VMICPCI-7699

Single-Slot Pentium II/III Embedded Module CompactPCI Peripheral SBC with High Availability, Hot Swap and Dual PMC

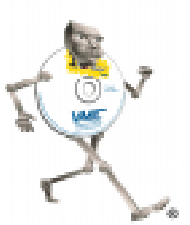
Product Manual

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

500-657699-000 Rev. B

COPYRIGHT AND TRADEMARKS

(I/O man figure)



(IOWorks man figure)

Table Of Contents

List Of Figures

List Of Tables

Overview

Introduction

VMIC's VMICPCI-7699 is a complete Intel Pentium II/III-compatible Pentium II Embedded Module processor-based computer with the additional benefits of Eurocard construction and full compatibility with the CompactPCI Specification Rev. 2.1. The VMICPCI-7699 with advanced CompactPCI interface and SDRAM that is dual-ported to the CompactPCI bus, is ideal for CompactPCI peripheral slot applications.

The single-slot CPU board functions as a standard Intel Pentium II/III, executing a power-on self-test, then boots up Windows 98 SE, or Windows NT. The standard features of the VMICPCI-7699 are discussed in Chapter 3 of this manual.

The VMICPCI-7699 operates as a CompactPCI peripheral slot CPU and interacts with other CompactPCI modules via the on-board embedded bridge. The VMICPCI-7699 is also compatible with the CompactPCI Hot Swap specification PICMG 2.1 R1.0 and supports both Full Hot Swap and High Availability Hot Swap. To enhance its use as a peripheral slot CPU, the VMICPCI-7699 includes an I^2C controller and supports geographical addressing. These features are discussed in Chapter 4 of this manual.

The VMICPCI-7699 programmer may quickly and easily control CompactPCI bus functions simply by linking to a library of interrupt and control functions. This library is available with VMIC's VMISFT-9421 IOWorks Access software for Windows NT users.

The VMICPCI-7699 also provides Embedded PC/RTOS features including general-purpose timers, a programmable Watchdog Timer, a bootable flash disk system, remote LANboot, and non-volatile SRAM. The board also provides for expansion via two PMC sites. These features make the unit ideal for embedded applications and are discussed in Chapter 4 of this manual.

Organization of the Manual

This manual is composed of the following chapters and appendices:

Chapter 1 - VMICPCI-7699 Features and Options describes the features of the base unit followed by descriptions of the associated features of the unit in operation on a CompactPCI bus.

Chapter 2 - Installation and Setup describes unpacking, inspection, hardware jumper settings, connector definitions, installation, system setup, and operation of the VMICPCI-7699.

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

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

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

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

Appendix B - System Driver Software provides details for installing drivers under Windows 98 SE and Windows NT.

Appendix C - Phoenix BIOS describes the menus and options associated with the Phoenix (system) BIOS.

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

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

Appendix F - Sample C Software provides example code to use with the VMICPCI-7699.

References

Pentium II Processor with Mobile Module
Order Number 243668-001
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080

Pentium III Processor with Mobile Module
Order Number 245304-002
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080

Intel 82440BX AGP set: 82443BX Host Bridge/Controller
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119

Intel 21554 Embedded PCI Bridge
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119

Intel 82371EB PCI ISA IDE Xcellerator (PIIX4E)
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119

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

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

ISA & EISA, Theory and Operation

Solari, Edward
Annabooks
15010 Avenue of Science, Suite 101
San Diego, CA 92128 USA
ISBN 0-929392 -15-9

82C54 CHMOS Programmable Internal Timer

Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119

Flash ChipSet Product Manual

SanDisk Corporation
140 Caspian Court
Sunnyvale, CA 94089-9820

DS 1384 Watchdog Timekeeping Controller

Dallas Semiconductor
4461 South Beltwood Pwky.
Dallas, TX 75244-3292

CMC Specification, P1386/Draft 2.0 from:

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

PMC Specification, P1386.1/Draft 2.0 from:

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

Intel 21554 Embedded PCI-to-PCI Bridge for the VMICPCI-7699 CompactPCI CPU

VMIC
12090 South Memorial Parkway
Huntsville, Alabama 35803-3308
(800) 322-3616
www.vmic.com

