is performed. Note that this "locked" instruction only performs a VMEbus RMW cycle if the VIC ICR bit 6 was previously set.

The following is an example C language function that can be used to perform master VMEbus RMW cycles:

```
/* Test and set bit 7 in specified VMEbus location. Function
returns a nonzero byte if the bit was a 0 on the read portion
of the RMW cycle
*/

unsigned char tas_bit7_in_byte( unsigned int, unsigned int );
/* prototype */

unsigned char tas_bit7_in_byte( unsigned int sgm, unsigned int off )
{
unsigned char by;
       asm
       {
              push  ds
              push  di
              mov   ds,sgm
              mov   di,off
              mov   by,0
              lock  bts   [ds:di],7
              setnc by
              pop   di
              pop   ds
       }
       return by;
}
```

To perform a RMW master access while in real mode, the C function is called as follows:

```
tas_bit7_in_byte( 0xE000, vme_offset );
```

Where the VMEbus A15-A0 offset address is assigned to the variable `vme_offset` prior to the call, and VMEbus A31-A16 and VMEbus AM5-AM0 were previously set by initializing the Extended/Standard Address Register and the VIC Address Modifier Source Register.

The `bts` instruction will perform a test and set on bit 7 of the byte at the specified VMEbus address. The lock prefix will force the 5x86 LOCK* signal to be asserted which will indicate to the VIC that the cycle is to be a RMW cycle.

The function returns a nonzero value if the bit was clear prior to the set.

Local bus RMW cycles performed by the 5x86 cannot be interrupted by the VMIVME-7586 VMEbus slave access. Therefore, it is desirable to set up a multiprocessing semaphore that is located in the VMIVME-7586 dual-port RAM. The semaphore should be accessed by the 5x86 using locked instructions. The semaphore may be accessed by another VMEbus master using VMEbus RMW cycles. Using this method avoids programming the VIC ICR for RMW cycles.

SECTION 10 - BLOCK TRANSFERS

The VMIVME-7586 supports VMEbus Block Transfers (BLTs) in both master and slave modes. In addition, a VMIVME-7586 with the VIC64 supports 64-bit block transfers. All BLT modes are described in this section.

DRAM REFRESH CONSIDERATIONS

During BLTs, the VIC chip becomes a bus master on the VMIVME-7586 local bus. Consequently, the BLT operation must ensure the VMIVME-7586 DRAM controller gains control of the VMIVME-7586 local bus at least every 15.6 µs in order to perform DRAM refresh or RAM corruption results.

Two modes of BLTs are allowed on the VMIVME-7586: single cycle local DMA mode and accelerated local DMA mode. Single cycle mode causes all local bus cycles used by the VIC to be single cycle. Using this mode guarantees DRAM refreshes will operate properly, but with the penalty of

slowing down VMEbus transfers. Accelerated local DMA mode maximizes the VMEbus transfer speed, but requires that the VMEbus BLT burst be limited to less than 15.6 µs.

For master BLT accesses, limiting the BLT burst can be ensured by programming the VIC Release Control Register (RCR). Bits 5-0 of the RCR are used to program the block transfer burst length. In general, master BLT accesses to a BLT compatible slave device will be 250-500 nanoseconds in duration. Therefore, to ensure release of the local bus by the VIC every 15.6 µs, bits 5-0 of the RCR should be set to 30 (0x1E) or less.

The VIC Block Transfer Control Register (BTCR) allows the user to specify the amount of time the VIC will wait between BLT bursts. To ensure DRAM refreshes, it is only required that the interleave time be at least 250 ns. This is accomplished by programming bits 3-0 of the BTCR with a value equal to 1 or more.

For slave BLT accesses to the VMIVME-7586, the VIC becomes the VMIVME-7586 local bus master. To ensure that the DRAM controller gains access to the local bus every 15.6 µs, the VIC must be programmed for single cycle mode or the master performing the BLT must use burst times that are less than 15.6 µs.

MASTER BLT OPERATION

The VMIVME-7586 performs BLT using the VIC local DMA mode. The "MOVEM" mode of VIC BLT is not supported on the VMIVME-7586. The VIC allows local DMA mode to be operated in accelerated mode or single cycle mode.

To program the VIC to emulate single cycle during master BLTs, set bits 1 and 0 of the Slave Select 0 Control Register 0 (SS0CR0). The Slave Select 0 Control Register 1 (SS0CR1) should be programmed to 0x33. The Local Bus Timing Register (LBTR) should be programmed to 0xFF.

To program the VIC for accelerated transfers during master BLTs, program bits 1 and 0 of the SS0CR0 to 10 (0x0A). The SS0CR1 register should be programmed to 0x00 to maximize local bus performance. The LBTR should be programmed to 0xFF.

The following sequence may be used to initiate master BLTs:

1. Program SS0CR0, SS0CR1, and LBTR for single cycle or accelerated local DMA BLT.
2. Program Block Transfer Definition Register (BTDR) to 0x0F.
3. Program Address Modifier Source Register to type address modifier corresponding to type of block transfer (standard supervisory, standard nonprivileged, extended supervisory, extended nonprivileged).
4. Program RCR and BTCR if using accelerated BLTs.
5. Program number of bytes to be transferred into BTLR0 and BTLR1.
6. Program A31-A16 of BLT slave address into EXT_STD_ADDR register.
7. Set bit 6 of BTCR for local DMA mode BLT and bit 4 of BTCR for data direction (read or write).
8. Perform 32-bit write to the Real Mode VMEbus Window. A31-A16 of slave BLT address is contained in EXT_STD_ADDR. A15-A0 of the slave BLT address is the offset address used during the Real Mode VMEbus Window access. The local bus address is the 32-bit value written to the Real Mode VMEbus Window.

NOTE:

D32 BLT TRANSFER REQUIRE LOCAL AND VME ADDRESSES TO BE LONGWORD ALIGNED. LIKEWISE, MBLT D64 TRANSFERS REQUIRE DOUBLE LONGWORD ALIGNMENT.

Furthermore, if the programmer desires the BLT transfer to be in little-endian format, the M_BIG_ENDIAN bit must be clear prior to performing step 8. If little-endian format is desired, the 32-bit value written to the Real Mode VMEbus Window must be byte swapped.

Example: Move block of data at local address 0x00158840 to VME address 0x01000000 in little-endian format

Write A31-A16 of the target VMEbus address into EXT_STD_ADDR register. Then write 0x40881500 to 0xE0000.

To transfer the same data in big-endian format, set the M_BIG_ENDIAN bit and then write 0x00158840 to 0xE0000.

SLAVE BLT OPERATION

As described in Section 4, the slave select 0 VIC input is used for A32 slave accesses and slave select 1 is used for A24 slave accesses. Prior to the slave BLT the VIC slave select registers must be programmed to allow for A32 or A24 BLT slave accesses. As described above, the registers should be programmed for single cycle or accelerated transfers.

If single cycles are desired, program bits 1 and 0 to 01 in the SS0CR0 or SS1CR0. Also, program the corresponding SS0CR1 or SS1CR1 register to 0x33 or greater. Program the LBTR to 0xFF.

If accelerated BLTs are desired, program bits 1 and 0 to 10 in the SS0CR0 or SS1CR0 registers. Also, program the corresponding SS0CR1 or SS1CR1 register to 0x0. Program the LBTR to 0xFF.

SECTION 11 - VME64 FUNCTIONS

A VMIVME-7586 with the VME64 installed may transfer 64 bits at a time using VME64 MBLT block transfers.

MASTER VME64 OPERATION

As defined in Section 13, the BTDR has additional bit fields that the VIC64 added. When initializing the VIC registers, bit 4 should be set when the BTDR is initialized per the sequence defined above.

Also, the VIC64 provides an extra block transfer register BTLR2. This should be initialized according to the Master BLT initialization sequence on page 4-43.

Also, note that the RCR is redefined for D64 operation such that burst length is determined by multiplying the register value by 4.

SLAVE VME64 OPERATION

In addition to initialization performed for standard BLTs, set bit 6 in the BTDR to enable slave block transfers.

SECTION 12 - 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 4-6 depicts the Even/Odd Byte assignments to the VMEbus data lines.

Table 4-6 VMEbus Byte Assignment to the Data Lines

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

It is important to note the major byte-ordering differences between the VMEbus and the Intel 5x86 CPU. In addition, communication between

Motorola compatible 680x0 VMEbus modules and the VMIVME-7586 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-7586 uses an Intel 5x86 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 4-14.

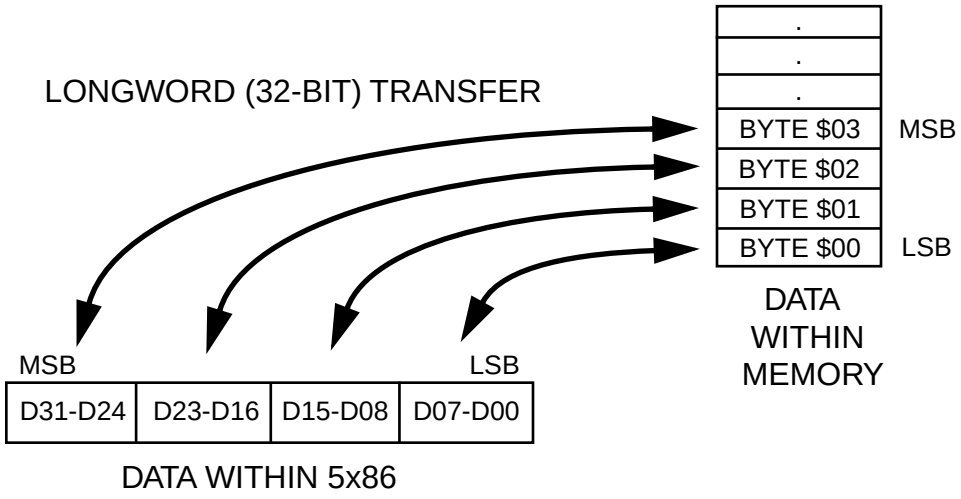


Figure 4-14 Byte Relationships Using the Little-Endian 5x86

Note that in little-endian processors like the 5x86, 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 5x86 in Figure 4-14 on page 4-47 with the same longword transfer using a big-endian processor like the Motorola 68040 in Figure 4-15.

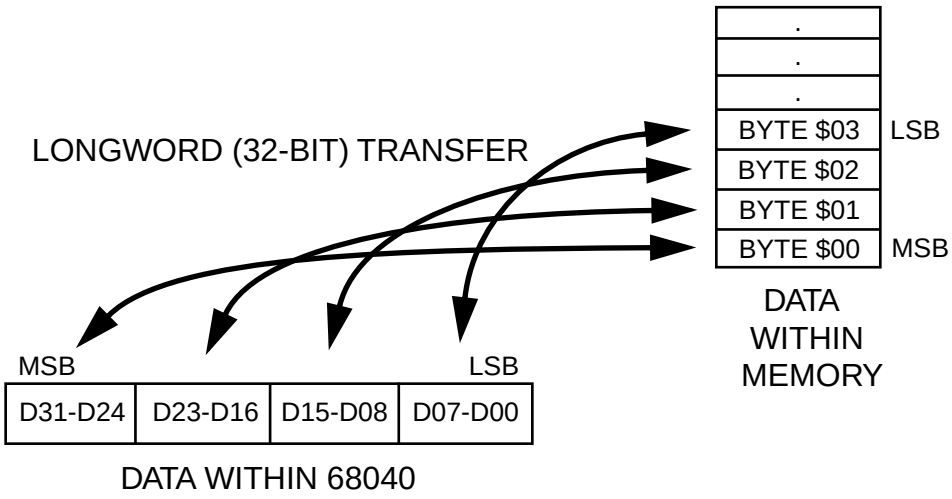


Figure 4-15 Byte Relationships Using the Big-Endian 68040

Note that the big-endian 68040 handles the same longword transfer in a completely different manner than the little-endian 5x86. 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 4-6 on page 4-46, 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 4-15 on page 4-48.

If a little-endian 5x86 were to have its data bus directly connected to the VMEbus (that is, 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 5x86 (see Figure 4-14 on page 4-47). This poses no problem if the 32-bit value written is always read back using a similar longword transfer (that is, all four bytes at once), since the swapped data gets swapped again and appears to the 5x86 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 5x86 to the VMEbus with its byte lanes crossed. For example, the 5x86 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-7586 to facilitate different kinds of byte swapping for varying circumstances.

VMIVME-7586 Byte-Swapping Hardware

The VMIVME-7586 employs three-way byte swap buffers to accommodate the byte-ordering inconsistencies between the 5x86 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 4-16 on page 4-50 diagrams the function of these swap buffers.

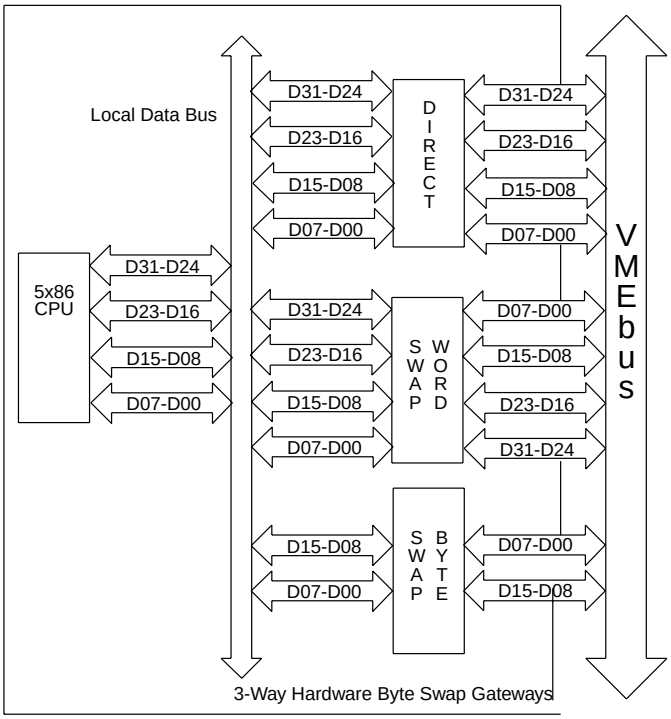


Figure 4-16 5x86-to-VMEbus Data Byte Lanes

The byte-swapping hardware is affected by two factors: the Big-Endian bit in the General Purpose Command Register, and the size of the transfer being carried out. Table 4-7 details this relationship.

Table 4-7 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 5x86 CPU to access the VMEbus much the same way a big-endian processor would. Two other facts should also be noted from Table 4-7: 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-7586 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.

Master/Slave Byte Swapping

On the VMIVME-7586, byte ordering can be programmed for either big endian or little endian in both directions (that is, whether the VMIVME-7586 is a VMEbus master or a VMEbus slave). Two bits in the General Purpose Command Register control the byte-ordering modes for master and slave accesses: Master Big-Endian, and Slave Big-Endian. The Master Big-Endian bit configures the byte-ordering mode when the VMIVME-7586 is acting as VMEbus master. The Slave Big-Endian bit configures the byte-ordering mode when the VMIVME-7586 is acting as a VMEbus slave.

SECTION 13 - VMIVME-7586 REGISTERS

REGISTER MAPS

The VMIVME-7586 has three groups of custom registers:

- System Registers (12)
- Interrupt Acknowledge Registers (3)
- VIC Registers (58)

All registers are in I/O addressing space, but only the System Registers have a fixed location: I/O $140 through I/O $152. The location of the Interrupt Acknowledge and VIC Registers are determined by the value in the VIC Base Register, a System Register at the fixed address of I/O $141.

Table 4-8 on page 4-53 shows the name, location, access mode, and bit map for each register. Similarly, Table 4-8 on page 4-53 shows the VIC registers. Subsequent sections discuss each register and its bit map in detail.

Table 4-8 System Register Map

REGISTER NAME	PAGE	MNEMONIC	ACCESS	I/O ADR	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
General-Purpose Command	4-58	GP_COMMAND	Byte Write	$140									Status LED	Master Big-Endian	Slave Big-Endian	Slave A32 Enable	Mailbox Enable	BLT Word Enable	VME Enable	Slave A24 Enable
Product ID	4-61	BRD_ID	Byte Read	$140									0	0	0	0	0	0	1	0
VIC Base Register	4-61	VIC_BASE	Byte Write	$141									A15	A14	A13	A12	A11	X	LPT Mode	VIC Enable
Extended/Standard Address	4-62	EXT_STD_ADDR	Word Read/Write	$142	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
On-Board Video Status	4-63		Byte Read	$143									X	X	X	X	X	X	X	On-Board Video Status
128 MB_en	4-63		Byte Write	$143									X	X	X	X	X	X	0	128 Mb Mode Enable
Rearm Interrupt	4-63	REARM_INT	Byte Write	$144									D	D	D	D	D	D	D	D
Slave A16 Address Mask	4-63	SL_A16_MASK	Word Write	$148	A15	A14	A13	A12	A11	A10	A9	A8	X	X	X	X	X	X	X	X
Slave A32/A24 Address Mask	4-63	SL_A32_A24_MASK	Word Write	$14A	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
Slave A16 Address Compare	4-63	SL_A16_ADDR	Word Write	$14C	A15	A14	A13	A12	A11	A10	A9	A8	X	X	X	X	X	X	X	X
Slave A32/A24 Address Compare	4-63	SL_A32_A24_ADDR	Word Write	$14E	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
Size Register	4-64		Word Read/Write	$150	X	X	X	X	X	X	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
Remap Register	4-64		Word Read/Write	$152	X	X	X	X	X	X	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
VIC VMEbus Interface Controller Chip	4-65		Byte	$400-$4FF	(see detailed map on following page)															
IRQ12 Interrupt ID	4-64	IRQ12_ID	Byte Read	$502									D	D	D	D	D	D	D	D
IRQ11 Interrupt ID	4-64	IRQ11_ID	Byte Read	$504									D	D	D	D	D	D	D	D
NMI Interrupt ID	4-64	NMI_ID	Byte Read	$508									D	D	D	D	D	D	D	D

Notes on Table 4-8: An “X” denotes a reserved location. 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).

A “D” bit can be either set or clear.

The I/O addresses listed for the VIC VMEbus Controller Chip and the IRQ and NMI Interrupt ID Registers are the default locations. Their actual locations are programmable through the VIC Base Register.

Table 4-9 VIC Register Map

REGISTER NAME	PAGE	MNEMONIC	I/O ADR	D7	D6	D5	D4	D3	D2	D1	D0
VMEbus Interrupter Interrupt Control	4-65	VIICR	$403	VMEbus Interrupt Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 1	4-66	VICR1	$407	IRQ1 Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 2	4-66	VICR2	$40B	IRQ2 Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 3	4-66	VICR3	$40F	IRQ3 Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 4	4-66	VICR4	$413	IRQ4 Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 5	4-66	VICR5	$417	IRQ5 Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 6	4-66	VICR6	$41B	IRQ6 Mask	X	X	X	X	IPL Value		
VMEbus Interrupt Control 7	4-66	VICR7	$41F	IRQ7 Mask	X	X	X	X	IPL Value		
DMA Status Interrupt Control	4-67	DMASICR	$423	DMA Status Interrupt Mask	X	X	X	X	IPL Value		
Local Interrupt Control 1	4-68	LICR1	$427	LIRQ1 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ1 State	IPL Value		
Local Interrupt Control 2	4-68	LICR2	$42B	LIRQ2 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ2 State	IPL Value		
Local Interrupt Control 3	4-68	LICR3	$42F	LIRQ3 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ3 State	IPL Value		
Local Interrupt Control 4	4-68	LICR4	$433	LIRQ4 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ4 State	IPL Value		
Local Interrupt Control 5	4-68	LICR5	$437	LIRQ5 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ5 State	IPL Value		
Local Interrupt Control 6	4-68	LICR6	$43B	LIRQ6 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ6 State	IPL Value		
Local Interrupt Control 7	4-68	LICR7	$43F	LIRQ7 Mask	High Polarity	Edge Enable	Auto-vector	LIRQ7 State	IPL Value		
ICGS Interrupt Control	4-70	ICGSICR	$443	ICGS3 Mask	ICGS2 Mask	ICGS1 Mask	ICGS0 Mask	X	IPL Value		
ICMS Interrupt Control	4-70	ICMSICR	$447	ICMS3 Mask	ICMS2 Mask	ICMS1 Mask	ICMS0 Mask	X	IPL Value		
Error Group Interrupt Control	4-71	EGICR	$44B	ACFAIL Interrupt Mask	Writepost Fail Interrupt Mask	Arbitration Timeout Interrupt Mask	SYSFAIL Interrupt Mask	SYSFAIL State	IPL Value		
ICGS Interrupt Vector Base	4-73	ICGSVBR	$44F	Status/ID Value						Global Switch Number	
ICMS Interrupt Vector Base	4-73	ICMSVBR	$453	Status/ID Value						Module Switch Number	
Local Interrupt Vector Base	4-74	LIVBR	$457	Status/ID Value					Local Interrupt Number		

Table 4-9 VIC Register Map (Continued)

REGISTER NAME	PAGE	MNEMONIC	I/O ADR	D7	D6	D5	D4	D3	D2	D1	D0
Error Group Interrupt Vector Base	4-75	EGIVBR	$45B	Status/ID Value					Group Interrupt Number		
Interprocessor Communications Switch	4-105	ICSR	$45F	ICGS Switches				ICMS Switches			
Interprocessor Communications 0	4-105	ICR0	$463	User Data							
Interprocessor Communications 1	4-105	ICR1	$467	User Data							
Interprocessor Communications 2	4-105	ICR2	$46B	User Data							
Interprocessor Communications 3	4-105	ICR3	$46F	User Data							
Interprocessor Communications 4	4-105	ICR4	$473	User Data							
VIC Version	4-105	ICR5	$477	VIC Version Number							
Reset/Halt Status	4-105	ICR6	$47B	IRESET Status	IRESET/ HALT Status	X	X	X	X	Reset/Halt Status	
Mailbox Semaphore	4-106	ICR7	$47F	SYSFAIL Mask	HALT/ RESET Control	VMEbus Master Status	ICR4	ICR3	ICR2	ICR1	ICR0
VMEbus Interrupt Request Status	4-76	VIRSR	$483	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	Enable Register
VMEbus Interrupt Vector Base 1	4-76	VIVBR1	$487	IRQ1 Status/ID Vector							
VMEbus Interrupt Vector Base 2	4-76	VIVBR2	$48B	IRQ2 Status/ID Vector							
VMEbus Interrupt Vector Base 3	4-76	VIVBR3	$48F	IRQ3 Status/ID Vector							
VMEbus Interrupt Vector Base 4	4-76	VIVBR4	$493	IRQ4 Status/ID Vector							
VMEbus Interrupt Vector Base 5	4-76	VIVBR5	$497	IRQ5 Status/ID Vector							
VMEbus Interrupt Vector Base 6	4-76	VIVBR6	$49B	IRQ6 Status/ID Vector							
VMEbus Interrupt Vector Base 7	4-76	VIVBR7	$49F	IRQ7 Status/ID Vector							
Transfer Timeout	4-77	TTR	$4A3	VMEbus Timeout Period			Local Bus Timeout Period			Arbitration Timeout	Include VMEbus Acquisition
Local Bus Timing	4-80	LBTR	$4A7	Minimum PAS Deasserted Time			Minimum DS Deasserted Time	Minimum PAS Asserted Time			
Block Transfer Definition	4-81	BTDR	$4AB	D64 Boundary Crossing	D64 Slave Enable	Accel. Block Transfer	D64 Master Enable	Boundary Crossing Enable		AMSR Enable	Dual-Path Enable

Table 4-9 VIC Register Map (Continued)

REGISTER NAME	PAGE	MNEMONIC	I/O ADR	D7	D6	D5	D4	D3	D2	D1	D0
Interface Configuration	4-82	ICR	$4AF	RMC Control 3	RMC Control 2	RMC Control 1	Deadlock Signaling		Meta-stability Interval	Turbo Enable	SCON Status
Arbiter/Requester Configuration	4-84	ARCR	$4B3	Arbitration Mode	VMEbus Request Level		DRAM Refresh	Fairness Timer Enable			
Address Modifier Source	4-85	AMSR	$4B7	AM2-0 Option	AM5-3 Slave Qual	Address Modifier Code					
Bus Error Status	4-86	BESR	$4BB	VMEbus Master	Local Bus Error	VMEbus Bus Error	VMEbus Timeout	Local Bus Timeout	SLSEL0 Self-Access	SLSEL1 Self-Access	VMEbus Acquire Timeout
DMA Status	4-87	DMASR	$4BF	Master Writepost Info	X	X	VMEbus Bus Error	Local Bus Error	BERR During DMA	LBERR During DMA	Block Transfer
Slave Select 0 Control Register 0	4-89	SS0CR0	$4C3	Periodic Interrupt Timer		SLSEL0 Supervisor Access	SLSEL0 D32 Enable	SLSEL0 Address Space Configuration		SLSEL0 Local transfer Mode	
Slave Select 0 Control Register 1	4-91	SS0CR1	$4C7	SLSEL0 Timing Field 1				SLSEL0 Timing Field 0			
Slave Select 1 Control Register 0	4-94	SS1CR0	$4CB	Slave Writepost Enable	Master Writepost Enable	SLSEL1 Supervisor Access	SLSEL1 D32 Enable	SLSEL1 Address Space Configuration		SLSEL1 Local Transfer Mode	
Slave Select 1 Control Register 1	4-96	SS1CR1	$4CF	SLSEL1 Timing Field 1				SLSEL1 Timing Field 0			
Release Control	4-99	RCR	$4D3	Release Mode		Block Transfer Burst Length					
Block Transfer Control	4-100	BTCR	$4D7	Module-Based DMA	Block Transfer with DMA	MOVEM Enable	Data Direction	Interleave Period			
Block Transfer Length 0	4-102	BTLR0	$4DB	Block Transfer Length (Least Significant Count)							
Block Transfer Length 1	4-102	BTLR1	$4DF	Block Transfer Length (Next Most Significant Count)							
Block Transfer Length 2	4-102	BTLR2	$4E7	Block Transfer Length (Most Significant Count)							
System Reset	4-103	SRR	$4E3	System Reset Field							

Notes on Table 4-8:

All VIC registers should only be accessed as bytes.

The I/O addresses listed for these registers are the default locations with the VIC base address of I/O $400. The actual base address is programmable through the VIC Base Register.

The VIC occupies 256 bytes of I/O addressing space from its base at offset $00 to offset $FF: all addresses within that space not assigned to a register are considered Reserved and should not be accessed.

Shaded registers are unique because they are available to other VMEbus masters in slave access mode.

Table 4-10 Slave Access Register Map

REGISTER NAME	PAGE	MNEMONIC	SHORT I/O ADDR	D7	D6	D5	D4	D3	D2	D1	D0
Interprocessor Communications 0	4-105	ICR0	$01	User Data							
Interprocessor Communications 1	4-105	ICR1	$03	User Data							
Interprocessor Communications 2	4-105	ICR2	$05	User Data							
Interprocessor Communications 3	4-105	ICR3	$07	User Data							
Interprocessor Communications 4	4-105	ICR4	$09	User Data							
VIC Version	4-105	ICR5	$0B	VIC Version Number							
Reset/Halt Status	4-105	ICR6	$0D	IRESET Status	IRESET/ HALT Status	X	X	X	X	Reset/Halt Status	
Mailbox Semaphore	4-106	ICR7	$0F	SYSFAIL Mask	HALT/ RESET Control	VMEbus Master Status	ICR4	ICR3	ICR2	ICR1	ICR0
Clear ICGS0 Switch	4-108	ICGS0C	$10	Any write access clears ICGS0 (privileged access only)							
Set ICGS0 Switch	4-108	ICGS0S	$11	Any write access sets ICGS0 (privileged access only)							
Clear ICGS1 Switch	4-108	ICGS1C	$12	Any write access clears ICGS1 (privileged access only)							
Set ICGS1 Switch	4-108	ICGS1S	$13	Any write access sets ICGS1 (privileged access only)							
Clear ICGS2 Switch	4-108	ICGS2C	$14	Any write access clears ICGS2 (privileged access only)							
Set ICGS2 Switch	4-108	ICGS2S	$15	Any write access sets ICGS2 (privileged access only)							
Clear ICGS3 Switch	4-108	ICGS3C	$16	Any write access clears ICGS3 (privileged access only)							
Set ICGS3 Switch	4-108	ICGS3S	$17	Any write access sets ICGS3 (privileged access only)							
Clear ICMS0 Switch	4-108	ICMS0C	$20	Any write access clears ICMS0							
Set ICMS0 Switch	4-108	ICMS0S	$21	Any write access sets ICMS0							
Clear ICMS1 Switch	4-108	ICMS1C	$22	Any write access clears ICMS1							
Set ICMS1 Switch	4-108	ICMS1S	$23	Any write access sets ICMS1							
Clear ICMS2 Switch	4-108	ICMS2C	$24	Any write access clears ICMS2							
Set ICMS2 Switch	4-108	ICMS2S	$25	Any write access sets ICMS2							
Clear ICMS3 Switch	4-108	ICMS3C	$26	Any write access clears ICMS3							
Set ICMS3 Switch	4-108	ICMS3S	$27	Any write access sets ICMS3							
Reserved			$28-$FF								

Notes on Table 4-8: All VIC slave-addressable registers should only be accessed as bytes.

The Short I/O addresses listed for these registers define only the offset address. The base address is supplied by the Slave A16 Address Compare Register.

All registers are addressable with either privileged or nonprivileged access except the Set/Clear ICGS registers, which may only be accessed as privileged.

The VIC slave-addressable registers occupy 256 bytes of Short I/O addressing space: all addresses within that space not assigned to a register are considered Reserved and should not be accessed.

Table 4-10 Slave Access Register Map (Continued)

REGISTER NAME	PAGE	MNEMONIC	SHORT I/O ADDR	D7	D6	D5	D4	D3	D2	D1	D0

Shaded Interprocessor Communications Registers are unique because they are also available to the local processor.

SYSTEM REGISTER DETAILS

General Purpose Command Register

The General Purpose Command Register (GP_COMMAND) is a write-only byte register at I/O address $140. The eight active bits in this register control certain global aspects of VMEbus activity. All active bits in this register are cleared on powerup or hard reset.

General Purpose Command Register GP_COMMAND (Write-Only byte at I/O $140)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Status LED	Master Big-Endian	Slave Big-Endian	Slave A32 Enable	Mailbox Enable	BLT Word Enable	VME Enable	Slave A24 Enable

General Purpose Command Register: Status LED bit (D7)

Bit 7 controls the front panel LED. Note that the default state is on. Thus, the front panel LED will be on after powerup 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

General Purpose Command Register: Big-Endian bits (D6, D5)

The 5x86 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-7586 has hardware on-board to support either transfer method, and the byte-swapping hardware is controlled by bits 5 and 6 of this register.

D6/D5	Big-Endian Bit Function
0	Use Little-Endian multiple-byte transfers (default)
1	Use Big-Endian multiple-byte transfers

Bit 6 controls byte swapping for the VMIVME-7586 as a VMEbus master, while bit 5 controls byte swapping when the VMIVME-7586 is being accessed as a VMEbus slave device.

Any byte swapping is transparent both to the local CPU and to VMEbus devices, but byte swapping is not supported during unaligned transfers. See the discussion on byte swapping in Section 12 for more details.

General Purpose Command Register: Slave A32/A24 Enable bits (D4, D0)

Bits 4 and 0 control slave access to the VMIVME-7586. The default clear state disables all slave accesses to the VMIVME-7586.

D4/D0	Slave Enable Bit Function
0	The VMIVME-7586 will not respond to slave accesses from the associated addressing mode (default)
1	The VMIVME-7586 will respond to slave accesses from the associated addressing mode

Bit 4 controls VMIVME-7586 slave access as an A32 Extended device. When set, the VMIVME-7586 may be accessed using A32 Extended VMEbus addressing mode. Bit 0 functions similarly, but determines whether or not the VMIVME-7586 responds to A24 Standard addressing.

If the slave interface is disabled using these bits and a device attempts to access the VMIVME-7586, the VMIVME-7586 will not generate a VMEbus DTACK signal, generally causing a bus error (as long as either the VMIVME-7586 or another board has an active bus error timer).

General Purpose Command Register: Mailbox Enable bit (D3)

Bit 3 controls slave access to the VMIVME-7586's mailbox registers within the VIC. The default clear state disables slave accesses to the mailbox registers.

D3	Mailbox Enable Bit Function
0	The mailbox registers will not respond to slave accesses (default)
1	The mailbox registers will respond to slave accesses

The Mailbox Enable bit allows the programmer to configure the VIC and slave address registers before enabling access to the mailbox registers. Some applications may also require separate control over the mailbox registers from dual-port RAM.

General Purpose Command Register: BLT Word Enable bit (D2)

Bit 2 enables 16-bit Block Mode Transfers (BLTs). The default clear state disables word transfers using BLT, allowing only longword transfers.

D2	BLT Word Enable Bit Function
0	Enable 32-bit VMEbus Block Mode Transfers (default)
1	Enable 16-bit VMEbus Block Mode Transfers

General Purpose Command Register: VME Enable bit (D1)

Bit 1 controls master VMEbus access. While this bit is clear (the default state after powerup or reset), the VMIVME-7586 cannot access the VMEbus as a master. While VMEbus access is disabled, the VMIVME-7586 VMEbus interface registers may still be accessed, but no data transfers to or from the VMEbus will occur, except through the slave interface. Likewise, no VMEbus signal can cause an interrupt on the VMIVME-7586.

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

The VMIVME-7586 is never completely isolated from the VMEbus, however, regardless of the VME Enable bit. The slave interface is enabled separately, for example. Also, the VMIVME-7586 will still generate a VMEbus SYSRESET* when the front panel reset button is pressed, as long

as the VMIVME-7586 is configured as the system controller. Likewise, if the VMIVME-7586 is not the system controller, it will still respond to a VMEbus SYSRESET* regardless of the VME Enable bit status.

If VMEbus accesses are disabled, an attempted access will terminate normally, but no data will actually be written, and any data read will be indeterminate.

Product ID Register

The Product ID Register (BRD_ID) is a read-only byte register at I/O address $140. This register always contains the value $01, which uniquely identifies the VMIVME-7586 from other VMIC VMEbus products.

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

VIC Base Register

The VIC Base Register (VIC_BASE) is a write-only byte register at I/O address $141. This register determines the base I/O address of both the VIC and the Interrupt Acknowledge Registers. All active bits in this register are cleared on powerup or hard reset.

VIC Base Register VIC_BASE (Write-Only byte at I/O $141)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
A15	A14	A13	A12	A11	X	LPT Mode	VIC Enable

VIC Base Register: A11-A15 bit field (D7-D3)

These are the upper five address bits that determine the base address of the VIC chip. For base address determination, just create a 16-bit address using this bit field for the most significant bits, and assume A10 is always set and all the lower bits are clear. For example, the default value of $00 places the VIC chip at a base of I/O $400 (only A10 is set). The IRQ12 Interrupt ID, IRQ11 Interrupt ID, and NMI Interrupt ID Registers are always at offsets of $102, $104, and $108 above the VIC base address.

Note that most VIC register addresses in this manual assume this field is left at its default value, placing the VIC base address at I/O $400 and the

Interrupt Acknowledge Registers starting at I/O $502. There is little reason to ever change the default setting.

VIC Base Register: LPT Mode bit (D1)

This utility bit controls whether or not the VMIVME-7586's parallel port is bidirectional or output only. A clear LPT Mode bit (the default) makes the port bidirectional. A set bit forces the port into an output-only configuration.

D1	LPT Mode Bit Function
0	The VMIVME-7586 printer port is bidirectional (default)
1	The VMIVME-7586 printer port is output-only

VIC Base Register: VIC Enable Bit (D0)

When set, the VIC Enable bit allows the VIC chip to be accessed at the base address determined by the A11-A15 field described above. The default clear state disables VIC access. Attempts to access the VIC registers or Interrupt Acknowledge Registers without the VIC Enable bit set initiates an access to the ISA bus. This access terminates properly but retrieves undefined information if the access was a read.

D0	VIC Enable Bit Function
0	VIC chip access is disabled (default)
1	The VIC chip responds to accesses at the location determined by the VIC Base Register

Extended/Standard Address Register

The Extended/Standard Address Register (EXT_STD_ADDR) is a write-only word register at I/O address $142. The register consists of a single bit field that supplies the upper VMEbus address bits for both the Real Mode and Protected Mode VMEbus Windows.

Extended/Standard Address Register EXT_STD_ADDR (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	A23	A22	A21	A20	A19	A18	A17	A16

When accessing the VMEbus through the Real Mode VMEbus Window, this register supplies any and all of the upper address bits, while the address modifier bits are supplied by the VIC Address Modifier Source Register.

When accessing the VMEbus through the Protected Mode VMEbus Window, only the most significant byte of this register (bits D8-D15) is used, since all other address bits can be obtained from the 5x86 address lines. The address modifier used in Protected Mode depends on which portion of the 1024 Mbyte window the VMEbus is accessed through. See page 4-10 for more details regarding VMEbus access in Protected Mode.

On-Board Video Status Register

The Video Status Register indicates that the on-board video controller is selected or are replaced by a PC/104 video controller. The system BIOS will look at this register to determine if a PC104 video adapter has been installed. If a video adapter is present on the PC104, the system BIOS will disable the on-board VGA controller and its BIOS and will install the adapter's BIOS extension. This enables the user to connect any standard PC104 display device. This register works in conjunction with jumpers E1, E2 and E3 as described in Table 2-1 on page 2-4

128 Mbyte Mode Enable Register

The 128 Mbyte Mode Enable Register is active high and enables the Master Mode protected window. See "128 mbyte - protected mode access" on page 13 for a complete description of the 128 Mbyte mode.

Rearm Interrupt Register

The Rearm Interrupt Register (REARM_INT) is a write-only byte register at I/O address $144. The data written here does not matter; any write to this register will rearm the IRQ11 and IRQ12 interrupts. This register should be written at the end of the interrupt service routines.

Slave Address Mask/Compare Registers

These four related write-only word registers are associated with slave mode access to the VMIVME-7586. See the discussion regarding these registers and slave mode on page 4-16. Programming these registers and then setting the Mailbox Enable bit in the General Purpose Command Register allows VMEbus resources to access the VIC Interprocessor Communication Registers.

Size Register

The Size Register controls address bits A16 through A24 through the I/O word port $150. The register is active low. Setting all bits to 1 ($1FF) in the port $150 will disable the associated remap functionality. Setting the I/O word port $150 to $001 would, for example, map a 128 Kbyte region from the VMEbus to an address region specified by the Remap Register. the default is $1FF, disabled.

Remap Register

The Remap Register is an alternative address which affects bits A16 through A24. The remap address can be programmed into the I/O word port $152; however, if the associated Size Register bit is set to 1, the remap address bit is ignored. For example, setting I/O word port $150 to $001 and I/O word port $152 to $1FE will map a 128 Kbyte region from the VMEbus to the physical address starting at $01FE 0000 and ending at $01FF FFFF. Powerup defaults disable remapping.

INTERRUPT ACKNOWLEDGE REGISTER DETAILS

There are three read-only byte Interrupt Acknowledge Registers at default I/O addresses $502, $504, and $508 corresponding to PC/AT interrupts IRQ12, IRQ11, and NMI, respectively. These registers should only be read within their corresponding interrupt service routines, since the act of reading any of these registers initiates an interrupt acknowledge cycle for the corresponding interrupt level.

When read, the value returned is the 8-bit ID used to determine which of the interrupt sources caused the interrupt. If multiple interrupts are queued, each read access will return the vector for the highest priority interrupt on that level. Table 4-4 on page 4-32 details the interrupt priorities. The function of the three Interrupt Acknowledge Registers is discussed in detail in Section 6 on page 4-32.

The IRQ11 and IRQ12 signals can only originate from the VIC, but the NMI can have other origins; therefore, special care must be taken in the NMI ISR to ensure that the VIC is the source of the pending NMI before accessing the NMI ID Register. The simplest way to accomplish this is to check bit 6 of the PC/AT I/O port at I/O_$61. A set bit indicates a VIC-sourced NMI. If the bit is clear, the ISR should pass control back to the system NMI handler without reading the NMI ID Register.

Note that the addresses listed for these registers are default addresses. Their actual location depends upon the value in the VIC Base Register. See the description of the VIC Base Register on page 4-61 for details.

VIC REGISTER DETAILS

All VIC registers are physically read/write byte registers, although the functions of some of the registers make them meaningful only as read-only or write-only. All VIC registers are shown here at their default locations. See the description of the VIC Base Register on page 4-61 for details regarding moving the VIC registers around in I/O addressing space.

For complete VIC information, the programmer is strongly urged to acquire the ***VIC64 User's Guide*** from Cypress Semiconductor (see Chapter 1 for a list of references). Comparing the information here with the information in the VIC manual, a few comments are in order:

- The register and bit definitions in this manual are customized for this controller; therefore, some more general functions described in the VIC manual either do not apply to the VMIVME-7586 or have slightly different applications. Such differences are pointed out in this manual wherever possible.
- The default values listed here are generally those resulting from a VIC Global Reset (as opposed to a VIC Internal Reset or VIC System Reset). The only exceptions result from BIOS register modifications, and these are noted in the text.

VMEbus Interrupter Interrupt Control Register

The VMEbus Interrupter Interrupt Control Register (VIICR) is a read/write byte register at VIC offset I/O address $403. This register provides enabling and IPL level encoding for the local interrupt issued when a VMEbus interrupt is acknowledged. This register is initialized to a value of $F8 upon powerup or hard reset.

VMEbus Interrupter Interrupt Control Register VIICR (VIC offset I/O $403)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
VMEbus Interrupt Mask	X	X	X	X	IPL Value	IPL Value	IPL Value

VMEbus Interrupter Interrupt Control Register: VMEbus Interrupt Mask bit (D7)

When this bit is clear, the VIC signals a local interrupt at the acknowledgment of a previously issued VMEbus interrupt. When set, the VIC will not issue a local interrupt.

D7	VMEbus Interrupt Mask Bit Function
0	The VIC issues a local interrupt upon acknowledgment of a VMEbus interrupt
1	The VIC will not issue a local interrupt at the acknowledgment of a VMEbus interrupt (default)

VMEbus Interrupter Interrupt Control Register: IPL Value bit field (D2-D0)

This value is inverted and driven onto the IPL lines when an interrupt is acknowledged.

VMEbus Interrupt Control Registers

The seven VMEbus Interrupt Control Registers (VICR1–7) are read/write byte registers at VIC offset I/O addresses $407, $40B, $40F, $413, $417, $41B, and $41F, respectively corresponding to VMEbus interrupts 1–7. These registers provide enabling of the VIC as VMEbus interrupt handler for any or all of the VMEbus interrupts. Seven registers exist to provide unique masking and IPL values for the seven VMEbus interrupts. Each register is initialized to a value of $F8 upon powerup or hard reset. See Section 6 on page 4-32 for a detailed discussion of interrupt handling procedures.

VMEbus Interrupt Control Registers VICR1–7 (VIC offset I/O $403–$41F)							
D7	D6	D5	D4	D3	D2	D1	D0
IRQ Mask	X	X	X	X	IPL Value	IPL Value	IPL Value

VMEbus Interrupt Control Registers: IRQ Mask bit (D7)

When this bit is clear, the VIC acts as a VMEbus interrupt handler by signaling a local interrupt at the specified IPL level. When set, the VIC does not handle the VMEbus interrupt and no local interrupt is issued.

D7	IRQ Mask Bit Function
0	The VIC signals the corresponding VMEbus interrupt with a local interrupt at the specified IPL level
1	The associated interrupt is masked off (default)

VMEbus Interrupt Control Registers: IPL Value bit field (D2-D0)

This interrupt level value is inverted and driven onto the IPL lines when an interrupt is acknowledged. The VMIVME-7586 supports PC/AT NMI interrupts on level 4, and IRQ11 and IRQ12 interrupts on IPL levels 2 and 1, respectively.

WARNING

ALL ACTIVE VIC INTERRUPT SOURCES MUST BE PROGRAMMED TO INTERRUPT ON EITHER LEVEL 4, LEVEL 2, OR LEVEL 1. PROGRAMMING THE INTERRUPT LEVEL TO ANY OTHER VALUE WILL RESULT IN UNDEFINED BEHAVIOR.

DMA Status Interrupt Control Register

The DMA Status Interrupt Control Register (DMASICR) is a read/write byte register at VIC offset I/O address $423. This register provides enabling and IPL-level encoding for the DMA-complete interrupt issued when any VIC local DMA operation completes (successfully or unsuccessfully). This register is initialized to a value of $F8 upon powerup or hard reset.

DMA Status Interrupt Control Register DMASICR (VIC offset I/O $423)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
DMA Status Interrupt Mask	X	X	X	X	IPL Value	IPL Value	IPL Value

DMA Status Interrupt Control Register: DMA Status Interrupt Mask bit (D7)

When this bit is clear, the VIC signals a local interrupt at the completion of any VIC local DMA operation. When set, the VIC will not issue a local interrupt.

D7	DMA Status Interrupt Mask Bit Function
0	The VIC issues a local interrupt upon completion of a local VIC DMA operation
1	The VIC will not issue a local interrupt at the completion of a local VIC DMA operation (default)

DMA Status Interrupt Control Register: IPL Value bit field (D2-D0)

This value is inverted and driven onto the IPL lines when an interrupt is acknowledged.