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

CompactPCI Specification PICMG 2.0 R2.1

PCI Industrial Computer Manufactures' Group
301 Edgewater Place
Suite 220
Wakefield, MA 01880
(617) 224-1100
(617) 224-1239 (FAX)
www.picmg.org

The following is useful information related to remote ethernet booting of the VMICPCI-7699:

Microsoft Windows NT Server Resource Kit

Microsoft Corporation
ISBN: 1-57231-344-7
www.microsoft.com

The following is useful information related to the operation of the I^2C controllers:

PCF8584 I^2C-bus Controller

Philips Semiconductor
811 East Arques Ave.
Sunnyvale, CA 94088-3409
(800) 234-7381
www.semiconductors.philips.com

The I^2C Specification version 2.0

Philips Semiconductor
811 East Arques Ave.
Sunnyvale, CA 94088-3409
(800) 234-7381
www.semiconductors.philips.com

Safety Summary

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

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

Ground the System

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

Do Not Operate in an Explosive Atmosphere

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

Keep Away from Live Circuits

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

Do Not Service or Adjust Alone

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

Do Not Substitute Parts or Modify System

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

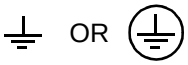
Dangerous Procedure Warnings

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

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

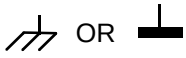
Safety Symbols Used in This Manual

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



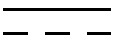
Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.

Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. Before operating the equipment, terminal marked with this symbol must be connected to ground in the manner described in the installation (operation) manual.



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

Alternating current (power line).



Direct current (power line).

Alternating or direct current (power line).

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

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

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

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

CHAPTER 1

Features and Options

Contents

Introduction

The VMICPCI-7699 provides all the typical functions of an Intel Pentium II/III-based PC motherboard with the following features:

- Single-slot CompactPCI bus 6U Eurocard form factor
- Includes a high-performance Intel Pentium II/III® Embedded Module processor
- Up to 384 Mbyte of Synchronous DRAM
- AGP video with 2 Mbyte SGRAM
- Real-time clock/calendar
- Front panel reset switch and miniature speaker
- On-board port for keyboard and mouse
- UltraEIDE hard drive and floppy drive (through the CompactPCI backplane connector)
- On-board fast Ethernet controller supporting 10BaseT and 100BaseTx interfaces
- Front panel 'vital sign' indicators (power, UltraEIDE hard drive activity, Ready, and Ethernet status)
- Three general-purpose programmable 16-bit timers
- Software-selectable Watchdog Timer with reset
- Up to 192 Mbyte of bootable flash on secondary IDE (optional)
- 32 Kbyte of non-volatile SRAM
- Two Serial ports (available through the CompactPCI backplane connectors)
- Dual PMC Sites
- One USB Port (available through the CompactPCI backplane connectors)
- CPCI Hot Swap support
- I^2C bus support
- CPCI Geographical Addressing

The VMICPCI-7699 supports standard I/O features such as those listed in Table 1-1. Figure 1-1 on page 23 shows a block diagram of the VMICPCI-7699 emphasizing the I/O features, including the embedded PCI bridge. The serial, USB, IDE, and floppy signals are also routed through the backplane to VMIC's (optional) VMIACC-0577 CompactPCI Rear Transition Utility Board.

Table 1-1 I/O Features

I/O FEATURE	IDENTIFIER	PHYSICAL ACCESS
Two high-performance 16550-compatible serial ports	COM1, COM2	Both through the CompactPCI Backplane Connectors
AT-Style Keyboard/Mouse Controller	M/K	Front Panel PS/2-Style Connector, Mini-DIN Circular (female)
AGP Video Controller with 2 Mbyte SGRAM	SVGA	Front Panel Micro DB9 Connector (Adapter Optional)
Real-Time Clock/Calendar with miniature speaker	Date, Time, and Sound	
On-board Fast Ethernet controller supporting 10BaseT and 100BaseTx interfaces	LAN	Front Panel RJ-45
Floppy Disk Controller	Drive A	Through CompactPCI Backplane (See Note)
Ultra EIDE Hard Drive Controller	Drive C	Through CompactPCI Backplane (See Note)
One USB Port	USB	Through CompactPCI Backplane (See Note)
Hardware Reset	RST	Front Panel Push-Button
Power Status, Hard Drive Activity, Ready, and Ethernet Status for each controller	LED Indicators	Front Panel

NOTE: The VMIACC-0577 CompactPCI Rear Transition Utility Board provides USB, DB25 male serial port connectors, IDE, and Floppy connectors, by way of the CompactPCI backplane connectors.