Local Interrupt Control Registers

The seven Local Interrupt Control Registers (LICR1-7) are read/write byte registers at VIC offset I/O addresses $427, $42B, $42F, $433, $437, $43B, and $43F, respectively corresponding to VMEbus interrupts 1-7. These registers provide enabling, IPL level, and control of local interrupts 1-7 (LIRQ1-7*). Note that by default, the LIRQ Mask bit in each LICR register is set, disabling the corresponding interrupt. Do not enable the interrupt until the interrupt handling routines are in place. Also, note that LIRC5 is reserved by VMIC and should never be altered from the default setting.

Local Interrupt Control Registers LICR1-7 (VIC offset I/O $427-$43F)							
D7	D6	D5	D4	D3	D2	D1	D0
LIRQ Mask	High Polarity	Edge Enable	Auto-vector	LIRQ1-7 State	IPL Value	IPL Value	IPL Value

Local Interrupt Control Registers: LIRQ Mask bit (D7)

When this bit is clear, the VIC is enabled to handle the corresponding local interrupt asserted on the LIRQ1-7* signals.

D7	LIRQ Mask Bit Function
0	The VIC handles the corresponding local interrupt asserted on the LIRQ1-7 signals
1	The VIC does not handle the asserted local interrupt (default)

Local Interrupt Control Registers: High Polarity bit (D6)

When this bit is set, the VIC responds to interrupts as active High if bit 5 is set (level sensitive) or on a rising edge if bit 5 is cleared (edge sensitive). When clear, the VIC responds to active Low or falling edges.

D6	High Polarity Bit Function
0	The VIC responds to active Low or falling edges (default)
1	The VIC responds to interrupts as active High if bit 5 is set (level sensitive) or on a rising edge if bit 5 is cleared (edge sensitive)

Local Interrupt Control Registers: Edge Enable bit (D5)

When this bit is clear, the VIC responds to the LIRQ1-7* as a level-sensitive interrupt. When set, the VIC responds to LIRQ1-7* as an edge-sensitive interrupt.

D5	Edge Enable Bit Function
0	The VIC responds to the LIRQ1-7* as a level-sensitive interrupt (default)
1	The VIC responds to LIRQ1-7* as an edge-sensitive interrupt

Local Interrupt Control Registers: Autovector bit (D4)

When this bit is set, autovector mode is enabled; in this mode, the VIC automatically supplies the interrupt status/ID vector for the local interrupt acknowledge cycle. Autovector mode is required for normal VMIVME-7586 operation. When clear, autovector mode is disabled; the interrupting source must provide the Status/ID vector to the processor.

D4	Autovector Bit Function
0	The interrupting source provides the status/ID vector for the local interrupt acknowledge cycle to the processor (default) (not supported on the VMIVME-7586)
1	Autovector mode. The VIC automatically supplies the interrupt status/ID vector for the local interrupt acknowledge cycle (required by the VMIVME-7586)

Local Interrupt Control Registers: LIRQ1-7 State bit (D3)

A clear bit indicates the LIRQ1-7* signal is asserted at the VIC.

Local Interrupt Control Registers: IPL Value bit field (D2-D0)

This value is inverted and driven onto the IPL lines when a local interrupt is presented on the LIRQ1-7* signals and bit 7 of this register is clear (enabled).

ICGS Interrupt Control Register

The ICGS Interrupt Control Register (ICGSICR) is a read/write byte register at VIC offset I/O address $443. This register provides enabling and IPL encoding for the four global switch interrupts. This register is initialized to a value of $F8 upon powerup or hard reset.

ICGS Interrupt Control Register ICGSICR (VIC offset I/O $443)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
ICGS3 Mask	ICGS2 Mask	ICGS1 Mask	ICGS0 Mask	X	IPL Value	IPL Value	IPL Value

ICGS Interrupt Control Register: ICGS Mask bits (D7-D4)

When this bit is clear, the VIC will issue and handle a local interrupt when the corresponding global switch is set. Bit 7 corresponds to ICGS3, bit 6 to ICGS2, bit 5 to ICGS1, and bit 4 to ICGS0.

D7-D4	ICGS Mask Bit Function
0	The VIC will issue and handle a local interrupt when the corresponding global switch is set
1	The VIC will not issue and handle a local interrupt when the corresponding global switch is set (default)

ICGS Interrupt Control Register: IPL Value bit field (D2-D0)

This value is inverted and driven onto the IPL signals when a global switch is acknowledged.

ICMS Interrupt Control Register

The ICMS Interrupt Control Register (ICMSICR) is a read/write byte register at VIC offset I/O address $447. This register provides enabling and

IPL encoding for the four module switch interrupts. This register is initialized to a value of $F8 upon powerup or hard reset.

ICMS Interrupt Control Register ICMSICR (VIC offset I/O $447)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
ICMS3 Mask	ICMS2 Mask	ICMS1 Mask	ICMS0 Mask	X	IPL Value	IPL Value	IPL Value

ICMS Interrupt Control Register: ICMS Mask bits (D7-D4)

When this bit is clear, the VIC will issue and handle a local interrupt when the corresponding module switch is set. Bit 7 corresponds to ICMS3, bit 6 to ICMS2, bit 5 to ICMS1, and bit 4 to ICMS0.

D7-D4	ICMS3 Mask Bit Function
0	The VIC will issue and handle a local interrupt when the corresponding module switch is set
1	The VIC will not issue and handle a local interrupt when the corresponding module switch is set (default)

ICMS Interrupt Control Register: IPL Value bit field (D2-D0)

This value is inverted and driven onto the IPL signals when a module switch is acknowledged.

Error Group Interrupt Control Register

The Error Group Interrupt Control Register (EGICR) is a read/write byte register at VIC offset I/O address $44B. This register provides enabling and IPL encoding for the error group interrupts.

Error Group Interrupt Control Register EGICR (VIC offset I/O $44B)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
ACFAIL Interrupt Mask	Write Post Fail Interrupt Mask	Arbitration Timeout Interrupt Mask	SYSFAIL Interrupt Mask	SYSFAIL State	IPL Value	IPL Value	IPL Value

Error Group Interrupt Control Register: ACFAIL Interrupt Mask bit (D7)

When this bit is clear, the VIC generates a local interrupt when ACFAIL* is detected as asserted.

D7	ACFAIL Mask Bit Function
0	The VIC generates a local interrupt when ACFAIL* is detected as asserted
1	The VIC will not generate a local interrupt when ACFAIL* is detected as asserted (default)

Error Group Interrupt Control Register: Write Post Fail Interrupt Mask bit (D6)

Write posting is not supported on the VMIVME-7586, therefore bit 6 should remain set.

Error Group Interrupt Control Register: Arbitration Timeout Interrupt Mask bit (D5)

When this bit is clear, the VIC generates a local interrupt when an arbitration timeout has occurred.

D5	Arbitration Timeout Interrupt Mask Bit Function
0	The VIC generates a local interrupt when an arbitration timeout has occurred
1	The VIC will not generate a local interrupt when an arbitration timeout has occurred (default)

Error Group Interrupt Control Register: SYSFAIL Interrupt Mask bit (D4)

When this bit is clear, the VIC generates a local interrupt when SYSFAIL* is asserted.

D4	SYSFAIL Interrupt Mask Bit Function
0	The VIC generates a local interrupt when SYSFAIL* is asserted
1	The VIC will not generate a local interrupt when an SYSFAIL is asserted (default)

Error Group Interrupt Control Register: SYSFAIL State bit (D3)

This bit is set whenever SYSFAIL* is detected asserted.

Error Group Interrupt Control Register: IPL Value bit field (D2-D0)

This value is inverted and driven onto the IPL signals when an error group interrupt is acknowledged.

ICGS Interrupt Vector Base Register

The ICGS Interrupt Vector Base Register (ICGSIVBR) is a read/write byte register at VIC offset I/O address $44F. This register provides the status/ID vector for the global switch interrupts. The status/ID must be programmed with a unique number for each VIC interrupt controller in the chassis to enable identification encoding for bits 1-0.

ICGS Interrupt Vector Base Register ICGSIVBR (VIC offset I/O $44F)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Status/ID Bit 5	Status/ID Bit 4	Status/ID Bit 3	Status/ID Bit 2	Status/ID Bit 1	Status/ID Bit 0	Global Switch Number	Global Switch Number

ICGS Interrupt Vector Base Register: Status/ID bit field (D7-D2)

These bits are user-definable and are used with bits 1-0 to provide a unique global switch interrupt status/ID vector.

ICGS Interrupt Vector Base Register: Global Switch Number bit field (D1-D0)

This read-only value indicates which global switch is pending during a global switch interrupt acknowledge cycle. These bits are used with bits 7-2 to provide a unique status/ID vector for each global switch. The numeric value of this field indicates the switch number. These bits are valid only during the interrupt acknowledge cycle.

ICMS Interrupt Vector Base Register

The ICMS Interrupt Vector Base Register (ICMSIVBR) is a read/write byte register at VIC offset I/O address $453. This register provides the status/ID vector for the module switch interrupts. The status/ID must be

programmed with a unique number for each VIC interrupt controller in the chassis to enable identification encoding for bits 1-0.

ICMS Interrupt Vector Base Register ICMSIVBR (VIC offset I/O $453)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Status/ID Bit 5	Status/ID Bit 4	Status/ID Bit 3	Status/ID Bit 2	Status/ID Bit 1	Status/ID Bit 0	Module Switch Number	Module Switch Number

ICMS Interrupt Vector Base Register: Status/ID bit field (D7-D2)

These bits are user-definable and are used with bits 1-0 to provide a unique module switch interrupt status/ID vector.

ICMS Interrupt Vector Base Register: Module Switch Number bit field (D1-D0)

This read-only value indicates which module switch is pending during a module switch interrupt acknowledge cycle. These bits are used with bits 7-2 to provide a unique status/ID vector for each module switch. The numeric value of this field indicates the switch number. These bits are valid only during the interrupt acknowledge cycle.

Local Interrupt Vector Base Register

The Local Interrupt Vector Base Register (LIVBR) is a read/write byte register at VIC offset I/O address $457. This register provides the status/ID vector for the local interrupts. The status/ID must be programmed with a unique number for each VIC interrupt controller in the chassis to enable identification encoding for bits 2-0.

Local Interrupt Vector Base Register LIVBR (VIC offset I/O $457)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Status/ID Bit 4	Status/ID Bit 3	Status/ID Bit 2	Status/ID Bit 1	Status/ID Bit 0	Local Interrupt Number	Local Interrupt Number	Local Interrupt Number

Local Interrupt Vector Base Register: Status/ID bit field (D7-D3)

These bits are user-definable and are used with bits 2-0 to provide a unique local interrupt status/ID vector.

Local Interrupt Vector Base Register: Local Interrupt Number bit field (D2-D0)

This read-only value indicates which local interrupt is pending during a local interrupt acknowledge cycle. These bits are used with bits 7-3 to provide a unique status/ID vector for each local interrupt. The numeric value of this field indicates the local interrupt number. These bits are valid only during the interrupt acknowledge cycle.

Error Group Interrupt Vector Base Register

The Error Group Interrupt Vector Base Register (EGIVBR) is a read/write byte register at VIC offset I/O address $45B. This register provides the status/ID vector for the error group interrupts. The status/ID must be programmed with a unique number for each VIC interrupt controller in the chassis to enable identification encoding for bits 2-0.

Error Group Interrupt Vector Base Register EGIVBR (VIC offset I/O $45B)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Status/ID Bit 4	Status/ID Bit 3	Status/ID Bit 2	Status/ID Bit 1	Status/ID Bit 0	Group Interrupt Number	Group Interrupt Number	Group Interrupt Number

Error Group Interrupt Vector Base Register: Status/ID bit field (D7-D3)

These bits are user-definable and are used with bits 2-0 to provide a unique interrupt status/ID vector.

Error Group Interrupt Vector Base Register: Group Interrupt Number bit field (D2-D0)

This read-only value indicates which group interrupt is pending during the interrupt acknowledge cycle. These bits are used with bits 7-3 to provide a unique status/ID vector for each error group interrupt. These bits are valid only during the interrupt acknowledge cycle.

D2	D1	D0	Error/Status Interrupt
0	0	0	ACFAIL* asserted
0	0	1	Write post failed
0	1	0	Arbitration timeout
0	1	1	SYSFAIL* asserted
1	0	0	VMEbus Interrupter interrupt acknowledge
1	0	1	DMA complete

Interprocessor Communications Registers

Please refer to the Interprocessor Communications Registers detailed beginning on page 4-103.

VMEbus Interrupt Request/Status Register

The VMEbus Interrupt Request/Status Register (VIRSR) is a read/write byte register at VIC offset I/O address $483. This register provides status and control of the VMEbus interrupts 7-1. This register is initialized to a value of $00 upon powerup or hard reset.

VMEbus Interrupt Request/Status Register VIRSR (VIC offset I/O $483)							
D7	D6	D5	D4	D3	D2	D1	D0
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	Enable Register

VMEbus Interrupt Request/Status Register: VMEbus Interrupt Switch bits (D7-D1)

Setting any of these bits asserts the VMEbus IRQi* signals corresponding to the bit positions, if bit 0 is set during the write. These bits are cleared by setting the appropriate bit and clearing bit 0.

VMEbus Interrupt Request/Status Register: Enable Register bit (D0)

This bit provides enabling and disabling for the VMEbus Interrupt Switches.

VMEbus Interrupt Vector Base Registers

The VMEbus Interrupt Vector Base Registers (VIVBR) are read/write byte registers at VIC offset I/O addresses $487, $48B, $48F, $493, $497, $49B,

$49F corresponding to IRQ1 – IRQ7, respectively. These registers contain a single 8-bit field that provides the status/ID vector for the VMEbus interrupt acknowledge cycles. These registers are initialized to a value of $0F upon powerup or hard reset.

I/O Address	Interrupt
$487	IRQ1
$48B	IRQ2
$48F	IRQ3
$493	IRQ4
$497	IRQ5
$49B	IRQ6
$49F	IRQ7

Transfer Timeout Register

The Transfer Timeout Register (TTR) is a read/write byte register at VIC offset I/O address $4A3. This register provides control of the local and VMEbus timeout timers. This register is initialized to a value of $68 upon powerup or hard reset.

Transfer Timeout Register TTR (VIC offset I/O $4A3)							
D7	D6	D5	D4	D3	D2	D1	D0
VMEbus Timeout Period	VMEbus Timeout Period	VMEbus Timeout Period	Local Bus Timeout Period	Local Bus Timeout Period	Local Bus Timeout Period	Arbitration Timeout	Include VMEbus Acquisition

Transfer Timeout Register: VMEbus Timeout Period bit field (D7-D5)

Defines the VMEbus timeout. Note that the hardware timer, if enabled, will override all settings here except the 4 µs value. See the discussion concerning the system controller functions on page 4-7.

D7	D6	D5	VMEbus Timeout (µs)
0	0	0	4
0	0	1	16
0	1	0	32
0	1	1	64 (default)
1	0	0	128
1	0	1	256
1	1	0	512
1	1	1	Infinite (timer disabled)

Transfer Timeout Register: Local Bus Timeout Period bit field (D4-D2)

Defines the local bus timeout.

D4	D3	D2	Local Bus Timeout (µs)
0	0	0	4
0	0	1	16
0	1	0	32 (default)
0	1	1	64
1	0	0	128
1	0	1	256
1	1	0	512
1	1	1	Infinite (timer disabled)

Transfer Timeout Register: Arbitration Timeout bit (D1)

When this bit is set, the VIC as VMEbus arbiter has detected a VMEbus arbitration timeout. This is only used when configured as the VMEbus system controller (SCON asserted).

D1	Arbitration Timeout Bit Function
0	The VIC as VMEbus arbiter has not detected a VMEbus arbitration timeout (default)
1	The VIC as VMEbus arbiter has detected a VMEbus arbitration timeout

Transfer Timeout Register: Include VMEbus Acquisition bit (D0)

When this bit is set, the local bus timer will include waiting for VMEbus acquisition. When clear, the local bus timer will stop and reset when the VMEbus is requested.

D0	Include VMEbus Acquisition Bit Function
0	The local bus timer will stop and reset when the VMEbus is requested (default)
1	The local bus timer will include waiting for VMEbus acquisition

Local Bus Timing Register

The Local Bus Timing Register (LBTR) is a read/write byte register at VIC offset I/O address $4A7. This register provides timing control for PAS* and DS* signals when the VIC is local bus master. In the following descriptions, ***n*** is the binary value specified in the bit fields. Clock latency may add one additional clock period (15.625 ns) to these times. This register is initialized to a value of $00 upon powerup or hard reset. When performing slave RMW cycles or BLT transfers, this register *must* be programmed with $FF.

Local Bus Timing Register LBTR (VIC offset I/O $4A7)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Minimum PAS Deasserted Time	Minimum PAS Deasserted Time	Minimum PAS Deasserted Time	Minimum DS Deasserted Time	Minimum PAS Asserted Time	Minimum PAS Asserted Time	Minimum PAS Asserted Time	Minimum PAS Asserted Time

Local Bus Timing Register: Minimum PAS Deasserted Time bit field (D7-D5)

This field specifies the minimum deasserted time for the PAS* signal whenever the VIC is the local bus master. The time is specified by the formula $(n+1) \times 15.625$ ns. This value must be set to maximum (all bits set) when performing slave RMW cycles or BLT transfers on the VMIVME-7586.

Local Bus Timing Register: Minimum DS Deasserted Time bit (D4)

This bit specifies the minimum deasserted time for the DS* signal whenever the VIC is the local bus master. A time of 15.625 ns is selected when this bit is clear; 31.250 ns is selected when this bit is set. This bit must be set when performing slave RMW cycles or BLT transfers on the VMIVME-7586.

D4	Minimum DS Deasserted Time Bit Function
0	The minimum deasserted time for the DS* signal is 15.625 ns (default)
1	The minimum deasserted time for the DS* signal is 31.250 ns

Local Bus Timing Register: Minimum PAS Asserted Time bit field (D3-D0)

This field specifies the minimum asserted time for the PAS* signal whenever the VIC is the local bus master. The time is specified by $(n+2) \times 15.625$ ns. The actual asserted time depends on a number of factors including local and VMEbus acknowledge timing. This value must be set to maximum (all bits set) when performing slave RMW cycles or BLT transfers on the VMIVME-7586.

Block Transfer Definition Register

The Block Transfer Definition Register (BTDR) is a read/write byte register at VIC offset I/O address $4AB. This register enables Block Transfer Mode. This register is initialized to a value of $00 upon powerup or hard reset. In order to enable Block Transfer Mode, all active bits in this field must be set.

Block Transfer Definition Register BTDR (VIC offset I/O $4AB)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
D64 Boundary Crossing	D64 Slave Enable	Accel Block Transfer	D64 Master Enable	VMEbus Boundary Crossing Enable	Local Boundary Crossing Enable	AMSR Enable	Dual Path Enable

Block Transfer Definition Register: D64 Boundary Crossing Enable bit (D7)

When set, the VIC64 assumes D64 transfers are aligned to a 2 Kbyte boundary.

Block Transfer Definition Register: D64 Slave Enable bit (D6)

When set, D64 slave block transfers are enabled.

Block Transfer Definition Register: Accelerated Block Transfer bit (D5)

When set, this bit enables accelerated block transfer operations by reducing the DSACK*-to-DTACK* time defined in the Slave Select Control registers by 1/2 clock periods.

Block Transfer Definition Register: D64 Master Enable bit (D4)

When set, this bit enables D64 master block transfers, although the Block Transfer with DMA bit (bit 6) of the Block Transfer Control Register must also be set.

Block Transfer Definition Register: Boundary Crossing Enable bits (D3-D2)

When set, these bits enable both local and VMEbus 256-byte boundary crossing. Both bits must be set to enable BLT transfers on the VMIVME-7586.

Block Transfer Definition Register: AMSR Enable bit (D1)

When set, the VIC issues the address modifier code from the Address Modifier Source Register (AMSR) during block transfers. This bit must be set to enable BLT transfers on the VMIVME-7586.

Block Transfer Definition Register: Dual Path Enable bit (D0)

When set, dual path mode is enabled on the VIC during master block transfers. This bit must be set to enable BLT transfers on the VMIVME-7586.

Interface Configuration Register

The Interface Configuration Register (ICR) is a read/write byte register at VIC offset I/O address $4AF. This register controls various features of the VIC including RMCs, deadlock signaling, metastability delays, and the "turbo" feature.

Interface Configuration Register ICR (VIC offset I/O $4AF)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
RMC Control Bit 3	RMC Control Bit 2	RMC Control Bit 1	Deadlock Signaling	Deadlock Signaling	Metastability Interval	Turbo Enable	SCON Status

Interface Configuration Register: RMC Control Bit 3 (D7)

This bit must remain clear.

Interface Configuration Register: RMC Control Bit 2 (D6)

When this bit is set, the VMEbus AS* is stretched when RMC* is asserted for VMEbus transfers.

D6	RMC Control Bit 2 Function
0	The VMEbus performs normal transfers (default)
1	The VMEbus AS* is stretched when RMC* is asserted for VMEbus transfers

In the current implementation, the RMC* signal is provided by the 5x86 LOCK* signal. This provides the capability to perform a RMW signal.

Interface Configuration Register: RMC Control Bit 1 (D5)

This bit must remain clear.

Interface Configuration Register: Deadlock Signaling bit field (D4-D3)

These bits configure deadlock signaling. Bit 4 is used to enable the assertion of HALT* and LBERR* in addition to the DEDLK* signal in deadlock situations. If bit 4 is enabled, bit 3 may be used to prevent the assertion of HALT* for RMC deadlocks. The default condition, using the DEDLK* signal alone, is recommended for the VMIVME-7586 and is the *only* condition possible for slave accesses.

D4	D3	Deadlock Signaling Bit Function
0	0	DEDLK* only (default) (only conditions possible for slave accesses)
0	1	
1	0	HALT*, LBERR*, DEDLK* (not recommended on the VMIVME-7586)
1	1	HALT*, LBERR*, DEDLK* (HALT* is not asserted for RMC cycles) (not recommended on the VMIVME-7586)

When performing slave accesses, bit 4 should always be clear.

Interface Configuration Register: Metastability Interval bit (D2)

When this bit is set, the VIC adds one additional CLK64M clock period (15.625 ns) of metastability delay on asynchronous inputs (from 3 CLK64M periods to 4).

D2	Metastability Interval Bit Function
0	3 CLK64M periods of metastability delay on asynchronous inputs (default)
1	4 CLK64M periods of metastability delay on asynchronous inputs

Interface Configuration Register: Turbo Enable bit (D1)

This bit must remain clear.

Interface Configuration Register: SCON Status bit (D0)

This read-only bit contains the value of the SCON* pin. When set, the VIC is not the VMEbus system controller. When clear, the VIC is the VMEbus system controller.

Arbiter/Requester Configuration Register

The Arbiter/Requester Configuration Register (ARCR) is a read/write byte register at VIC offset I/O address $4B3. This register provides configuration of the fairness timeout and DRAM refresh features. The VMEbus request level is also configured from this register. This register is initialized to a value of $60 upon powerup or hard reset.

Arbiter/Requester Configuration Register ARCR (VIC offset I/O $4B3)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Arbitration Mode	VMEbus Request Level	VMEbus Request Level	DRAM Refresh	Fairness Timer Enable	Fairness Timer Enable	Fairness Timer Enable	Fairness Timer Enable

Arbiter/Requester Configuration Register: Arbitration Mode bit (D7)

When this bit is set, the VIC performs priority VMEbus arbitration. When clear, the VIC performs round-robin arbitration. This bit is only relevant when the VIC is configured as the VMEbus system controller (SCON asserted).

D7	Arbitration Mode Bit Function
0	The VIC performs round-robin VMEbus arbitration (default)
1	The VIC performs priority VMEbus arbitration

Arbiter/Requester Configuration Register: VMEbus Request Level bit field (D6-D5)

The VMEbus request level is set according to the following table:

D6	D5	VMEbus Request Level
0	0	BR0
0	1	BR1
1	0	BR2
1	1	BR3 (default)

Arbiter/Requester Configuration Register: DRAM Refresh bit (D4)

When this bit is set, the VIC performs CAS-before-RAS (DS* before PAS*) refresh functions. This bit should always remain clear on the VMIVME-7586.

D4	DRAM Refresh Bit Function
0	The VIC performs no DRAM refresh functions (default)
1	The VIC performs CAS-before-RAS (DS* before PAS*) refresh functions. This bit should never be set on the VMIVME-7586.

Arbiter/Requester Configuration Register: Fairness Timer Enable bit field (D3-D0)

The VMEbus fair requester is enabled in this bit field according to the following table:

D3-D0	Timeout Period/Mode
$0	Fairness disabled (default)
$F	Timeout disabled
All other patterns	2 μs times number

Address Modifier Source Register

The Address Modifier Source Register (AMSR) is a read/write byte register at VIC offset I/O address $4B7. This register provides the user-definable address modifiers (AM codes) that can be sourced by the VIC for VMEbus master cycles, or used in validating AM codes during VMEbus slave cycles. This register is initialized to a value of $00 upon powerup or hard reset.

Address Modifier Source Register AMSR (VIC offset I/O $4B7)							
D7	D6	D5	D4	D3	D2	D1	D0
AM2-0 Option	AM5-3 Slave Qual	Address Modifier Code	Address Modifier Code	Address Modifier Code	Address Modifier Code	Address Modifier Code	Address Modifier Code

Address Modifier Source Register: AM2-0 Option bit (D7)

When this bit is set, the VIC issues the AM2-0 codes based on address bits A27/A26 (VIC signals FC2/FC1). AM5-3 will be issued from bits 5-3 of this register. Refer to Table 4-1 on page 4-11 to see the impact of this bit on

user-defined address modifier codes. This bit is clear by default; it is rarely necessary to change the default setting.

D7	AM2-0 Option Bit Function
0	The user-defined address modifier code (bits 5-0) is unqualified (default)
1	The VIC issues the AM2-0 codes based on bits 5-3 of this register and address bits A27/A26 according to Table 4-1 on page 4-11

Address Modifier Source Register: AM5-3 Slave Qual bit (D6)

When this bit is set, the VIC uses bits 5-3 in qualifying for slave accesses in addition to the address space size information defined by bits 3 and 2 of the SSiCR0s. This bit is overridden if bits 3 and 2 of the SSiCR0s are both set.

D6	AM5-3 Slave Qual Bit Function
0	The address modifier code (bits 5-0) is unqualified (default)
1	The VIC uses bits 5-3 in qualifying for slave accesses in addition to the address space size information defined by bits 3 and 2 of the SSiCR0s are both set

Address Modifier Source Register: Address Modifier Code bit field (D5-D0)

The AM code that is issued during master cycles or used for qualifying slave cycles. This register is used only when enabled for user-defined AM codes. Otherwise, standard VMEbus AM codes are used.

Bus Error Status Register

The Bus Error Status Register (BESR) is a read/write byte register at VIC offset I/O address $4BB. This register provides BERR/LBERR*, self-access, VMEbus mastership, and timeout status. All bits except bit 7 are flags that are automatically cleared on reset and must otherwise be cleared manually by the local processor after being set by status conditions. If these bits are

to be used for a specific operation, it is important that they be cleared prior to starting that operation.

Bus Error Status Register BESR (VIC offset I/O $4BB)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
VMEbus Master	Local Bus Error	VMEbus Bus Error	VMEbus Timeout	Local Bus Timeout	SLSEL0 Self-Access	SLSEL1 Self-Access	VMEbus Acquire Timeout

Bit	Bit Name	Function
D7	VMEbus Master	This bit is set whenever the VIC is VMEbus master. Unlike the VMEbus Master Status bit (D5) in the Mailbox Semaphore Register (ICR7), this bit is not qualified.
D6	Local Bus Error	This bit is set when a local bus error is signaled by a source other than the VIC (LBERR asserted to the VIC). Once set, this bit must be cleared manually. (Local bus errors will never occur in the current VMIVME-7586 configuration.)
D5	VMEbus Bus Error	This bit is set when a VMEbus bus error is signaled (BERR asserted). Once set, this bit must be cleared manually.
D4	VMEbus Timeout	This bit, when set, indicates the VIC has signaled a VMEbus timeout. This bit is relevant only if the VIC is system controller and the VMEbus timeout is enabled. Once set, this bit must be cleared manually.
D3	Local Bus Timeout	This bit, when set, indicates a local bus timeout occurred without qualification. Once set, this bit must be cleared manually.
D2	SLSEL0 Self-Access	This bit is set when the VIC is selected by the assertion on the SLSEL0 signal, while operating as VMEbus master. Once set, this bit must be cleared manually.
D1	SLSEL1 Self-Access	This bit is set when the VIC is selected by the assertion on the SLSEL1 signal, while operating as VMEbus master. Once set, this bit must be cleared manually.
D0	VMEbus Acquire Timeout	This bit, when set, indicates that a local bus timeout has occurred during an attempted acquisition of the VMEbus. Once set, this bit must be cleared manually.

DMA Status Register

The DMA Status Register (DMASR) is a read/write byte register at VIC offset I/O address $4BF. This register provides status of a VIC DMA transfer. This includes the block transfer with local DMA function and the module-based DMA function. Status bits are included to show various

BERR and LBERR statuses and DMA termination statuses. This register is initialized to a value of $60 upon powerup or hard reset.

DMA Status Register DMASR (VIC offset I/O $4BF)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Master Write Post Info	X	X	VMEbus Bus Error	Local Bus Error	BERR During DMA	LBERR During DMA	Block Transfer

DMA Status Register: Master Write Post Info bit (D7)

This bit is set whenever master write post information is stored. Since write posting is not supported on the VMIVME-7586, this bit should always be clear.

DMA Status Register: VMEbus Bus Error bit (D4)

This bit is set when a VMEbus bus error is signaled (BERR* asserted). This bit is a read-only copy of bit 5 of the BESR.

DMA Status Register: Local Bus Error bit (D3)

This bit is set when a local bus error is signaled by a source other than the VIC (LBERR* asserted to the VIC). This bit is a read-only copy of bit 6 of the BESR. Local bus errors cannot occur in the current VMIVME-7586 configuration, therefore, this bit will remain clear.

DMA Status Register: BERR During DMA bit (D2)

This bit, when set, indicates a BERR* was signaled during a DMA transfer. Once set, this bit must be cleared manually.

DMA Status Register: LBERR During DMA bit (D1)

This bit, when set, indicates a LBERR* was signaled during a DMA transfer. Once set, this bit must be cleared manually. Local bus errors cannot occur in the current VMIVME-7586 configuration, therefore, this bit will remain clear.

DMA Status Register: Block Transfer bit (D0)

This bit, when set, indicates an interleaved block transfer is in progress. Once set, this bit is cleared automatically at the completion of a local DMA operation.

Slave Select 0 Control Register 0

The Slave Select 0 Control Register 0 (SS0CR0) is a read/write byte register at VIC offset I/O address $4C3. This register provides control of the slave selection 0 facilities of the VIC, which is dedicated to A32 access only on the VMIVME-7586. Enabling of the IRQ2 timer interrupt is also configured in this register. This register is initialized to a value of $00 upon powerup or hard reset.

Slave Select 0 Control Register 0 SS0CR0 (VIC offset I/O $4C3)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Periodic Interrupt Timer	Periodic Interrupt Timer	Supervisory Access	D32 Enable	Address Space Configuration	Address Space Configuration	Local Transfer Mode	Local Transfer Mode

Slave Select 0 Control Register 0: Periodic Interrupt Timer bit field (D7-D6)

These bits enable and determine the frequency of the periodic LIRQ2 interrupt. If the VIC is to handle this local interrupt, LICR2 must be enabled. The frequencies for this interrupt are given below:

D7	D6	Timer Mode
0	0	Timer disabled (default)
0	1	50 Hz output on LIRQ2*
1	0	1000 Hz output on LIRQ2*
1	1	100 Hz output on LIRQ2*

Slave Select 0 Control Register 0: Supervisory Access bit (D5)

When this bit is set, SLSEL0* slave accesses are restricted to supervisory accesses. Other accesses are BERRed. Supervisory accesses are checked with the AM(2) signal.

D5	Supervisory Access Bit Function
0	SLSEL0 slave accesses are not restricted to supervisory accesses (default)
1	SLSEL0 slave accesses are restricted to supervisory accesses

Slave Select 0 Control Register 0: D32 Enable bit (D4)

When this bit is set, D32 slave operations are enabled for SLSEL0. This bit has no effect for enabling D32 master accesses.

D4	D32 Enable Bit Function (D32 Slave Access Control)
0	D32 slave operations are disabled for SLSEL0 (default)
1	D32 slave operations are enabled for SLSEL0

Slave Select 0 Control Register 0: Address Space Configuration bit field (D3-D2)

The SLSEL0 address space is configured according to the following table:

D3	D2	Address Space
0	0	A32 (extended) (default)
0	1	A24 (standard)
1	0	A16 (short)
1	1	User-defined, uses AMSR

Slave Select 0 Control Register 0: Local Transfer Mode bit field (D1-D0)

These bits set the local transfer mode when the VIC is local bus master for both slave and master block transfers.

D1	D0	Mode
0	0	No support is given for slave block transfers on SLSEL0. The VIC will BERR* any attempt to receive a VMEbus block transfer. Master block transfers with local DMA will not function in this mode. (Default)
0	1	Emulate single-cycle transfers on the local bus. Not supported by the VMIVME-7586.
1	0	Accelerated transfers on the local bus. In this mode, the VIC asserts the PAS* signal for the entire slave block transfer and master block transfer with local DMA. The DSACKi* signals should be held asserted in this mode. This method must be used for block transfers on the VMIVME-7586.
1	1	Reserved.

When performing BLT transfers, bit 1 must always be set and bit 0 must be clear (for accelerated local bus transfers).

Slave Select 0 Control Register 1

The Slave Select 0 Control Register 1 (SS0CR1) is a read/write byte register at VIC offset I/O address $4C7. This register provides the various access and acquisition timings for slave transfers and slave block transfers for SLSEL0* in addition to data acquisition timing for master block transfers with local DMA. This register is initialized to a value of $00 upon powerup or hard reset.

Slave Select 0 Control Register 1 SS0CR1 (VIC offset I/O $4C7)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Timing Field 1	Timing Field 1	Timing Field 1	Timing Field 1	Timing Field 0	Timing Field 0	Timing Field 0	Timing Field 0

Slave Select 0 Control Register 1: Timing Field 1 bit field (D7-D4)

This bit field establishes the following data access/acquisition timings:

- Second and subsequent cycle of a slave block transfer for SLSEL0* (SBAT1)
- Second and subsequent cycle of a master block transfer with local DMA (MBAT1)

The delays are programmed in multiples of the 64 MHz clock period according to the following table:

D7	D6	D5	D4	Timing Delay
0	0	0	0	0 ns (default)
0	0	0	1	31.2500 ns
0	0	1	0	39.0625 ns
0	0	1	1	46.8750 ns
0	1	0	0	54.6875 ns
0	1	0	1	62.5000 ns
0	1	1	0	70.3125 ns
0	1	1	1	78.1250 ns
1	0	0	0	85.9375 ns
1	0	0	1	93.7500 ns
1	0	1	0	101.5625 ns
1	0	1	1	109.3750 ns
1	1	0	0	117.1875 ns
1	1	0	1	125.0000 ns
1	1	1	0	132.8125 ns
1	1	1	1	140.6250 ns

Slave Select 0 Control Register 1: Timing Field 0 bit field (D3-D0)

This bit field establishes the following data access/acquisition timings:

- Single-cycle slave access timing for SLSEL0* (SAT)
- First cycle of a slave block transfer for SLSEL0* (SBAT0)
- First cycle of a master block transfer with local DMA (MBAT0)

The delays are programmed in multiples of the 64 MHz clock period according to the following table:

D3	D2	D1	D0	Timing Delay
0	0	0	0	0 ns (default)
0	0	0	1	31.2500 ns
0	0	1	0	39.0625 ns
0	0	1	1	46.8750 ns
0	1	0	0	54.6875 ns
0	1	0	1	62.5000 ns
0	1	1	0	70.3125 ns
0	1	1	1	78.1250 ns
1	0	0	0	85.9375 ns
1	0	0	1	93.7500 ns
1	0	1	0	101.5625 ns
1	0	1	1	109.3750 ns
1	1	0	0	117.1875 ns
1	1	0	1	125.0000 ns
1	1	1	0	132.8125 ns
1	1	1	1	140.6250 ns

Slave Select 1 Control Register 0

The Slave Select 1 Control Register 0 (SS1CR0) is a read/write byte register at VIC offset I/O address $4CB. This register provides control of the slave selection 1 facilities of the VIC, which is dedicated for A24 access on the VMIVME-7586. Master and slave write posting is enabled in this register as well. This register is initialized to a value of $00 upon powerup or hard reset.

Slave Select 1 Control Register 0 SS1CR0 (VIC offset I/O $4CB)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Slave Write Post Enable	Master Write Post Enable	Supervisory Access	D32 Enable	Address Space Configuration	Address Space Configuration	Local Transfer Mode	Local Transfer Mode

Slave Select 1 Control Register 0: Slave Write Post Enable bit (D7)

When this bit is set, slave write posting is enabled. Write posting is not supported on the VMIVME-7586, therefore, bit 7 should always remain clear.

Slave Select 1 Control Register 0: Master Write Post Enable bit (D6)

When this bit is set, master write posting is enabled. Write posting is not supported on the VMIVME-7586, therefore, bit 6 should always remain clear.

Slave Select 1 Control Register 0: Supervisory Access bit (D5)

When this bit is set, SLSEL1* slave accesses are restricted to supervisory accesses. Other accesses are BERRed. Supervisory accesses are checked with the AM(2) signal.

D5	Supervisory Access Bit Function
0	SLSEL1* slave accesses are not restricted to supervisory accesses (default)
1	SLSEL1* slave accesses are restricted to supervisory accesses

Slave Select 1 Control Register 0: D32 Enable bit (D4)

When this bit is set, D32 slave operations are enabled for SLSEL1. This bit has no effect for enabling D32 master accesses.

D4	D32 Enable Bit Function
0	D32 slave operations are disabled for SLSEL1 (default)
1	D32 slave operations are enabled for SLSEL1

Slave Select 1 Control Register 0: Address Space Configuration bit field (D3-D2)

In the VMIVME-7586 implementation, SLSEL1 is dedicated to A24 slave access, therefore, the only value supported here is %01. The SLSEL1 address space is configured according to the following table:

D3	D2	Address Space
0	0	A32 (extended) (default)
0	**1**	**A24 (standard) (only mode supported by the VMIVME-7586)**
1	0	A16 (short)
1	1	User-defined, uses AMSR

Slave Select 1 Control Register 0: Local Transfer Mode bit field (D1-D0)

These bits set the local transfer mode when the VIC is local bus master for both slave and master block transfers.

D1	D0	Mode
0	0	No support is given for slave block transfers on SLSEL1. The VIC will BERR* any attempt to receive a VMEbus block transfer. (Default)
0	1	Emulate single-cycle transfers on the local bus. Not supported by the VMIVME-7586.
1	0	Accelerated transfers on the local bus. In this mode, the VIC asserts PAS* for the entire slave block transfer. The DSACKi* signals should be held asserted in this mode. This mode must be used for block transfers on the VMIVME-7586.
1	1	Reserved.

When performing BLT transfers, bit 1 must always be set and bit 0 must be clear (for accelerated local bus transfers).

Slave Select 1 Control Register 1

The Slave Select 1 Control Register 1 (SS1CR1) is a read/write byte register at VIC offset I/O address $4CF. This register provides the various access and acquisition timings for slave transfers and slave block transfers for SLSEL1*. This register is initialized to a value of $00 upon powerup or hard reset.

Slave Select 1 Control Register 1 SS1CR1 (VIC offset I/O $4CF)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Timing Field 1	Timing Field 1	Timing Field 1	Timing Field 1	Timing Field 0	Timing Field 0	Timing Field 0	Timing Field 0

Slave Select 1 Control Register 1: Timing Field 1 (D7-D4)

This bit field establishes the following data access/acquisition timing:

- Second and subsequent cycle of a slave block transfer for SLSEL1* (SBAT1)

The delays are programmed in multiples of the 64 MHz clock period according to the following table:

D7	D6	D5	D4	Timing Delay
0	0	0	0	0 ns (default)
0	0	0	1	31.2500 ns
0	0	1	0	39.0625 ns
0	0	1	1	46.8750 ns
0	1	0	0	54.6875 ns
0	1	0	1	62.5000 ns
0	1	1	0	70.3125 ns
0	1	1	1	78.1250 ns
1	0	0	0	85.9375 ns
1	0	0	1	93.7500 ns
1	0	1	0	101.5625 ns
1	0	1	1	109.3750 ns
1	1	0	0	117.1875 ns
1	1	0	1	125.0000 ns
1	1	1	0	132.8125 ns
1	1	1	1	140.6250 ns

Slave Select 1 Control Register 1: Timing Field 0 bit field (D3-D0)

This bit field establishes the following data access/acquisition timings:

- Single-cycle slave access timing for SLSEL1* (SAT)
- First cycle of a slave block transfer for SLSEL1* (SBAT0)

The delays are programmed in multiples of the 64 MHz clock period according to the following table:

D3	D2	D1	D0	Timing Delay
0	0	0	0	0 ns (default)
0	0	0	1	31.2500 ns
0	0	1	0	39.0625 ns
0	0	1	1	46.8750 ns
0	1	0	0	54.6875 ns
0	1	0	1	62.5000 ns
0	1	1	0	70.3125 ns
0	1	1	1	78.1250 ns
1	0	0	0	85.9375 ns
1	0	0	1	93.7500 ns
1	0	1	0	101.5625 ns
1	0	1	1	109.3750 ns
1	1	0	0	117.1875 ns
1	1	0	1	125.0000 ns
1	1	1	0	132.8125 ns
1	1	1	1	140.6250 ns

Release Control Register

The Release Control Register (RCR) is a read/write byte register at VIC offset I/O address $4D3. This register configures the VMEbus release mode. The burst count for block transfers with local DMA (and VME64 block transfers for controllers with the VIC64 option installed) is also configured in the RCR. This register is initialized to a value of $00 upon powerup or hard reset.

Release Control Register RCR (VIC offset I/O $4D3)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Release Mode	Release Mode	Block Transfer Burst Length	Block Transfer Burst Length	Block Transfer Burst Length	Block Transfer Burst Length	Block Transfer Burst Length	Block Transfer Burst Length

Release Control Register: Release Mode bit field (D7-D6)

This bit field defines the release mode used by the VIC when releasing the VMEbus after the completion of a VMEbus transfer.

D7	D6	Release Mode
0	0	ROR - Release-on-Request (default)
0	1	RWD - Release-When-Done
1	0	ROC - Release on BCLR* assertion
1	1	BCAP - VMEbus Capture and Hold

Release Control Register: Block Transfer Burst Length bit field (D5-D0)

The burst length for block transfers with local DMA are configured in this bit field. The value indicates the number of cycles per block transfer (not the number of bytes). A value of 0 in this bit field indicates the maximum 64 cycles per burst. All other values correspond directly to the burst count.

For MBLT D64 block transfers (only available on controllers with the VIC64 option installed), the burst length is 4 times the actual field contents, and a value of 0 implies 256 cycles (4 x 64).

Block Transfer Control Register

The Block Transfer Control Register (BTCR) is a read/write byte register at VIC offset I/O address $4D7. This register provides control of the VIC block transfers. The local interleave periods and data direction are defined in this register. The enabling bits for all of the VIC's block transfer modes are located here as well. These enabling bits are mutually exclusive and more than one should not be set at the same time. Note that the VMIVME-7586 supports neither module-based DMA nor MOVEM transfers. This register is initialized to a value of $00 upon powerup or hard reset.

Block Transfer Control Register BTCR (VIC offset I/O $4D7)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Module-Based DMA	Block Transfer with DMA	MOVEM Enable	Data Direction	Interleave Period	Interleave Period	Interleave Period	Interleave Period

Block Transfer Control Register: Module-Based DMA bit (D7)

Since the VMIVME-7586 does not support module-based DMA transfers, this bit must always remain clear. When this bit is set, module-based DMA transfers are enabled.

D7	Module-Based DMA Bit Function
0	Concludes a module-based DMA transfer in progress (default)
1	Enables module-based DMA transfers (Bits D6 and D5 must be clear) (not supported by the VMIVME-7586)

Block Transfer Control Register: Block Transfer with DMA bit (D6)

When this bit is set, block transfers with local DMA are enabled. After this bit is set, the next assertion of MWB* is considered the initiation cycle of a VMEbus block transfer with local DMA. Clearing this bit concludes a block transfer with local DMA in progress. It is important to set this bit immediately before and clear this bit immediately after the actual block

transfer. When performing BLT transfers, bit 6 must not be cleared until after the BLT is complete.

D6	Block Transfer with DMA Bit Function
0	Concludes a block transfer with local DMA in progress (default)
1	Enables block transfers with local DMA (Bits D7 and D5 must be clear)

Block Transfer Control Register: MOVEM Enable bit (D5)

Since MOVEM transfer mode is not supported by the VMIVME-7586, this bit should always remain clear. When this bit is set, MOVEM transfers are enabled.

Block Transfer Control Register: Data Direction bit (D4)

This bit defines the direction of a block transfer with local DMA (MOVEM data direction determined by the R/W signal). When set, VMEbus block reads occur. When clear, VMEbus block writes occur.