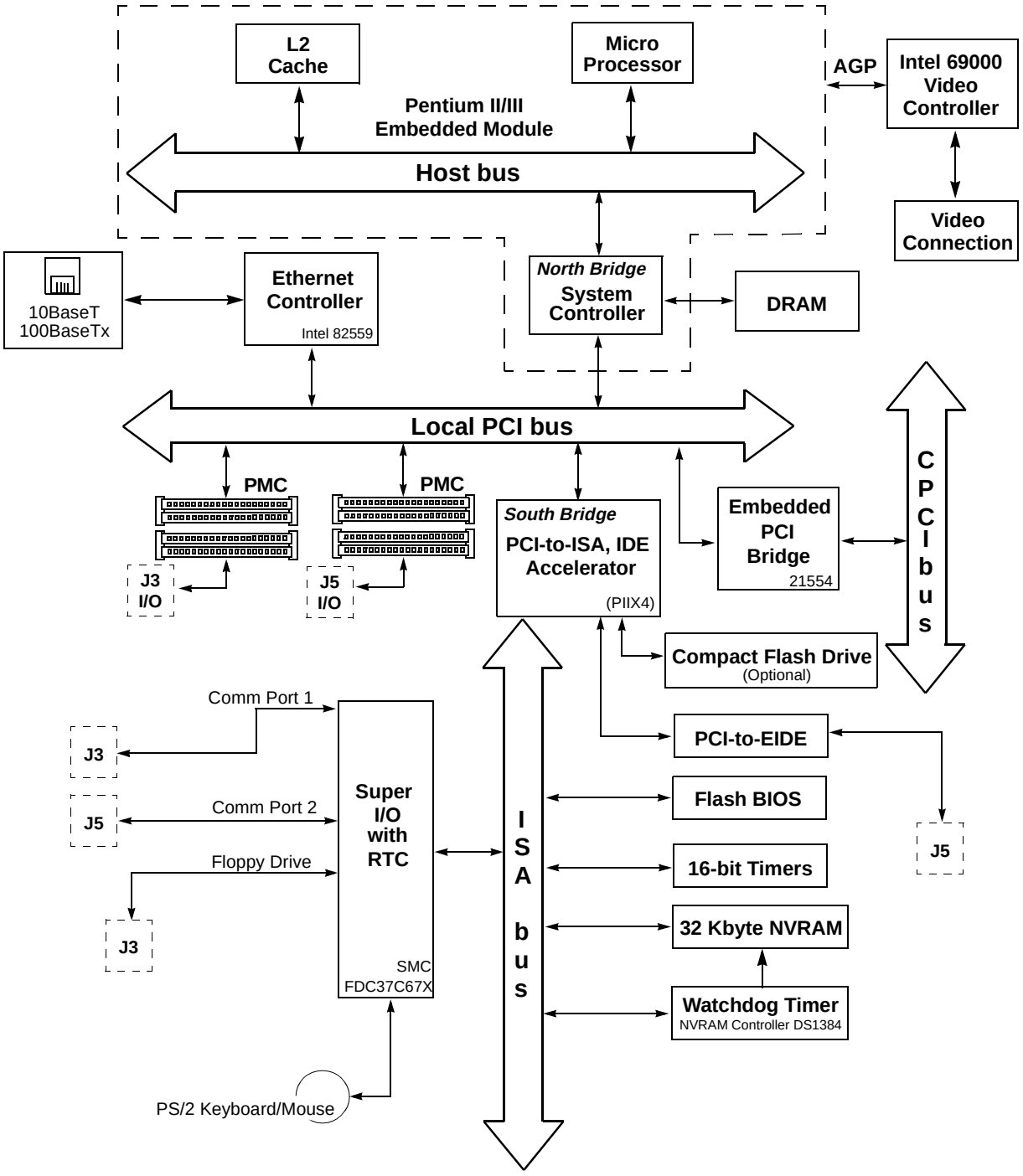


Figure 1-1 VMICPCI-7699 Block Diagram

VMICPCI-7699 Product Options

VMIC's VMICPCI-7699 is built around three fundamental hardware configurations. These configurations involve processor power, RAM memory, and the optional Compact Flash memory size. *These options are subject to change based on emerging technologies and availability of vendor configurations.*

CHAPTER
2

Installation and Setup

Contents

Introduction

This chapter describes the hardware jumper settings, connector descriptions, installation, system setup, and operation of the VMICPCI-7699.

Unpacking Procedures

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

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.

Hardware Setup

The VMICPCI-7699 is factory populated with user-specified options as part of the VMICPCI-7699 ordering information. The CPU speed, RAM size, and flash memory size are not user-upgradable. To change CPU speeds, or RAM / Flash size, contact customer service to receive a Return Material Authorization (RMA).

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

Or E-mail us at customer.service@vmic.com

The VMICPCI-7699 is tested for system operation and shipped with factory-installed header jumpers. The physical location of the jumpers and connectors for the single board CPU are illustrated in Figure 2-1 on page 27. The definitions of the CPU board jumpers and connectors are included in Table 2-1 through Table 2-11. Please note that the VMICPCI-7699 offers two PMC sites designated PMC #1 and PMC #2 as shown in Figure 2-1 on page 27.

CAUTION: All jumpers marked *User Configurable* in the following tables may be changed or modified by the user. All jumpers marked factory configured should not be modified by the user.

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

Modifying any other jumper will void the Warranty and may damage the unit. The default jumper condition of the VMICPCI-7699 is expressed in Table 2-1 through Table 2-11 with **bold text** in the table cells.