D4	Data Direction Bit Function
0	VMEbus block writes occur (default)
1	VMEbus block reads occur

Block Transfer Control Register: Interleave Period bit field (D3-D0)

The interleave period for block transfers is defined here. The interleave period is 250 ns times the value programmed in this bit field.

Block Transfer Length Registers

The Block Transfer Length Registers (BTLR1-0, or BTLR2-0 on controllers with the VIC64 option installed) are read/write byte registers at VIC offset I/O addresses $4E7 (BTLR2 on the VIC64 only), $4DB (BTLR1), and $4DF (BTLR0). If the VIC64 option is not installed, location $4E7 is considered reserved. These registers configure the byte count for block transfers with local DMA, including D64 transfers if the VIC64 option is installed. BTLR2, if present, is the most significant byte of the transfer count. BTLR1 is considered the next most significant byte and BTLR0 the least significant. Bit 0 of BRLR0 must never be set because this implies at least one 8-bit transfer is required to complete the block transfer. Only D16, D32, and D64 block transfers are supported. If bit 0 of BTLR0 is set, the block transfer length is ignored and only one burst is performed. These registers are initialized to a value of $00 upon powerup or hard reset.

Block Transfer Length Registers BTLR2 (VIC offset I/O $4E7)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Block Transfer Length (Most Significant Count on VIC64 only; bits 23-16)							

Block Transfer Length Registers BTLR1 (VIC offset I/O $4DB)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Block Transfer Length (Next Most Significant Count; bits 15-8)							

Block Transfer Length Registers BTLR0 (VIC offset I/O $4DF)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Block Transfer Length (Least Significant Count; bits 7-0)							

Block Transfer Length Registers: Block Transfer Length bit field (D7-D0)

Defines the block transfer length in bytes. BTLR2 contains the most significant 8 bits (bits 23-16) on VIC64 chips only. BTLR1 contains the next most significant 8 bits of the length (bits 15-8), and BTLR0 the least (bits 7-0). The count must always be even; therefore, bit D0 of BTLR0 must always be clear.

System Reset Register

The System Reset Register (SRR) is a read/write byte register at VIC offset I/O address $4E3. The system reset register provides the means to perform a VMEbus system reset (SYSRESET* asserted). Writing a value of $F0 causes this function to occur. A system reset is also performed within the VIC. This register is initialized to a value of $FF upon powerup or hard reset.

System Reset Register SRR (VIC offset I/O $4E3)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
System Reset Field							

System Reset Register: System Reset bit field (D7-D0)

Writing this bit field with a value of $F0 causes SYSRESET* to be asserted for a minimum of 200 ms and a system reset to be performed within the VIC. All VIC registers will thereupon return to their default values.

INTERPROCESSOR COMMUNICATIONS REGISTERS

The Interprocessor Communications Registers (also known as “Mailbox Registers”) facilitate multiple processors on the VMEbus. Unlike the other VIC registers and System Registers, most of these registers are available to both the local processor as well as all other VMEbus controllers. Using these registers, all VMEbus controllers can synchronize their activities and even interrupt one another.

All Interprocessor Communications Registers appear in both local I/O space and VMEbus Short I/O space, except for the Interprocessor Communications Switch Register (which is a local-only register), and the Set/Clear ICGS and ICMS Switch Registers (which are slave-only). Local addresses listed are default addresses defined by the VIC Base Register. VMEbus slave Short I/O addresses have a base address defined by the value in the Slave A16 Address Compare Register.

See the Interprocessor Communications section on page 4-13 for more details concerning the configuration and use of the mailbox registers. Also, refer to Table 4-8 on page 4-53 for a complete slave register map.

Interprocessor Communications Switch Register

The Interprocessor Communications Switch Register (ICSR) is a read/write byte register at VIC offset I/O address $45F. This register provides setting, clearing, and monitoring of the interprocessor switch interrupts by way of the local bus. If the switch interrupts are enabled, setting these bits (more precisely, a clear-to-set transition) causes a local interrupt to occur in the same way as if the switch was set over the VMEbus. This register is initialized to a value of $00 upon powerup or hard reset. This is the only Interprocessor Communications Register that is *not* available to other VMEbus masters; it is a local-only register.

Interprocessor Communications Switch Register ICSR (VIC offset I/O $45F)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
ICGS Switch 3	ICGS Switch 2	ICGS Switch 1	ICGS Switch 0	ICMS Switch 3	ICMS Switch 2	ICMS Switch 1	ICMS Switch 0

Interprocessor Communications Switch Register: ICGS Switches bit field (D7-D4)

Bits 4, 5, 6, and 7 correspond to ICGS 0, 1, 2, and 3, respectively. If the switch interrupts are enabled, a clear-to-set transition causes the associated interrupt.

Interprocessor Communications Switch Register: ICMS Switches bit field (D3-D0)

Bits 0, 1, 2, and 3 correspond to ICMS 0, 1, 2, and 3, respectively. If the switch interrupts are enabled, a clear-to-set transition causes the associated interrupt.

Interprocessor Communication Registers

The five Interprocessor Communication Registers (ICR0-4) are read/write byte registers at VIC offset I/O addresses $463, $467, $46B, $46F, and $473, respectively. Their VMEbus slave Short I/O offset addresses are $01, $03, $05, $07, and $09, responding to both privileged and nonprivileged accesses. These are general-purpose read/write registers that can be accessed from either the local bus or the VMEbus. These registers are initialized to a value of $00 upon powerup or hard reset.

Interprocessor Communication Registers ICR0-4 (VIC offset I/O $463-$473) (VMEbus Slave Short I/O offset address $01-$09)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Data Field							

VIC Version Register

The VIC Version Register (ICR5) is a read-only byte register at VIC offset I/O address $477. Its VMEbus slave Short I/O offset address is $0B, responding to both privileged and nonprivileged accesses. This register provides the VIC or VIC64 version/revision number.

VIC Version Register ICR5 (VIC offset I/O $477) (VMEbus Slave Short I/O offset address $0B)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
VIC Version Number							

Reset/Halt Status Register

The Reset/Halt Status Register (ICR6) is a read/write byte register at VIC offset I/O address $47B. Its VMEbus slave Short I/O offset address is $0D, responding to both privileged and nonprivileged accesses. This register provides local or remote reset and HALT*.

Reset/Halt Status Register ICR6 (VIC offset I/O $47B) (VMEbus Slave Short I/O offset address $0D)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
IRESET Status	IRESET/ HALT Status	X	X	X	X	Reset/ HALT Status	Reset/ HALT Status

Reset/Halt Status Register: IRESET Status bit (D7)

On a VMEbus read, this bit indicates that the VIC is in a reset state. On a local bus read, this bit is set whenever ACFAIL* is asserted. This bit is read-only.

Reset/Halt Status Register: IRESET/HALT Status bit (D6)

This bit is read-only from the VMEbus and is set upon assertion of IRESET* and/or HALT*. It is set whether HALT* is asserted by external sources or by the VIC. SYSFAIL* is asserted when this bit is set if the SYSFAIL* mask bit (ICR7, bit 7) is cleared.

Reset/Halt Status Register: Reset/HALT Status bit field (D1-D0)

These bits are read-only from the VMEbus and provide reset/HALT* status of the VIC and local resources according to the following table:

D1	D0	Reset/HALT Status
0	0	Local resources are running and operational. This pattern must be written by the local CPU after a reset condition to indicate that local resources are running and operational.
0	1	HALT* has been asserted longer than 6 µs by a source other than the VIC. These bits may both be reset by the local CPU to indicate local resources are running and operational.
1	0	The VIC has performed a local reset function and the VIC is not the system controller. These bits may both be reset by the local CPU to indicate local resources are running and operational.
1	1	Indicates that the CPU has just been released from a system reset.

Mailbox Semaphore Register

The Mailbox Semaphore Register (ICR7) is a read/write byte register at VIC offset I/O address $47F. Its VMEbus slave Short I/O offset address is $0F, responding to both privileged and nonprivileged accesses. This register provides semaphores to the five general-purpose interprocessor communication registers (ICR4-0). The remaining bits indicate VMEbus master status, generate HALT* and RESET*, and mask SYSRESET*.

Mailbox Semaphore Register ICR7 (VIC offset I/O $47F) (VMEbus Slave Short I/O offset address $0F)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
SYSFAIL Mask	HALT/ RESET Control	VMEbus Master Status	ICR4	ICR3	ICR2	ICR1	ICR0

Mailbox Semaphore Register: SYSFAIL Mask bit (D7)

When this bit is set, the VIC is prohibited from asserting SYSFAIL* in response to bit 6 of ICR6 being set (which, by default, is set after any reset).

D7	SYSFAIL Mask Bit Function
0	The VIC may assert SYSFAIL* in response to bit 6 of ICR6 being set (default)
1	The VIC is prohibited from asserting SYSFAIL* in response to bit 6 of ICR6 being set

Bit 7 of this register controls the VMEbus SYSFAIL* mask signal. The VMIVME-7586 asserts SYSFAIL* automatically after any reset. Since other controllers on the VMEbus may not function normally while the SYSFAIL* line is active, VMIC recommends that the SYSFAIL Mask bit be set immediately after powerup or reset initialization routines, even if no VMEbus activity is planned. This is especially important if the VMIVME-7586 is not the slot-1 controller, since an active SYSFAIL* line may prevent the controller from performing its normal functions.

NOTE:

THE VMIVME-7586 ACTIVATES THE VMEbus SYSFAIL* SIGNAL UPON POWERUP OR RESET. IT IS IMPORTANT TO DEACTIVATE SYSFAIL* AS SOON AS POSSIBLE TO ALLOW NORMAL VMEbus ACTIVITY.

Mailbox Semaphore Register: HALT/RESET Control bit (D6)

This bit may be used to assert the HALT* and RESET* pins by way of software. Whenever this bit is set, the VIC asserts HALT* and RESET* until this bit is cleared or any reset occurs.

Mailbox Semaphore Register: VMEbus Master Status bit (D5)

This read-only bit is set whenever the VIC is the VMEbus master, and the VIC is asserting AS*. This bit is not set when the VIC is VMEbus master to an idle bus in ROR and BCAP release modes. Bit 7 of the BESR may be used to indicate that the VIC is VMEbus master when AS* is not asserted.

Mailbox Semaphore Register: ICR4-0 bit field (D4-D0)

These bits provide semaphores to the five interprocessor communication registers ICR4-0, respectively. Each bit is set when the corresponding ICR is written. These bits can be read or written from the local bus or the VMEbus.

Set/Clear ICGS Switch Registers (Slave-Only)

The eight Set/Clear ICGS Switch Registers (ICGS0-3 S/C) are write-only byte registers at VMEbus Short I/O offset addresses $10-17, responding to privileged accesses only. Address $10 clears ICGS0 and address $11 sets ICGS0; address $12 clears ICGS1 and address $13 sets ICGS1, and so on (see Table 4-8 on page 4-53 for a complete slave register map). These registers are slave-only and not addressable in the local processor's I/O space (although they are available to the local processor through the VMEbus just as any VMEbus resource). The data written to these registers is irrelevant; any write access performs the associated function. For example, any write to address $11 sets the ICGS0 switch, causing an interrupt on all masters that have not masked their ICGS0 switch.

Set/Clear ICGS Switch Registers (Slave-Only) ICGS0-3 S/C (VMEbus Slave Short I/O offset address $10-$17 - privileged only)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Any write access performs the associated set or clear function.							

Set/Clear ICMS Switch Registers (Slave-Only)

The eight Set/Clear ICMS Switch Registers (ICMS0-3 S/C) are write-only byte registers at VMEbus Short I/O offset addresses $20-27, responding to both privileged and nonprivileged accesses. Address $20 clears ICMS0 and address $21 sets ICMS0; address $22 clears ICMS1 and address $23 sets ICMS1, and so on (see Table 4-8 on page 4-53 for a complete slave register map). These registers are slave-only and not addressable in the local processor's I/O space (although they are available to the local processor through the VMEbus just as any VMEbus resource). The data written to

these registers is irrelevant; any write access performs the associated function. For example, any write to address $21 sets the ICMS0 switch, causing an interrupt on all masters that have not masked their ICMS0 switch.

Set/Clear ICMS Switch Registers (Slave-Only) ICMS0-3 S/C (VMEbus Slave Short I/O offset address $20-$27)							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Any write access performs the associated set or clear function.							

CHAPTER

5 MAINTENANCE

SECTION 1 - 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**.

SECTION 2 - MAINTENANCE PRINTS

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

APPENDIX

A

CONNECTOR PINOUTS

IN THIS APPENDIX:

SECTION 1 - INTRODUCTION	**A-1**
SECTION 2 - ETHERNET CONNECTOR PINOUT	**A-3**
SECTION 3 - FLOPPY DRIVE CONNECTOR PINOUT	**A-4**
SECTION 4 - IDE HARD DRIVE CONNECTOR PINOUT	**A-5**
SECTION 5 - KEYBOARD CONNECTOR PINOUT	**A-6**
SECTION 6 - PC/104 CONNECTOR PINOUT	**A-7**
SECTION 7 - PRINTER CONNECTOR PINOUT	**A-9**
SECTION 8 - SERIAL CONNECTOR PINOUT	**A-10**
SECTION 9 - VIDEO CONNECTOR PINOUT	**A-11**
SECTION 10 - VMEbus CONNECTOR PINOUT	**A-12**

SECTION 1 - INTRODUCTION

The VMIVME-7586 PC/AT Compatible VMEbus Controller has several connectors for its many I/O ports. Figure A-1 on page A-2 shows the locations of the connectors. Wherever possible, the VMIVME-7586 uses connectors and pinouts typical for any desktop PC. This ensures maximum compatibility with a minimum of confusion.

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

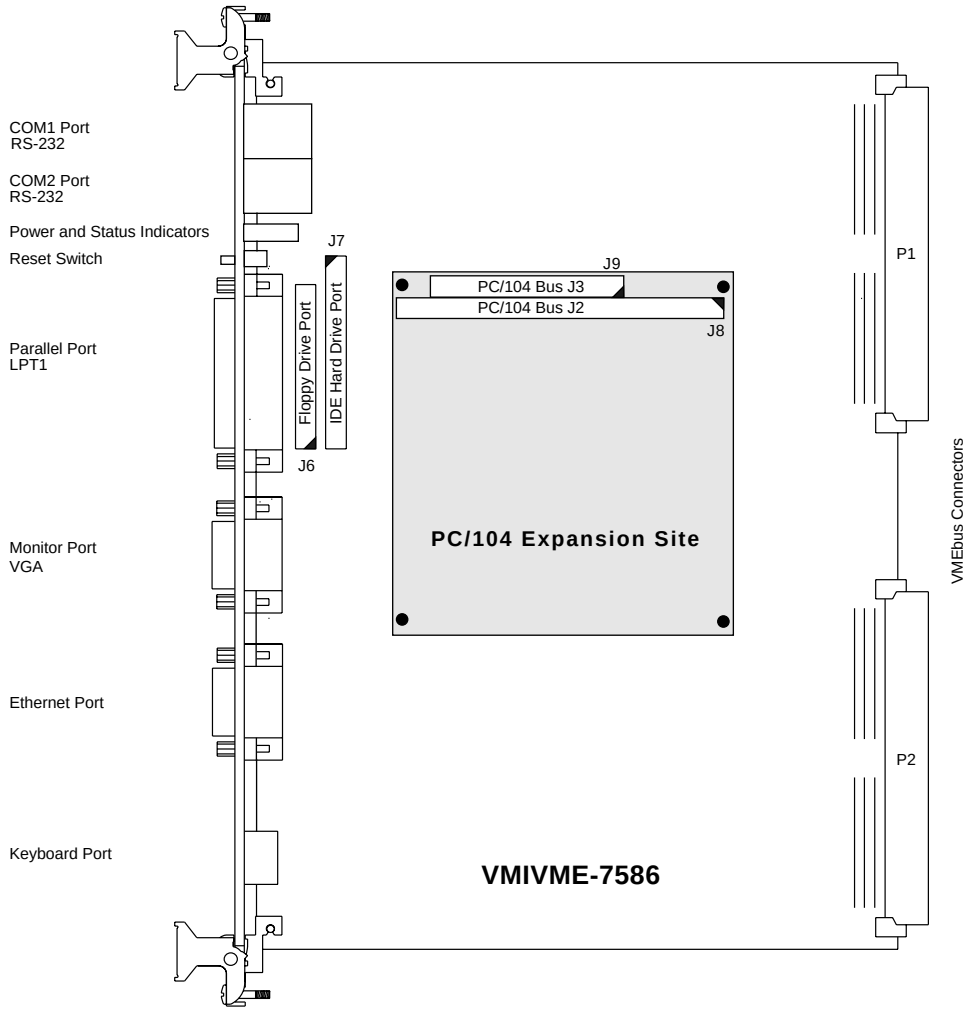
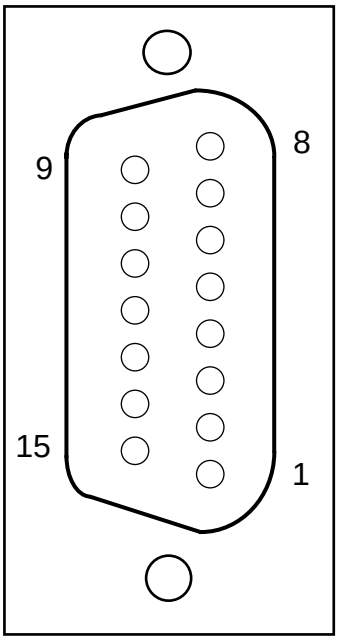


Figure A-1 VMIVME-7586 Connector Locations

SECTION 2 - ETHERNET CONNECTOR PINOUT

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

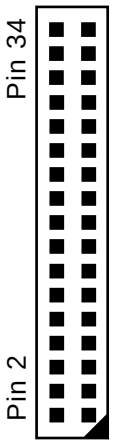


ETHERNET CONNECTOR		
PIN	**DIR**	**FUNCTION**
1		GND
2	In	CD+
3	Out	TX+
4		GND
5	In	RX+
6		GND
7		Reserved
8		GND
9	In	CD-
10	Out	TX-
11		GND
12	In	RX-
13		+12 VDC
14		GND
15		Reserved

Figure A-2 Ethernet Connector Pinout

SECTION 3 - FLOPPY DRIVE CONNECTOR PINOUT

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

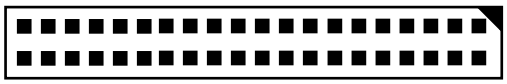


FLOPPY DRIVE CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1		Ground
2	From Controller	Drive Density 0 (not connected on some controllers)
3		Ground
4		Reserved
5		Ground
6	From Controller	Drive Density 1 (not connected on some controllers)
7		Ground
8	From Drive	Index
9		Ground
10	From Controller	Motor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

Figure A-3 Floppy Drive Connector Pinout

SECTION 4 - IDE HARD DRIVE CONNECTOR PINOUT

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



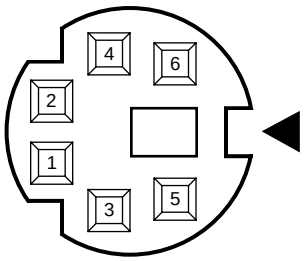
Pin 40 Pin 2

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

Figure A-4 IDE Hard Drive Connector Pinout

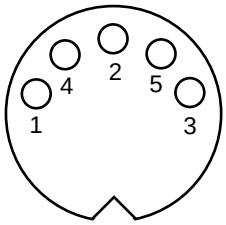
SECTION 5 - KEYBOARD CONNECTOR PINOUT

The keyboard connector is a standard 6-pin female mini-DIN PS/2 style connector shown in Figure A-5; an adapter is supplied to connect a keyboard with a larger PC/AT-style connector to the VMIVME-7586. The PC/AT-style connector pinout is shown in Figure A-6.



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

Figure A-5 PS/2 Keyboard Connector Pinout



KEYBOARD CONNECTOR	
PIN	**FUNCTION**
1	+ Keyboard Clock (+5 VDC Signal Level)
2	+ Keyboard Data (+5 VDC Signal Level)
3	Free
4	Ground
5	+5 VDC

Figure A-6 PC/AT Keyboard Connector Pinout

SECTION 6 - PC/104 CONNECTOR PINOUT

Controllers supporting PC/104 expansion sites have a pair of header connectors which function as standard PC/104 connectors J1 and J2. The PC/104 specification designates pins by rows lettered A-D and numbers. The pin numbering is rather unusual; please refer to Figure A-7 to determine the physical location of the pins in Table A-1.

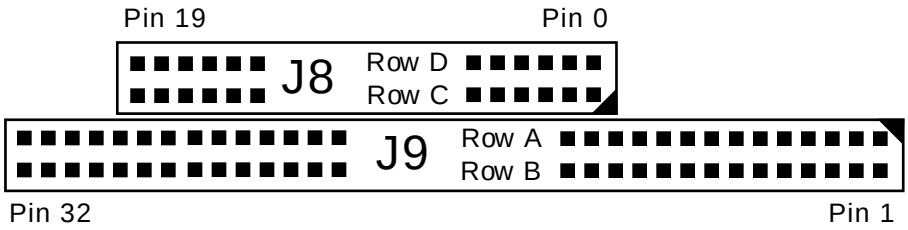


Figure A-7 PC/104 Connector Diagram

Table A-1 PC/104 Connector Pinout

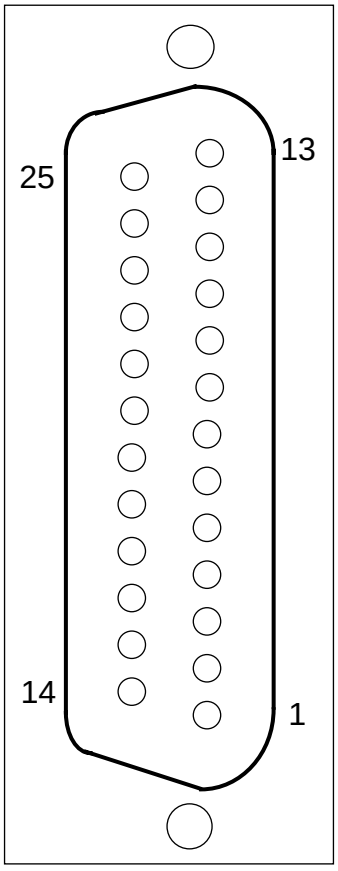
PIN	J1 ROW A	J1 ROW B	J2 ROW C	J2 ROW D
0			0 V	0 V
1	IOCHCHK	0 V	SBHE	MEMCS16
2	SD7	RESETDRV	LA23	IOCS16
3	SD6	+5 V	LA22	IRQ10
4	SD5	IRQ9	LA21	IRQ11
5	SD4	-5 V	LA20	IRQ12
6	SD3	DRQ2	LA19	IRQ15
7	SD2	-12 V	LA18	IRQ14
8	SD1	ENDXFR	LA17	DACK0
9	SD0	+12 V	MEMR	DRQ0
10	IOCHRDY	(key)	MEMW	DACK5
11	AEN	SMEMW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

Table A-1 PC/104 Connector Pinout (Continued)

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

SECTION 7 - PRINTER CONNECTOR PINOUT

The printer port shown in Figure A-8 uses a D25 female connector typical of any PC/AT system.

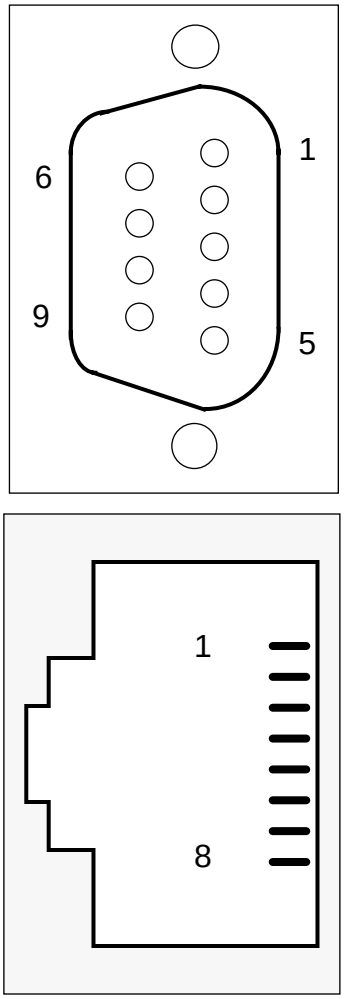


PRINTER CONNECTOR		
PIN	**DIRECTION**	**FUNCTION**
1	In/Out	Data Strobe
2	In/Out	Bidirectional Data D0
3	In/Out	Bidirectional Data D1
4	In/Out	Bidirectional Data D2
5	In/Out	Bidirectional Data D3
6	In/Out	Bidirectional Data D4
7	In/Out	Bidirectional Data D5
8	In/Out	Bidirectional Data D6
9	In/Out	Bidirectional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

Figure A-8 Printer Connector Pinout

SECTION 8 - SERIAL CONNECTOR PINOUT

Each standard RS-232 serial port connectors is either a D9 male as shown in the upper drawing in Figure A-9 or an RJ45 jack like the lower drawing in Figure A-9. The controller boards with RJ45 jacks are supplied with adapters to connect standard D9 serial peripherals. Because only eight of the nine RS-232 signals are brought out to the RJ45 connectors, the signal at pin 8 (pin 1 on the D9 adapter) is assigned to either the DCD signal or the RI signal, depending upon a jumper selection (see Chapter 2 for complete board jumper information).



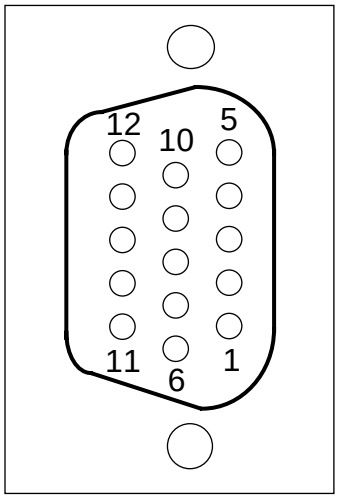
SERIAL COM PORT CONNECTORS				
D9 PIN	**RJ45 PIN**	**DIR**	**RS-232 SIGNAL**	**FUNCTION**
1*	8*	In	DCD	Data Carrier Detect
2	2	In	RX	Receive Data
3	3	Out	TX	Transmit Data
4	4	Out	DTR	Data Terminal Ready
5	1		GND	Signal Ground
6	5	In	DSR	Data Set Ready
7	6	Out	RTS	Request to Send
8	7	In	CTS	Clear to Send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

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

Figure A-9 Serial Connector Pinouts

SECTION 9 - VIDEO CONNECTOR PINOUT

The video port uses a standard high-density D15 VGA connector. Figure A-10 shows the pinout.



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

Figure A-10 Video Connector Pinout

SECTION 10 - VMEbus CONNECTOR PINOUT

Figure A-11 shows the location of the VMEbus P1 and P2 connectors and their orientation. Table A-2 shows the pin assignments for the VMEbus connectors. Note that only Row B of connector P2 is used: all other pins on P2 are reserved and should not be connected.

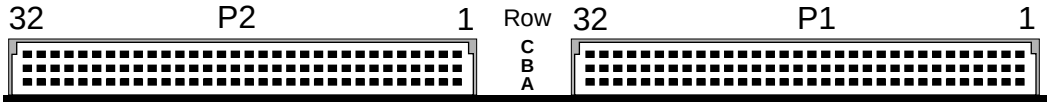


Figure A-11 VMEbus Connector Diagram

Table A-2 VMEbus Connector Pinout

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

Table A-2 VMEbus Connector Pinout (Continued)

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

APPENDIX

B

ETHERNET OPTION

IN THIS APPENDIX:

SECTION 1 - INTRODUCTION	**B-1**
SECTION 2 - ETHERNET SOFTWARE COMPATIBILITY	**B-2**
SECTION 3 - ETHERNET DRIVER SOFTWARE	**B-3**
SECTION 4 - ETHERNET DIAGNOSTIC SOFTWARE	**B-3**
SECTION 5 - TECHNICAL DETAILS	**B-5**

SECTION 1 - INTRODUCTION

The VMIVME-7586 PC/AT compatible VMEbus controller board can be ordered with the Ethernet Mezzanine option. The Ethernet Mezzanine provides local area network access for the VMIVME-7586 controller by means of a standard Ethernet AUI port on the controller's front panel. A customer-supplied AUI adapter connects to the Ethernet AUI port to provide the final interface to the physical network, which can include 10Base5 (Thick Coax), 10Base2 (Thin Coax), or 10BaseT (Twisted Pair). The Ethernet Mezzanine provides an EPROM socket that allows the controller to boot directly from the network with the proper software.

Figure B-1 on page B-2 shows the factory-installed Ethernet Mezzanine on the VMIVME-7586 controller board. Note that the Ethernet Mezzanine AUI connector on the front panel of a VMIVME-7586 is not functional unless the Ethernet Mezzanine is installed.

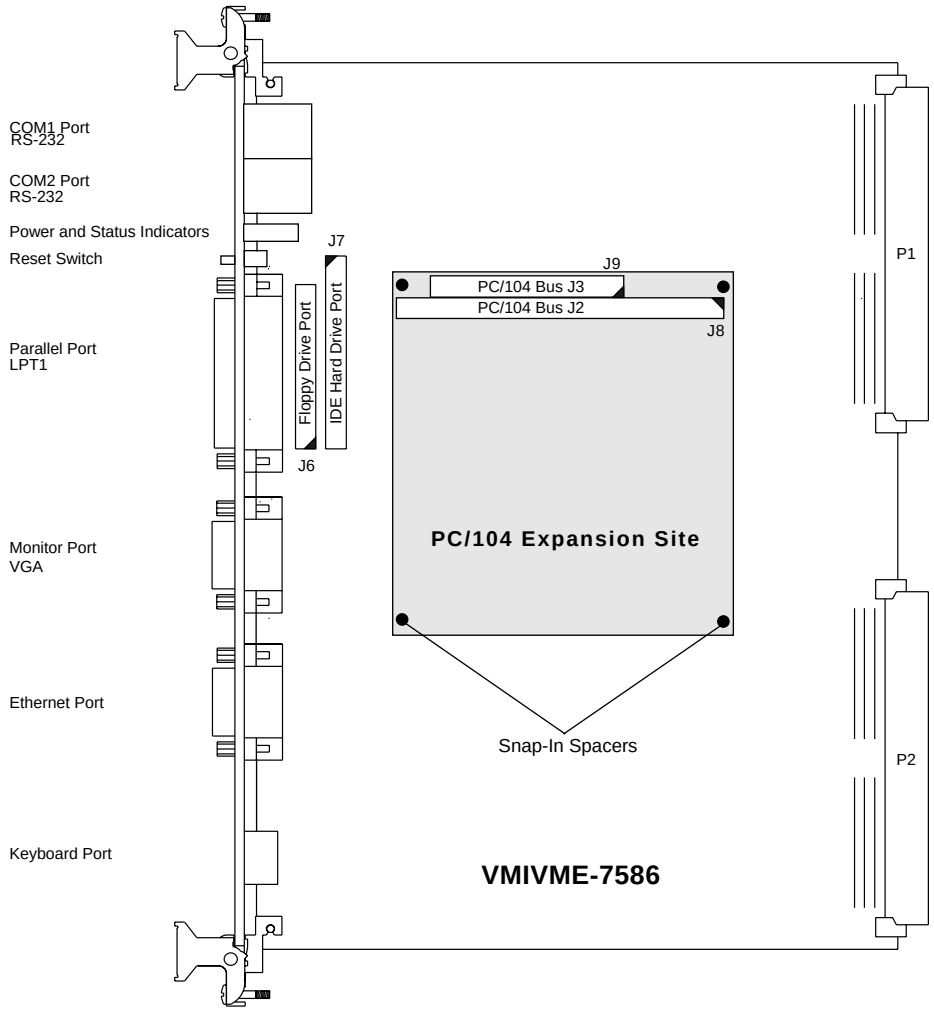


Figure B-1 Location of the Ethernet Mezzanine

SECTION 2 - ETHERNET 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-7586 with the Ethernet Mezzanine installed.

SECTION 3 - ETHERNET DRIVER SOFTWARE

Customers must supply their own driver software for use with the Ethernet Mezzanine. The Ethernet Mezzanine supports the following popular driver software:

- Novell Netware 2.15 (IPX)
- Novell Netware 3.11 (ODI)
- Microsoft Lan Manager 2.0 (NDIS)
- Microsoft Lan Manager Boot ROM
- FTP Packet Driver
- SCO Unix 386 V5
- NDIS 3.0

SECTION 4 - ETHERNET DIAGNOSTIC SOFTWARE

The Ethernet Mezzanine option is provided with hardware diagnostic software on 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 EPROM size and address, IRQ selection, etc. It is recommended that these factory default settings not be changed under normal circumstances. However, if changes are required, the **ATLES** program can be used to change the default settings.

Test the Ethernet Mezzanine with the NIC Inspector as follows:

1. From the root directory of the floppy disk, enter 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. The only valid choice for the Ethernet Mezzanine interrupt level is 5. There may be some warning messages indicating that level 4 is in use. If so, ignore them and continue.
4. Press the **<Enter>** key.
5. Press and release the **<Alt>** key to select the menu.
6. 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 to the Ethernet Mezzanine settings. These changes are stored into the small, nonvolatile EEPROM.

  Some users may wish to install a Boot EPROM in the 28-pin socket at U1. This previously programmed Boot EPROM then allows 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, type ATLES and press **<Enter>** 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. Then press **<Enter>**.
  4. Highlight `Exit` (using the down arrow key) and press **<Enter>**.
  5. Press the **<F5>** key to begin diagnostic tests.

SECTION 5 - TECHNICAL DETAILS

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

The **ATLES** program allows writing to the EEPROM, which provides the Atlantic IC its operating parameters at powerup. These parameters include the memory location of the Boot EPROM.

Valid selections for the memory occupied by the Boot EPROM on the VMIVME-7586 are listed in Table B-1 on page B-6.

STOP!

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 including a FATAL option.

An addressing conflict with the VGA area prevents 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.

Table B-1 Boot EPROM Address Selection

EPROM SIZE	ADDRESS	BOOT EPROM LOCATION
none		Factory Default (no EPROM installed)
8 or 16 Kbyte	C000	**DO NOT SELECT !!!** (Conflicts with VGA area)
	C400	**DO NOT SELECT !!!** (Conflicts with VGA area)
	C800	8 Kbyte from $C8000 to $C9FFF or 16 Kbyte to $CDFFF
	CC00	8 Kbyte from $CC000 to $CDFFF
	D000	**DO NOT SELECT !!!** (Conflicts with VMEbus Window)
	D400	**DO NOT SELECT !!!** (Conflicts with VMEbus Window)
	D800	**DO NOT SELECT !!!** (Conflicts with VMEbus Window)
	DC00	**DO NOT SELECT !!!** (Conflicts with VMEbus Window)
32 Kbyte	C000	**DO NOT SELECT !!!** (Conflicts with VGA area)
	C800	32 Kbyte from $C8000 to $CFFFF
	D000	**DO NOT SELECT !!!** (Conflicts with VMEbus Window)
	D800	**DO NOT SELECT !!!** (Conflicts with VMEbus Window)

DO NOT SELECT A BOOT EPROM ADDRESS CONFLICTING WITH THE VGA AREA. This produces a nonrecoverable FATAL error.

APPENDIX

C

FLASH MEMORY OPTION

IN THIS APPENDIX:

SECTION 1 - INTRODUCTION	**C-1**
SECTION 2 - PREPARING THE FLASH MEMORY	**C-2**
SECTION 3 - COPYING FILES TO FLASH MEMORY	**C-3**
SECTION 4 - USING FLASH MEMORY AS BOOT DEVICE	**C-4**
SECTION 5 - REPROGRAMMING FLASH MEMORY	**C-4**
SECTION 6 - TECHNICAL DETAILS	**C-4**
SECTION 7 - PROGRAMMING	**C-6**

SECTION 1 - INTRODUCTION

The VMIVME-7586 PC/AT Compatible VMEbus Controller can be ordered with the 2 Mbyte Flash Memory Mezzanine option. The Flash Memory Mezzanine allows the user to configure the VMIVME-7586 as a diskless VMEbus master running software stored in the flash memory.

The Flash Memory Mezzanine emulates a 1.44 Mbyte floppy diskette using VMIC-proprietary Flash BIOS. The user can transfer the programs, control, and batch information from a 1.44 Mbyte floppy diskette (easily formatted with DOS system files) to the Flash Memory Mezzanine. The VMIVME-7586 CPU can then boot from the Flash Memory Mezzanine and perform all PC/AT functions in a single 6U VMEbus slot without a hard or floppy drive.

SECTION 2 - PREPARING THE FLASH MEMORY

The Flash Memory Mezzanine has a jumper that controls write access to the flash memory. This horizontal jumper is located on the side of the Flash Memory Mezzanine as shown in Figure C-1.

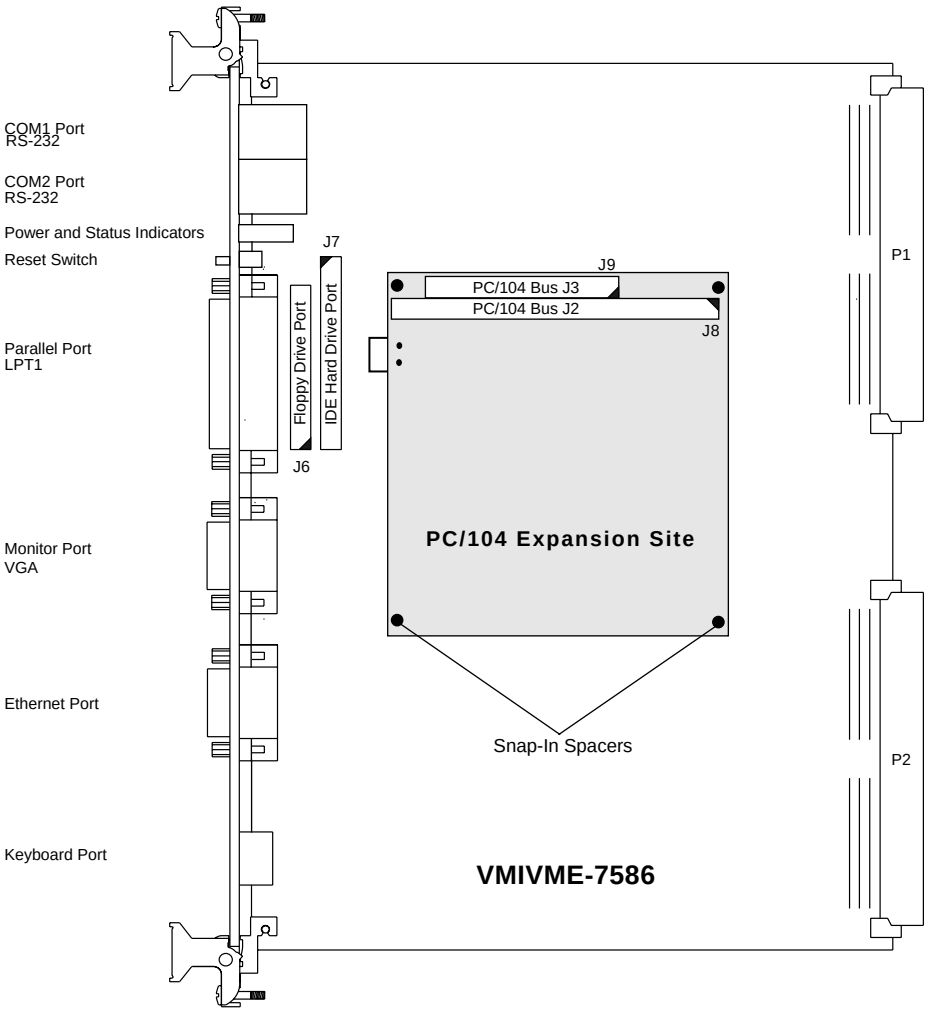


Figure C-1 Flash Mezzanine Jumper Location

Follow the instructions below to prepare the Flash Memory Mezzanine to act as the boot device:

1. Install the Program Enable jumper to allow the flash memory to be programmed.
2. Attach a 3.5 inch floppy drive (and also possibly a hard drive) to the VMIVME-7586 in the VMEbus chassis. Follow the instructions appropriate for the hard/floppy drive module.
3. Apply power to the chassis and run the BIOS Setup program (see the section regarding BIOS Setup in Appendix D for details).
4. Perform whatever actions are necessary in the BIOS Setup program to configure the system to the devices required by the user.
5. Save the new configuration to CMOS and reboot the VMIVME-7586.
6. Observe DOS boot up. The Flash BIOS should print out the following message during boot:

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

SECTION 3 - COPYING FILES TO FLASH MEMORY

1. Prepare the Flash Memory Mezzanine for programming according to the previous procedure.
2. Format a bootable floppy diskette. Use the DOS format command with the /S option command to format a 1.44 Mbyte diskette to install the **IO.SYS** and **MSDOS.SYS** files.

```
FORMAT A: /S
```

3. Copy **CONFIG.SYS**, **AUTOEXEC.BAT**, and any other user files onto the newly-formatted 1.44 Mbyte floppy diskette. Note: the flash will be seen as a 1.44 Mbyte drive A.
4. Remove the bootable floppy diskette.
5. The VMIVME-7586 with 2 Mbyte Flash Memory option is shipped with a 3.5 inch diskette containing the programming utility **PFLASH.EXE**. Insert the VMIC diskette in the floppy drive and execute **PFLASH.EXE**.

   The **PFLASH** program prompts for the user-formatted floppy to be reinserted. Remove the VMIC diskette and reinsert the bootable floppy diskette.
6. Upon successful completion of the flash programming software, remove the floppy from the drive: A.

SECTION 4 - USING FLASH MEMORY AS BOOT DEVICE

1. To configure the VMIVME-7586 as a diskless VMEbus master, first make sure the Flash Memory Mezzanine has system files copied onto it according to the previous procedure.
2. Turn the power off to the VMEbus chassis and remove the floppy and hard drive cables.
3. Apply power to the chassis and run the BIOS Setup program (see the section regarding BIOS Setup in Appendix D for details).
4. Perform whatever actions are necessary in the BIOS Setup program to disable both floppy drives **A** & **B:** and hard drive **C:**. Also, ensure the system boot sequence searches drive **A:** before drive **C:**.
5. If the VMIVME-7586 is to be configured as a diskless VMEbus CPU without a keyboard, select keyboard "Not Installed" under the boot options.
6. Save the new configuration to CMOS and reboot the VMIVME-7586.
7. Remove the Program Enable jumper to write-protect the flash memory.

SECTION 5 - REPROGRAMMING FLASH MEMORY

The Flash Memory Mezzanine may be programmed more than 10,000 times when the **PFLASH** programming utility is exclusively employed. Follow the procedures in these previous sections:

- **PREPARING THE FLASH MEMORY**
- **COPYING FILES TO FLASH MEMORY**
- **USING FLASH MEMORY AS BOOT DEVICE**

Refer to the PROGRAMMING section later in this appendix for information about programming the Flash Memory Mezzanine without using the **PFLASH** utility.

SECTION 6 - TECHNICAL DETAILS

The Flash Memory Mezzanine is an ISA bus peripheral responding to 8 byte in I/O space at I/O $300 through I/O $307, and 32 Kbyte in memory space between $C8000 through $CFFFF for the BIOS.

The Flash BIOS is configured from the factory to be read-only at memory range $C8000 – $CFFFF with a byte-wide data path. All I/O accesses are at 16-bit wide addresses $300, $302, and $304 (even though the field at $302 contains only four valid data bits). All I/O bus accesses to the Flash Memory Mezzanine are zero wait-state.

The Flash Memory Mezzanine bulk nonvolatile storage is I/O mapped using an address pointer and a bidirectional data port. The registers have the following format:

Lower Address Register (Write-Only word at I/O $300)															
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
16 Least Significant Address Bits															

Upper Address Register (Write-Only word at I/O $302)															
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
X	X	X	X	X	X	X	X	X	X	X	X	A20	A19	A18	A17
4 Most Significant Address Bits															

Flash Data Register (Read-Only word at I/O $304)															
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
XD	XD	XD	XD	XD	XD	XD	XD	XD	XD	XD	XD	XD	XD	XD	XD
16-bit Flash Data															

SECTION 7 - PROGRAMMING

Please refer to the following publications by Intel or AMD for 28F020 programming algorithm information:

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

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

APPENDIX

D

BASIC INPUT / OUTPUT SYSTEM

IN THIS APPENDIX:

SECTION 1 - INTRODUCTION	**D-1**
SECTION 2 - STANDARD FEATURES	**D-1**
SECTION 3 - QUICK SETUP	**D-2**
SECTION 4 - PROGRAM DESCRIPTION	**D-5**
SECTION 5 - PROGRAM MENUS AND MENU ITEMS	**D-8**
SECTION 6 - EXITING THE PhoenixBIOS	**D-23**
SECTION 7 - STATUS AND ERROR MESSAGES	**D-26**

SECTION 1 - INTRODUCTION

This chapter describes how to use the PhoenixBIOS setup program and explains each configuration setting. The menus displayed throughout this manual represent a typical system. The actual menus displayed on your screen depend on the installed hardware and features.