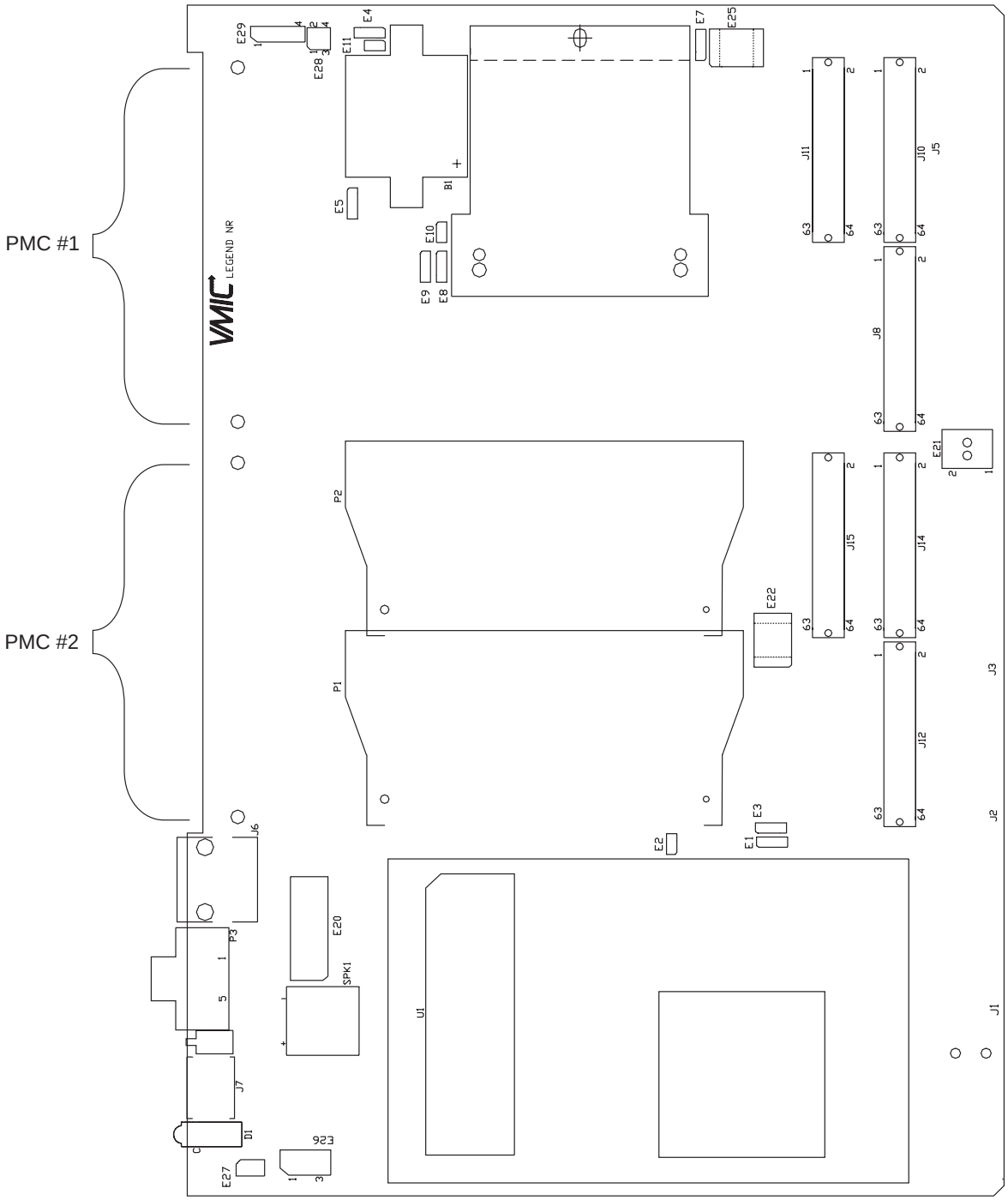


Figure 2-1 VMICPCI-7699 Embedded Module and Jumper Locations

Table 2-1 CPU Board Connectors

Connector	Function
J7	Mouse/Keyboard
J6	Ethernet
P3	Video
E20, E22, E25	Factory Reserved Do Not Use
E26	Fan
J8, J10, J11	PMC Slot 1
J12, J14, J15	PMC Slot 2
J1, J2, J3, J5	CompactPCI
E29	I^2C Header

Note: USB, COM1, COM2, Floppy, and EDIE are available via CPCI J3 and J5 connectors

Table 2-2 Primary Lockout (**User Configurable**) - Jumper (E1)
(Refer to Section 16.2.1 in Volume II)

Select	Jumper Position
Enabled	1 - 2
Disabled	**2 - 3**

Table 2-3 Factory Configured - CPCI Reset Gating - Jumper (E2)

Select	Jumper Position
CPCI Reset Gated By Boot_Done	In
CPCI Reset Not Gated By Boot_Done	**OUT**

Table 2-4 Factory Configured - Hot-Swap Controller Reset Source - Jumper (E3)

Select	Jumper Position
None	Out
Healthy Only	1-2
Healthy# and CPCI Reset (Selected by E2)	**2-3**

Table 2-5 Factory Configured - Boot Block Lock - Jumper (E4)

Select	Jumper Position
Boot Block Programming	1 - 2
No Program	**2 - 3**

Table 2-6 Watchdog Timer (**User Configurable**) - Jumper (E7)

Select	Jumper Position
Watchdog Timer Reset	1-2
No Watchdog Timer Reset or SERR#	**Out**
Watchdog Timer SERR#	2-3

Table 2-7 Programmable Timer Clock Select (**User Configurable**) - Jumper (E8)

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

Table 2-8 PIIX4 Signal Monitor (**User Configurable**) - Jumper (E9)

Select	Jumper Position
None	Out
ENUM#	**1-2**
Hot Swap LED / Switch	2-3

Table 2-9 Password Clear (**User Configurable**) - Jumper (E10)

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

NOTE: The BIOS has the capability (not currently enabled) of password protecting casual access to the unit's CMOS set-up screens. The Password Clear jumper allows the user to clear the password in the case of a forgotten password.

To clear the CMOS password:

1. Turn off power to the unit.
2. Install a jumper at E10.
3. Power up the unit.
4. Turn off the power to the unit and remove the jumper from E10.

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

Table 2-10 WDT Battery Connection (**User Configurable**)- Jumper (E11)

Select	Jumper Position
WDT is battery backed	**In**
WDT is not battery backed	Out

Table 2-11 Factory Configured - PIC ISP - Jumpers (E21)

Select	Jumper Position
Normal	**1-2, 3-4, 5-6**
ISP	Out

Power Requirements

The VMICPCI-7699 requires +5V, +3.3V, +12V and -12V from the CompactPCI backplane. Below are the voltages and current levels.

Supply	Current (Typical)	Current (Maximum)
+5V	2.8A	3.5A
+3.3V	2.2A	2.5A
+12V	105mA	200mA
-12V	50mA	75mA

The VMICPCI-7699 provides power to the two PMC sites in accordance with the PMC specification. The maximum current provided on the +5V supply is 1.5A per PMC site. The maximum current provided on the +3.3V supply is 1.5A per PMC site.

The +12V and -12V supplies are provided to both PMC sites and to the rear-transition board (such as the ACC-0577 board). The total current provided to the VMICPCI-7699 (as indicated above), the two PMC sites and the rear-transition board must not exceed 1A each, in accordance with the CompactPCI HotSwap Specification.

Installation

The VMICPCI-7699 conforms to the CompactPCI physical specification for a 6U board. The VMICPCI-7699 is a peripheral slot only board. It can be plugged directly into any standard chassis accepting this type of board. The following pictures illustrate the symbols used to identify the slots in a standard CompactCPI chassis.

This symbol identifies the System Controller slot

This symbol identifies peripheral slot

The following steps describe the VMIC-recommended method for installation and powerup of the VMICPCI-7699:

1. If a PMC module is to be used, connect it to the VMICPCI-7699 prior to board installation. Refer to the Product Manual for the PMC module for configuration and setup.
2. The VMICPCI-7699 must be installed in a designated peripheral slot of the CompactPCI backplane (See symbols above for selection of the correct slot).

NOTE: The VMICPCI-7699 requires forced air cooling (300 LFM). Please refer to the specification sheet for details. It is advisable to install blank panels over any exposed slots. This will ensure air flow over the VMICPCI-7699 board.

3. Insert the VMICPCI-7699 into a CompactPCI chassis peripheral slot. While ensuring that the board is properly aligned and oriented in the supporting board guides, slide the board smoothly forward against the mating connector. Use the ejector handles to firmly seat the board.
4. All needed peripherals can be accessed from the front panel and the rear I/O VMIACC-0577 Rear Transition utility board. Each connector is clearly labeled on the front panel, and detailed pinouts are in Appendix A.
5. Connect a keyboard and, optionally, a mouse if the system has not been previously configured.
6. The VMICPCI-7699 features an optional Flash Disk resident on the board. Refer to Chapter 4 for set up details.
7. If an external drive module is installed, the BIOS Setup program must be used to configure the drive types. See Appendix C to properly configure the system.
8. If a drive module is present, install the operating system according to the manufacturer's instructions.

Refer to Appendix B for instructions on installing VMICPCI-7699 peripheral driver software during operating system installation.

BIOS Setup

The VMICPCI-7699 has an on-board BIOS Setup program that controls many configuration options. These options are saved in a special non-volatile, 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 VMICPCI-7699 is shipped from the factory with hard drive type configuration set to AUTO in the CMOS.

Details of the VMICPCI-7699 BIOS setup program are included in Appendix C.

Front Panel Connectors

The front panel connections, including connector pinouts and orientation, for the VMICPCI-7699 are defined in detail in Appendix A. Rear panel connections are defined in detail in the Installation Guide for the VMIACC-0577.

PMC Site Connector

The VMICPCI-7699 supports IEEE-P1386 Common Mezzanine Card Specification with two 5 V PMC sites. Figure 2-2 shows the installation of a PMC card on the VMICPCI-7699.