SECTION 2 - STANDARD FEATURES

The PhoenixBIOS setup program contains standard system configuration settings necessary for board operation. Standard BIOS features are listed below:

- Ultra-fast memory testing
- Speed independence (8254 PIT)
- Power-On Self Tests (POST)

- Detection of coprocessor
- Memory caching
- Selection of diskette types
- Selection of fixed-disk type
- User-defined fixed-disk types
- Selection of video display
- Support for servers without keyboard and video
- Support for 2.88 Mbyte diskettes
- Support for extended memory up to 64 Mbyte
- Optional suppression of error messages
- Optional control of key-click
- Last Boot Failed Mode
- NumLock control during bootup
- Automatic configuration of IDE fixed disks

SECTION 3 - QUICK SETUP

If you are already familiar with BIOS setup programs, this section contains summarized procedures necessary to load the default PhoenixBIOS Setup values and configure required system settings. For more detailed descriptions of all the BIOS Setup options, refer to the ***Program Menus and Menu Items*** section later in this chapter.

1. Powerup or reboot the system. The PhoenixBIOS displays the message:

   ```
   Press <F2> to enter SETUP...
   ```

2. Press **<F2>** and the **Main** menu appears (see Figure D-1 on page D-3).

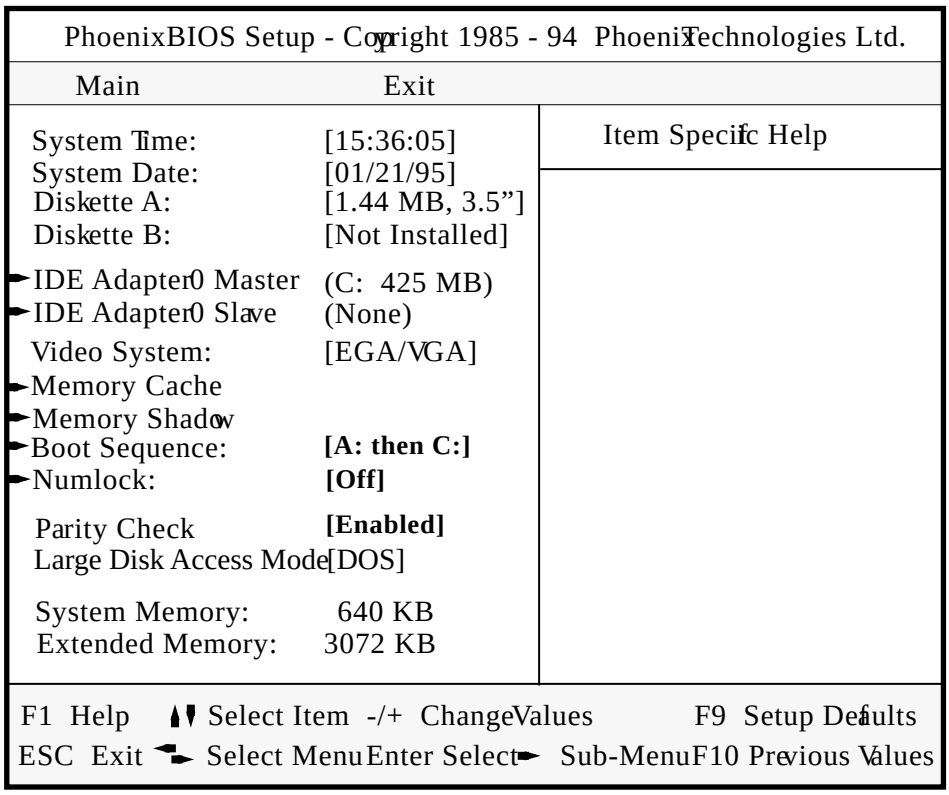


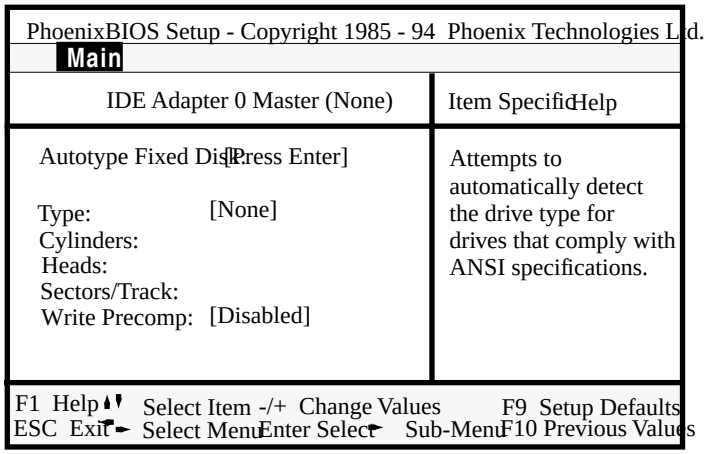
Figure D-1 PhoenixBIOS Opening Display

3. From the **Main** menu, press **<F9> Setup Defaults** and confirm at the prompt by pressing **<Y>** then **<Enter>**.

   Once the defaults are loaded, the **Main** menu is once again displayed.

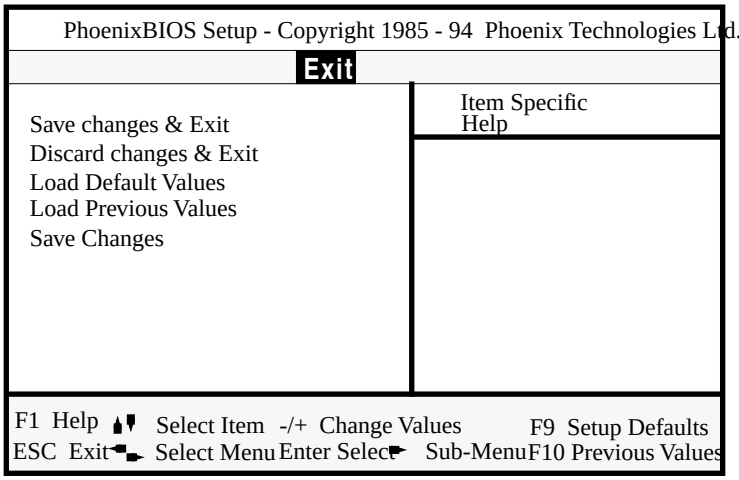
4. Use the **<Tab>** to highlight the **System Time** and **System Date,** then use the **<Shift + Tab>** and **<-/+>** keys to set new values.

5. To use the autodetect settings for the primary hard disk, select the **IDE Adapter 0 Master** sub-menu and press **<Enter>**. The **IDE Adapter 0 Master** sub-menu displays.



6. Select **Autotype Fixed Disk** and press **<Enter>**. The BIOS automatically detects parameters for the hard drive and lists them.
7. Check the parameters listed in the **IDE Adapter 0 Master** sub-menu and make any necessary corrections if the hard disk was not properly autodetected.
8. Press **<Esc>** to return to the **Main** menu.
9. If you have a secondary drive, select the **IDE Adapter 0 Slave** sub-menu and press **<Enter>**. The **IDE Adapter 0 Slave** sub-menu displays.
10. Select **Autotype Fixed Disk** and press **<Enter>**. The BIOS automatically detects parameters for the hard drive.
11. Check the parameters listed in the **IDE Adapter 0 Slave** sub-menu and make necessary changes if the disk was not properly autodetected.
12. Press **<Esc>** to return to the **Main** menu.
13. From the **Main** menu, configure the floppy drive type(s). The type of drive must be properly entered or boot errors will occur.
14. To save your selections, move to the **Exit** menu and press **<Enter>**.

The **Exit** menu is displayed.



15. Select the **Save Changes & Exit** item and confirm.

16. The VMIVME-7586 automatically reboots with the new values in effect.

If any error messages are displayed, refer to the ***Status and Error Messages*** section later in this chapter for a description of the problem.

SECTION 4 - PROGRAM DESCRIPTION

When the system is powering up, the PhoenixBIOS displays the message:

```
Press <F2> to enter SETUP...
```

Press **<F2>** to start the program and display the **Main Menu** as shown in Figure D-3 on page D-8.

The PhoenixBIOS setup program consists of two menus, the **Main Menu** and **Exit Menu**.

- The **Main Menu** contains items and sub-menus to configure standard system components such as the time and date, floppy/hard disk drives, video adapter type, and keyboard type.
- The **Exit Menu** contains items to save updated item values, load previous or default values, and discard any changes.

This section includes a description of the program's menu displays and explains how to use the control keys to set up items.

USER INTERFACE

The **Main Menu**, shown below, is the opening display. Using the following menu display as an example, this section describes screen elements in both the **Main** and **Exit** menus..

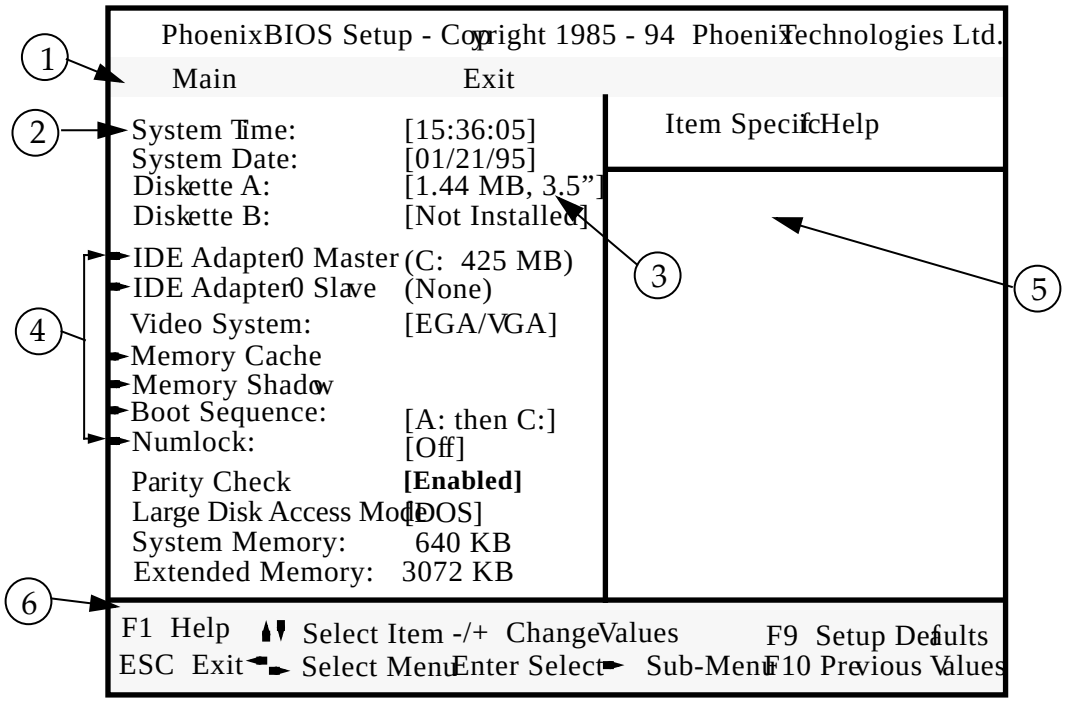


Figure D-2 Elements of the PhoenixBIOS User Interface

The numbered screen elements are described below:

1. The two top-level menus are displayed on the menu bar at the top of the display. The highlighted menu is the active menu.
2. All available menu items and sub-menus are displayed.
3. Menu items contain a value field, enclosed by brackets "[]", where you can either key in the new values or use the **<-/+>** keys.
4. Arrows indicate sub-menus. Move to the sub-menu and press **<Enter>** to access advanced settings.
5. The Help Window provides a brief description of each selected menu item.

6. Keystroke control summary in the legend bar shows all keystrokes necessary to move through the program's menu system. The next section, ***Control Key Summary,*** explains the function of each key.

CONTROL KEY SUMMARY

Use the following keys to move through the program and modify program values:

Table D-1 PhoenixBIOS Setup Keystroke Actions

KEYSTROKE	ACTION
<Esc>	Returns to previous screen (or exits from **Main Menu)**.
<Enter>	Moves to a selected sub-menu. Also used to initiate autodetect function in the **IDE Adapter Master/Slave** sub-menus.
Left, Right Arrow Keys	Moves the cursor from one menu to the next.
Up, Down Arrow <↑/↓> Keys	Moves the cursor from one menu item to the next.
Plus, Minus <-/+> Keys	Changes the value of a selected menu item.
<F1> Help	Displays the general help file.
<F9> Setup Defaults	Loads the PhoenixBIOS factory default settings.
<F10> Previous Values	Restores the values previously saved in CMOS.

Keystroke operations from menu to sub-menu remain the same. To select an item and change its value:

1. Use the **<↑/↓>** keys to move the cursor to the desired menu item.
2. Then use the **<Tab>** and **<Shift+Tab>** keys to move to the value field, which is enclosed by brackets.
3. To change values, use the **<+/->** keys and press **<Enter>** when done.

NOTE:

THE DEFAULT VALUE FOR <F9> AND <F10> IS ALWAYS "N". TO EXECUTE THESE ITEMS, PRESS <ENTER> TO CONTINUE.

SECTION 5 - PROGRAM MENUS AND MENU ITEMS

MAIN MENU

The **Main Menu** (Figure D-3) contains items and sub-menus to configure system components such as the time and date, floppy/hard disk drives, video adapter type, and keyboard type. It also contains sub-menus, indicated by arrows, to access advanced system settings.

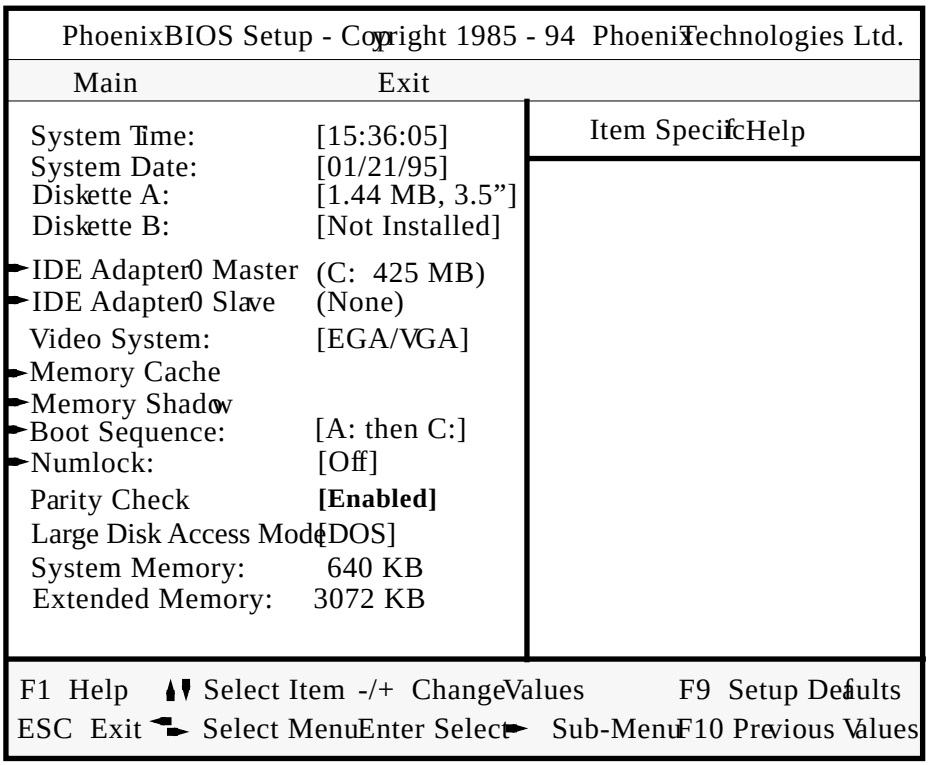


Figure D-3 PhoenixBIOS Main Menu

To access a sub-menu from the **Main Menu**, select the desired sub-menu and press **<Enter>**. To exit sub-menus, press **<Esc>**.

Main Menu items are described below along with the available values. Sub-menus are described in the following sections.

NOTE:

FACTORY DEFAULT SETTINGS FOR MENU ITEMS ARE DENOTED BY AN ASTERISK (*).

System Time

Sets the internal system clock to **hour:minutes:seconds**.

System Date

Sets the internal system date to include the **month/date/year**.

Diskette A:/B:

Configures floppy drives **A:** and **B:** to the appropriate disk size. Select one of the following 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**

2.88 MB 3-1/2 inch

Not Installed - used for diskless workstations

Video System

Sets the type of video monitor. The following settings provide compatibility to any standard video monitor:

Monochrome

***EGA/VGA**

Color 80 x 25

Large Disk Mode

Selects the operating system:

***DOS**

Other - select this option if you have an operating system other than DOS (such as, UNIX).

System Memory/ Extended Memory

Reports the system's available base and extended memory. **This is not a user-configurable item.** The PhoenixBIOS automatically senses and reports system memory. Memory is reported in 64 Kbyte increments. The BIOS reports up to 640 Kbyte in the **Base Memory** field and 65,472 Kbyte in the

Extended Memory field. Extended memory is addressed from just above the 1 Mbyte Real Mode boundary to the 64 Mbyte level.

IDE Adapter 0 Master/IDE Adapter 0 Slave Sub-menus

The IDE adapters control the hard disk drives. The PhoenixBIOS supports up to two IDE adapters; one master drive and one optional slave drive. This section describes the hard disk configuration settings for both sub-menus.

To access these sub-menus:

From the **Main Menu**, select the desired sub-menu and press **<Enter>**. Figure D-4 below shows the **IDE Adapter 0 Master** sub-menu.

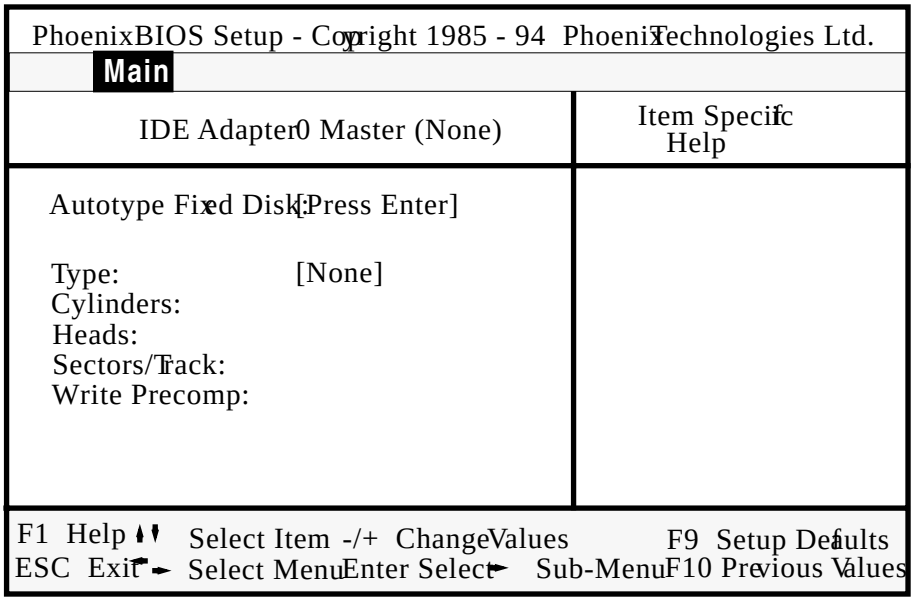


Figure D-4 PhoenixBIOS IDE Adapter Master Sub-menu

The menu items available in both sub-menus are described below.

Autotype Fixed Disk

Automatically determines the parameters, **Type, Cylinders, Heads, Sectors/Track** and **Write Precomp**, for the existing hard drive. The BIOS displays the determined parameters and requests user confirmation. If confirmed, the drive parameters are saved as listed.

Type

Sets the disk type if you do not use the **Autotype Fixed disk** feature. You can select a predefined fixed disk type or manually configure all drive parameters.

User - user-defined disk parameters. Set all disk parameters.

CAUTION! Incorrect settings can cause your system to malfunction. If you are not sure of the disk parameters, use the **Autotype Fixed disk** menu item.

1 to 39 - selects a drive type. Each fixed-disk drive supports 39 predefined drive types. Table D-2 on page D-12 lists these predefined types and their default values. When you select a drive type, the remaining parameters are automatically set.

Table D-2 PhoenixBIOS Fixed Disk Table

Type	Cylinders	Head	Sectors	Wrt Pre
1	306	4	17	Should always be set to None.
2	615	4	17	
3	615	6	17	
4	940	4	17	
5	940	6	17	
6	615	4	17	
7	462	8	17	
8	733	5	17	
9	900	15	17	
10	820	3	17	
11	855	5	17	
12	855	7	17	
13	306	8	17	
14	733	7	17	
15	Reserved			
16	612	4	17	
17	977	5	17	
18	977	5	17	
19	1024	7	17	
20	733	5	17	
21	733	7	17	
22	733	5	17	
23	306	4	17	
24	612	4	17	
25	612	2	17	

Table D-2 PhoenixBIOS Fixed Disk Table (Continued)

Type	Cylinders	Head	Sectors	Wrt Pre
26	614	4	17	
27	820	6	17	
28	977	5	17	
29	1218	15	36	
30	1224	15	17	
31	823	10	17	
32	809	6	17	
33	830	7	17	
34	830	10	17	
35	1024	5	17	
36	1024	8	17	
37	615	8	17	
38	925	9	17	
39	925	9	17	

Cylinders

Sets the number of cylinders:

1 to 2048

Heads

Sets the number of read/write heads:

1 to 16

Sectors/Track

Sets the number of sectors per track:

1 to 64

Write Precomp

Sets the number of the cylinder at which to change the write timing:

1-2048

None - should always be selected for IDE hard disks.

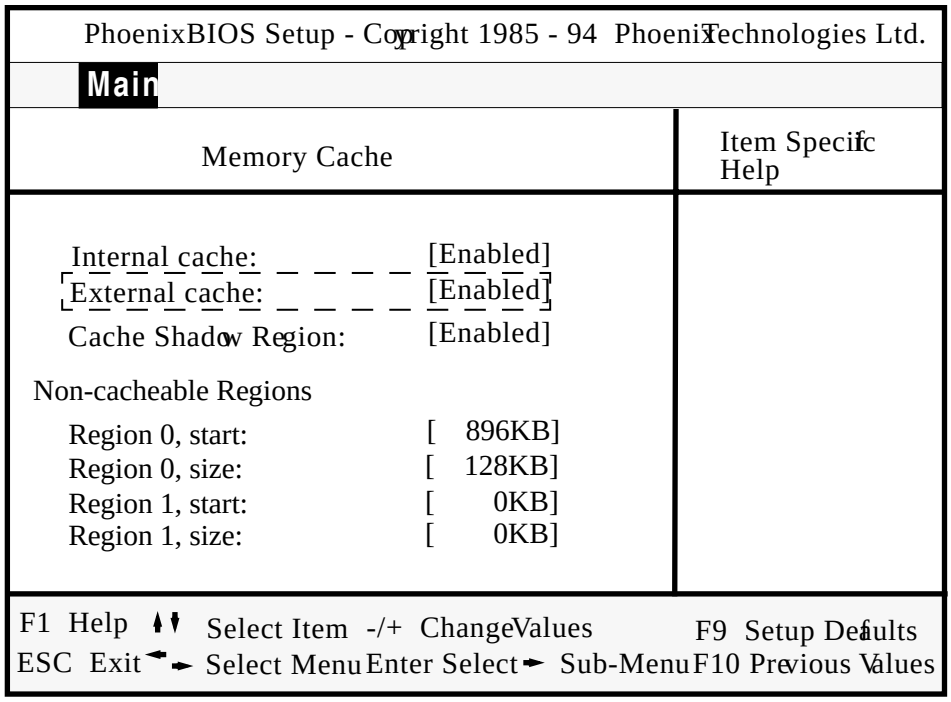
MEMORY CACHE SUB-MENU

The Memory Cache sub-menu contains items that enable caching and the cache shadowing. Memory cache is a special storage area of static RAM (SRAM). The cache shadow is a specified portion of dynamic RAM (DRAM). Both of these options provide faster data access for the CPU.

To use memory cache features:

From the **Main Menu**, select **Memory Cache**. Press **<Enter>**.

The **Memory Cache** sub-menu appears as shown in Figure D-5.



[ ] This menu item appears only on controllers with an external cache option.

Figure D-5 PhoenixBIOS Memory Cache Sub-menu

The items contained in this sub-menu are described along with their available options.

NOTE:

THE FACTORY DEFAULT SETTINGS FOR MENU ITEMS ARE DENOTED BY AN ASTERISK (*).

Internal Cache

Controls the processor's internal eight Kbyte cache memory:

***Enabled - causes data contained in the main memory to be copied into the cache where it can be processed faster.**

Disabled - turns off internal cache.

External Cache

This menu item is displayed only on controllers with an external cache option. Controls the external cache:

***Enabled - causes data to be copied into external cache.**

Disabled - turns off the external cache.

NOTE:

IF CACHE ITEMS ARE DISABLED, THE SPEED PERFORMANCE OF YOUR SYSTEM WILL BE DRASTICALLY REDUCED.

Cache Shadow Region

Controls the cache shadow:

***Enabled - uses the cache shadow region defined in the Shadow Memory Region item in the Memory Shadow sub-menu** (see Figure D-6 on page D-17)**.**

Disabled - turns off cache shadow.

Noncacheable Region

Specifies the following areas of regular and extended memory as noncacheable regions:

Region 0, start:

Defines the start of noncacheable **Region 0** in multiples of 64 Kbyte. The start point is the offset from the beginning of memory. The factory default is **896 Kbyte.**

Region 0, size:

Multiples of 16 - defines the size of noncacheable region 0 in Kbyte. The factory default is **128 Kbyte.**

Disabled - makes this region available for cache.

Region 1, start:

Defines the start of noncacheable **Region 1** in multiples of 64 Kbyte. The start point is the offset from the beginning of memory. The factory default is **0 Kbyte**.

Region 1, size:

Multiples of 16 - defines the size of noncacheable **Region 1** in Kbyte. The default setting is **0 Kbyte**.

Disabled - makes this region available for cache.

MEMORY SHADOW SUB-MENU

Memory shadow is a technique for relocating data from slower ROM to faster dynamic RAM (DRAM). The Memory Shadow sub-menu contains items that allow you to enable video shadowing and designate the system's shadow memory regions.

To access the **Memory Shadow** sub-menu:

From the **Main Menu**, select **Memory Shadow** and press **<Enter>**.

The **Memory Shadow** sub-menu appears (Figure D-6 on page D-17).

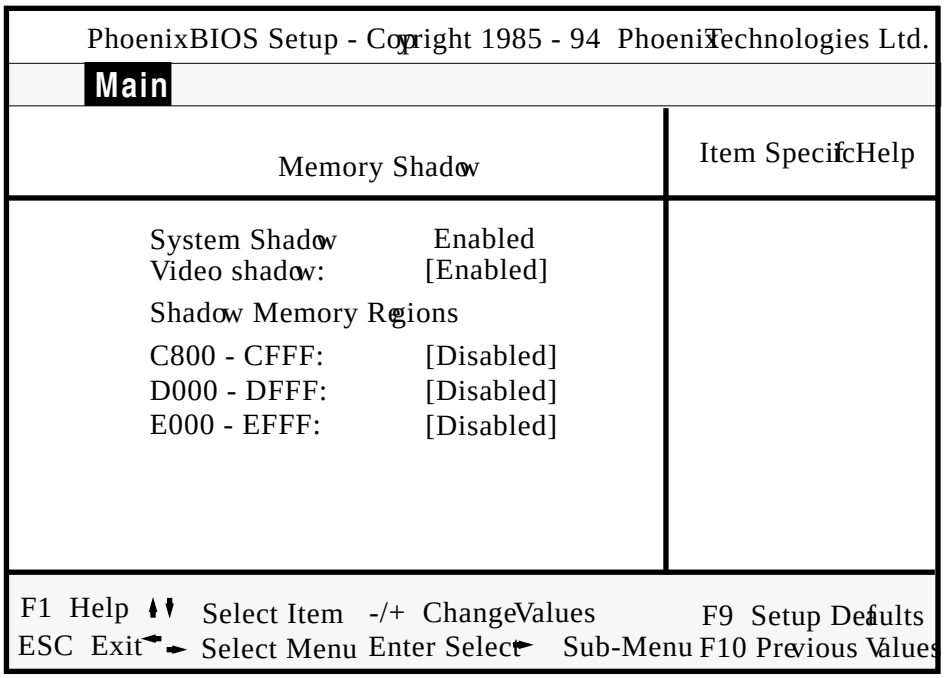


Figure D-6 PhoenixBIOS Memory Shadow Sub-menu

The items contained in this sub-menu are described below along with their available options.

NOTE:

THE FACTORY DEFAULT SETTINGS FOR MENU ITEMS ARE DENOTED BY AN ASTERISK (*).

System Shadow

Permanently enabled.

Video Shadow

Shadows video BIOS.

***Enabled**

Disabled

Shadow Memory Regions

Defines the ROM shadow regions:

C800 - CFFF

Enabled

***Disabled**

D000 - DFFF

Enabled

***Disabled**

E000 - EFFF

Defines the VMEbus access, which should *not* be shadowed. This setting should always be disabled.

Enabled

***Disabled**

BOOT OPTIONS SUB-MENU

The Boot Options sub-menu contains items that enable you to configure system boot-up operations.

From the **Main Menu**, select **Boot Sequence** and press **<Enter>**.

The **Boot Options** sub-menu appears (Figure D-7 on page D-19).

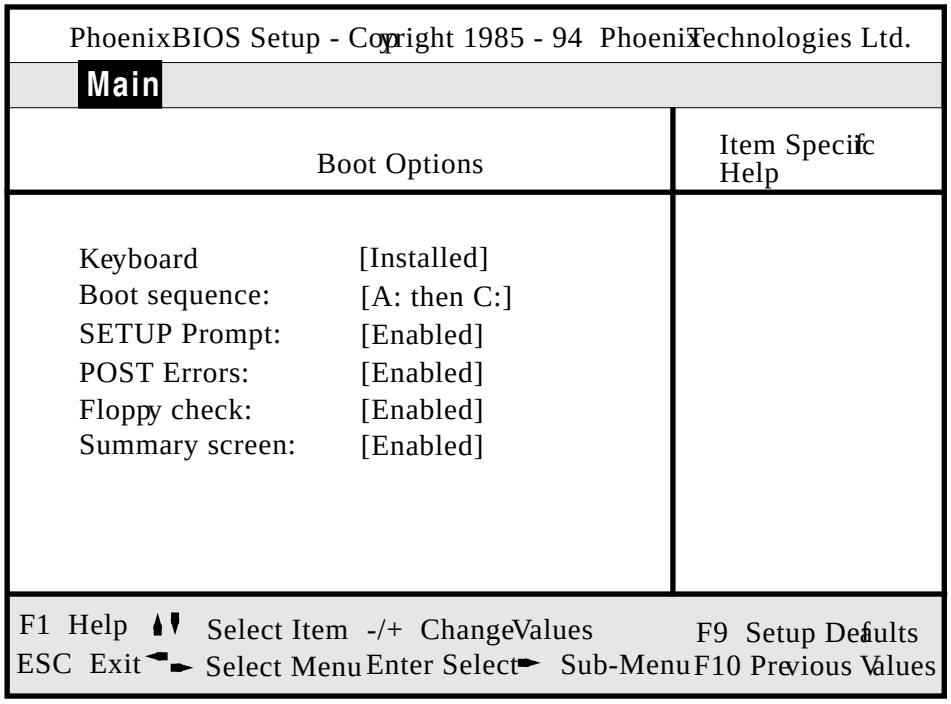


Figure D-7 PhoenixBIOS Boot Options Sub-menu

The following section provides descriptions of menu items and their corresponding options.

NOTE:

THE DEFAULT SETTINGS FOR MENU ITEMS ARE DENOTED BY AN ASTERISK (*).

Keyboard

Enables the system to boot without an installed keyboard.

***Installed - system boots with a keyboard installed.**

Uninstalled -system boots without a keyboard.

Boot Sequence

Sets the system's boot drive.

***A: then C: - system attempts to boot from floppy drive A: then hard drive C:.**

C: then A: - system attempts to boot from hard drive **C:** then floppy drive **A:**.

C: only - system attempts to boot from drive **C:** only.

SETUP Prompt

Controls the display of the BIOS program's **Press <F2> to run Setup...** prompt from appearing when the system boots.

***Enabled - displays the prompt.**

Disabled - does not display the prompt; however, the system still responds to the **<F2>** key request for setup. This option provides a little faster boot process.

POST Errors

Controls the display of the SETUP prompt, **Press <F1> to resume, <F2> to Setup** when a Power-On Self Test (POST) error occurs. POST is a series of system diagnostic routines performed during system boot.

***Enabled - displays the SETUP prompt along with the appropriate error message.**

Disabled - If the **SETUP Prompt** menu item is **Disabled**, the BIOS displays the appropriate error message but does not display the program's setup prompt. The POST routines continue to run until completed.

For help in resolving POST problems, all PhoenixBIOS error and status messages are contained in the ***List of Messages*** section later in this chapter.

Floppy Check

Seeks diskette drives during bootup. Disabling speeds boot time and increases the longevity of the diskette drives.

***Enabled**

Disabled

Summary Screen

Controls display of the system configuration screen (Figure D-8) during system start-up.

Enabled

Disabled

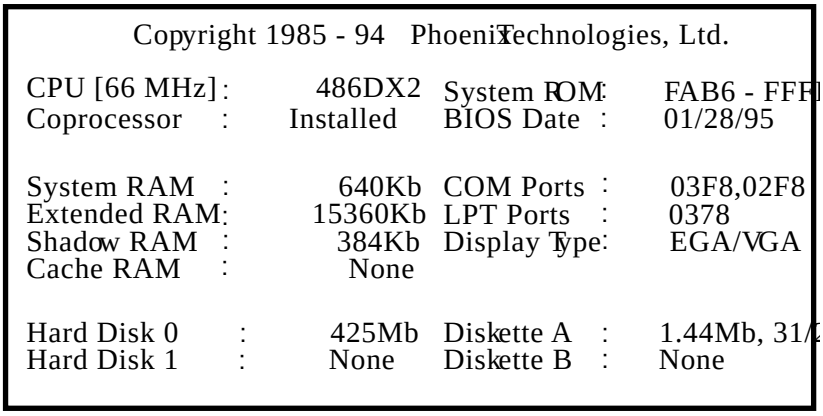


Figure D-8 PhoenixBIOS Example Summary Screen

KEYBOARD FEATURES

The **Keyboard Features** sub-menu contains menu items that enable you to set keyboard functionality.

From the **Main Menu**, select **Numlock** and press **<Enter>**.

The **Keyboard Features** sub-menu appears (Figure D-9).

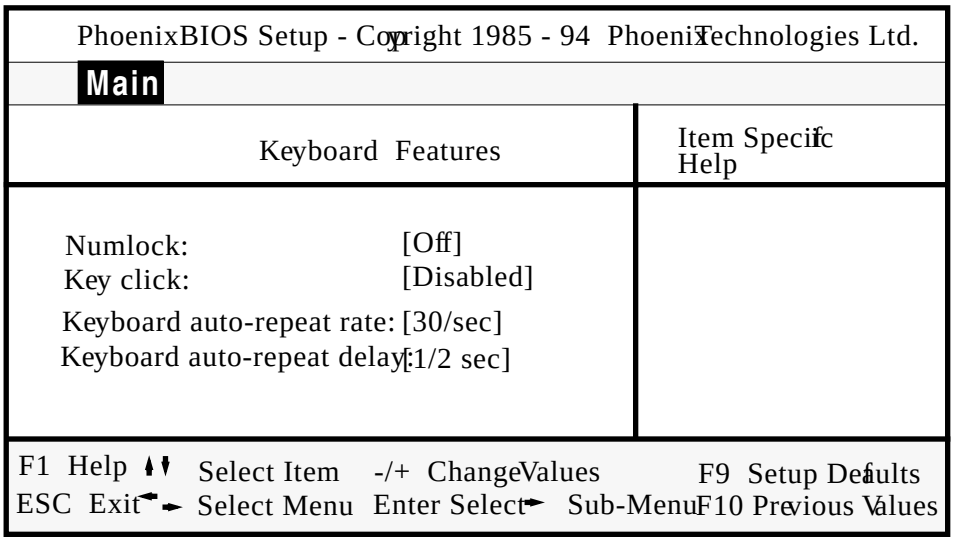


Figure D-9 PhoenixBIOS Keyboard Features Sub-menu

Menu items contained in this sub-menu are described below along with their available options.

NOTE:

THE DEFAULT SETTINGS FOR MENU ITEMS ARE DENOTED BY AN ASTERISK (*).

NumLock

Sets the **<NumLock>** keyboard function:

on

***off -** **causes the numeric keypad arrow keys to serve as cursor movement keys.**

Auto - turns **<NumLock>** on if it finds a numeric key pad.

Key click

Controls audible key click.

***Enabled**

Disabled

Keyboard Autorepeat Rate

Controls the speed at which a keystroke is repeated per second. The possible values for this item include:

2, 6, 10, 13.3, 18.5, 21.8, 26.7, and *__30__ characters per second.

Keyboard Autorepeat Delay

Controls the delay time between keystrokes. The possible delay values include:

1, 3/4, 1/2, and *__1/4__ milliseconds.

SECTION 6 - EXITING THE PhoenixBIOS

EXIT MENU

The **Exit Menu** contains items that allow you to save any changed settings, reload previous or default settings, and exit the program.

To access the **Exit Menu**, use the left/right arrow keys to select from the menu bar - OR - press **<Esc>** from the **Main Menu**.

The **Exit Menu** appears (Figure D-10).

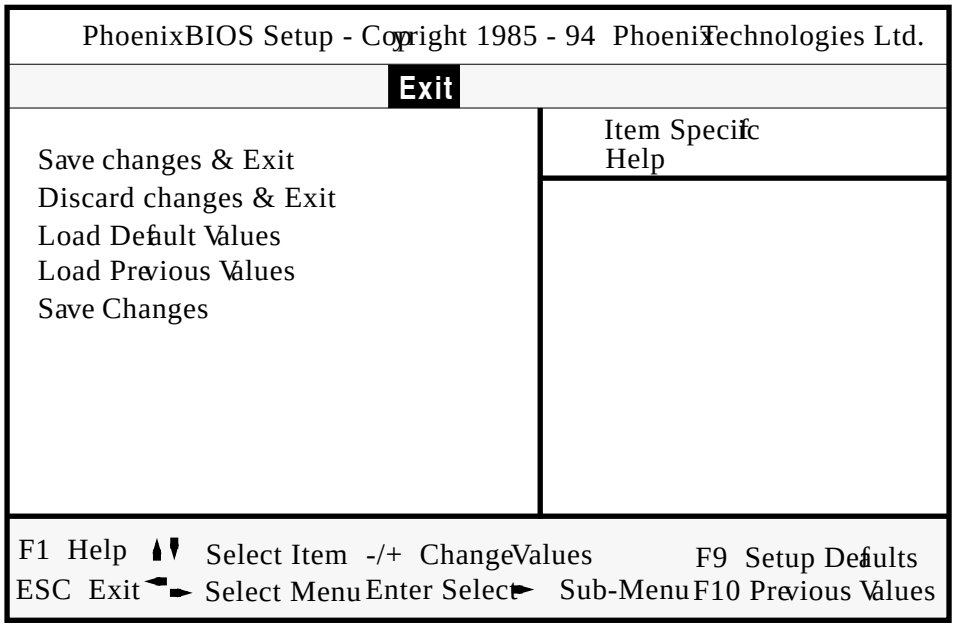


Figure D-10 PhoenixBIOS Exit Menu

Menu items contained in the **Exit Menu** are described below. To use Exit items, highlight the desired item and press **<Enter>**.

Save Changes & Exit

Stores selected values, exits the setup program, and reboots the system. The new menu items, stored in CMOS (battery-backed CMOS RAM), are used.

Discard Changes & Exit

Exits the program without saving any menu settings. At system boot up, the previous values are used.

Load Default Values

Resets the program to use its original factory values. The program displays the message:

Notice

Default Values have been loaded!

[Continue]

Press **<Enter>** to continue or **<Esc>** to cancel.

This option works as a backup in case the BIOS program detects a problem in the integrity of user-selected CMOS values. During bootup, the BIOS displays these messages:

System CMOS checksum bad - run SETUP

Press <F1> to resume, <F2> to Setup

Press **<F2>** to run the setup program and select the **Get Default Values** item to return to the factory defaults. The only CMOS variables not reset are the date, time, and drive types on the **Main Menu**.

Load Previous Values

Resets all menu items to values previously set in CMOS.

Select this item and press **<Enter>** to reload the menu items. Select **Y** at the confirmation prompt to continue; **N** to return to the **Exit Menu** without retrieving previous values.

Save Changes

Stores selected menu items in CMOS without exiting the program. You can return to the other menu if you want to review and change your selections.

Select this item and press **<Enter>.** Press **Y** at the confirmation prompt to save selections and return to the **Main Menu**; **N** to return to the **Exit Menu** without saving.

SECTION 7 - STATUS AND ERROR MESSAGES

Table D-3 describes status and error messages you may encounter when using the BIOS setup program. Messages are listed in alphabetical order.

If your system fails after you made changes in the setup program, you may be able to correct the problem by entering the program and restoring the factory defaults.

Table D-3 PhoenixBIOS Status and Error Messages

Message	Description/User Action
nnnn Cache SRAM Passed	Where *nnnn* is the amount of system cache in Kbyte successfully tested.
Diskette drive A error -or- Diskette drive B error	Drive A: or B: is present but fails the BIOS POST diskette tests. Check to see that the drive is defined with the proper diskette type in the setup program and that the diskette drive is attached correctly.
Entering SETUP...	Starting setup program.
Extended RAM Failed at offset:*nnnn*	Extended memory not working or not configured properly at offset *nnnn*.
nnnn Extended RAM Passed	Where *nnnn* is the amount of RAM in Kbyte successfully tested.
Failing Bits:*nnnn*	The hex number *nnnn* is a map of the bits at the RAM address (in System, Extended, or Shadow memory) which failed the memory test. Each one in the map indicates a failed bit.
Fixed Disk 0 Failure or Fixed Disk 1 Failure or Fixed Disk Controller Failure	The fixed disk is not working or not configured correctly. Check to see if the fixed disk is attached properly. Run the setup program to make sure the fixed disk type is correctly identified.
Incorrect Drive A type - run SETUP	Floppy drive A: type not correctly identified in the setup program.
Incorrect Drive B type - run SETUP	Floppy drive B: type not correctly identified in the setup program.
Invalid NVRAM media type	There is a problem with NVRAM (CMOS) access.
Keyboard controller error	The keyboard controller failed testing. Check the keyboard and keyboard controller.

Table D-3 PhoenixBIOS Status and Error Messages (Continued)

Message	Description/User Action
`Keyboard error`	Keyboard is not working. Check keyboard and cable connections.
`Keyboard error nn`	The BIOS discovered a “stuck” key and displays the scan code *nn* identifying the problem key.
`Keyboard locked - Unlock key switch`	Unlock the system to proceed.
`Monitor type does not match CMOS - Run SETUP`	The monitor type was not correctly identified in the setup program.
`Operating system not found`	The operating system cannot be located on either drive A: or drive C:. Enter the setup program and see if the fixed disk and drive A: are properly identified.
`Parity Check 1`	A parity error was found in the system bus. BIOS attempts to locate the address and display it on the screen. If it cannot locate the address, it displays “*????*”.
`Parity Check 2`	A parity error found in the I/O bus. BIOS attempts to locate the address and display it on the screen. If it cannot locate the address, it displays “*????*”.
`Press <F1> to resume, <F2> to Setup`	Displayed after any recoverable error message. Press <F1> to start the boot process or <F2> to enter the setup program and change any settings.
`Press <F2> to enter SETUP`	Optional message displayed during POST. It can be turned off in the setup program.
`Previous boot incomplete - Default configuration used`	Previous POST did not complete successfully. POST loads default values and offers to run the setup program. If the failure was caused by incorrect values and they are not corrected, the next boot will likely fail. On systems with control of wait states, improper setup settings can also terminate POST and cause this error on the next boot. Run the setup program and verify that the wait-state configuration is correct. This error is cleared the next time the system is booted.
`Real time clock error`	The real-time clock fails the BIOS test. It may require board repair.

Table D-3 PhoenixBIOS Status and Error Messages (Continued)

Message	Description/User Action
`Shadow RAM Failed at offset: nnnn`	Shadow RAM failed at offset *nnnn* of the 64 Kbyte block at which the error was detected.
nnnn `Shadow RAM Passed`	Where *nnnn* is the amount of shadow RAM in Kbyte successfully tested.
`System battery is dead - Replace and run SETUP`	The CMOS clock battery indicator shows the battery is dead. Replace the battery and run the setup program to reconfigure the system.
`System BIOS shadowed`	System BIOS copied to shadow RAM.
`System cache error - Cache disabled`	RAM cache failed the BIOS test. BIOS disabled the cache.
`System CMOS checksum bad - run SETUP`	System CMOS has been corrupted or modified incorrectly, perhaps by an application program that changes data stored in CMOS. Run the setup program and reconfigure the system either by getting the Default Values and/or making your own selections.
`System RAM Failed at offset: nnnn`	System RAM failed at offset *nnnn* in the 64 Kbyte block at which the error was detected.
nnnn `System RAM Passed`	Where *nnnn* is the amount of system RAM in Kbyte successfully tested.
`System timer error`	The timer test failed. This requires repair of the system board.
`UMB upper limit segment address: nnnn`	Displays the address *nnnn* of the upper limit of Upper Memory Blocks, indicating released segments of the BIOS which may be reclaimed by a virtual memory manager.
`Video BIOS shadowed`	The video BIOS successfully copied to shadow RAM.