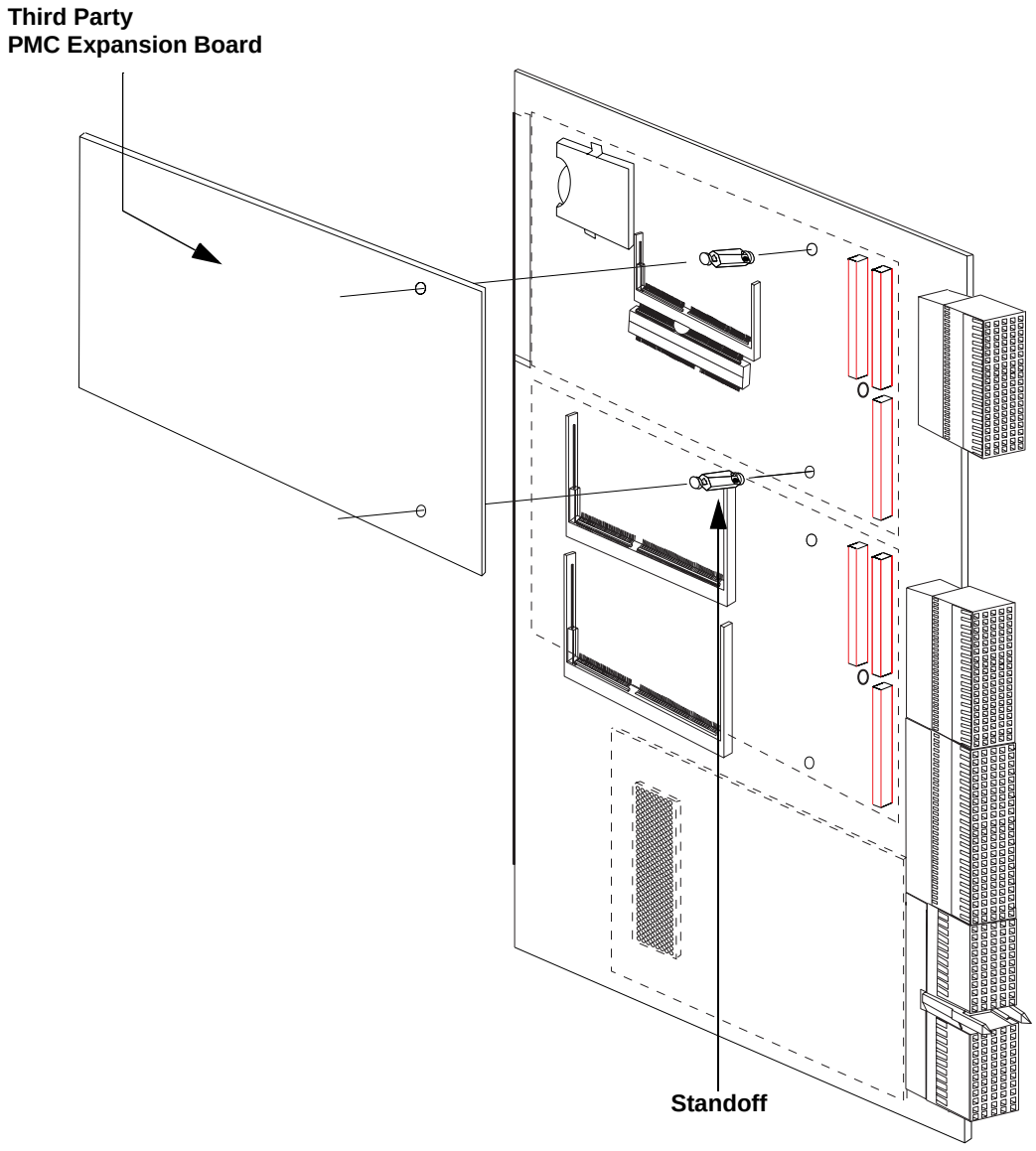
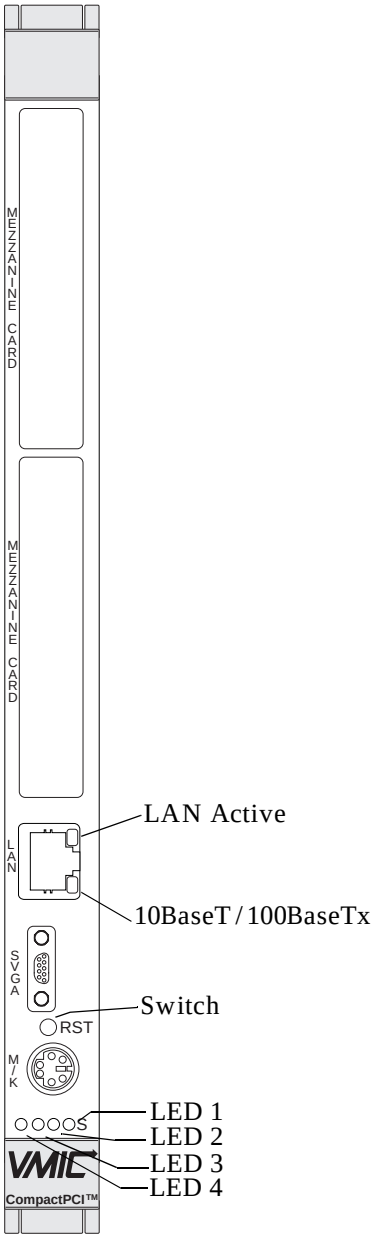


Figure 2-2 Installing a PMC Card on the VMICPCI-7699

LED Definition



Switch	*Reset* - Allows the system to be reset from the front panel.
LED 1	*System* - Indicates BIOS Boot is in progress. When LED is off, module is ready.
LED 2	*Hard Drive Indicator* - Indicates when hard drive activity is occurring.
LED 3	*Power* - Indicates when power is applied to the board.
LED 4	*Hot Swap* - Lights during Hot Swap insertion and extraction.
LAN Active	Indicates the Ethernet is active, (yellow LED).
10 / 100BaseTx	Indicates whether 10BaseT or 100BaseTx mode. Yellow LED indicates 10BaseT, and Green LED indicates 100BaseTx.

Figure 2-3 Front Panel LED Positions

CHAPTER
3

Standard CPU Functions

Contents

Introduction

The VMICPCI-7699 is an Intel Pentium II/III Embedded Module-based computer. This design includes a high-speed microprocessor with memory utilizing current SDRAM technology, and optional M-Systems DiskOnChip flash. Reference the VMIC product specifications for available component options.

Because the design is Intel Pentium II/III compatible, it retains standard memory and I/O maps along with a standard interrupt architecture. Furthermore, the VMICPCI-7699 includes an Intel 82559 Ethernet controller.

The following sections describe in detail the standard functions of the VMICPCI-7699.

CPU Socket

The VMICPCI-7699 CPU socket is factory populated with a high-speed Pentium II or III processor. The CPU speed and RAM / flash size are user specified as part of the VMICPCI-7699 ordering information.

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

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

Physical Memory

The VMICPCI-7699 provides Synchronous DRAM (SODIMM) as on-board system memory. Memory can be accessed as bytes, words, or longwords.

The VMICPCI-7699 accepts up to two 144-pin SODIMM SDRAM modules. The on-board DRAM is dual-ported to the CPCI bus through the PCI-to-PCI bridge and is addressable by the local processor.

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

The VMICPCI-7699 provides Compact Flash memory (See Note) accessible through the secondary IDE port. The VMICPCI-7699 BIOS includes an option to allow the board to boot from the flash memory or from the Ethernet port.

The VMICPCI-7699 memory includes 32 Kbyte of non-volatile SRAM addressed at $D8000 to $DFFFF. All but the first 20 bytes are accessible as SRAM. Bytes $D8000 through $D800D are reserved for the Watchdog Timer registers, and bytes $D800E through $D8013 are reserved for other on-board functions. The SRAM can be accessed by the CPU at any time, and is used to store system data that must not be lost during power-off conditions.

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

I/O Port Map

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

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

The assigned and user-available I/O addresses are summarized in the I/O Address Map, Table 3-1.

Table 3-1 VMICPCI-7699 I/O Address Map

I/O ADDRESS RANGE	SIZE IN BYTES	HW DEVICE	PC/AT FUNCTION
$000 - $00F	16		DMA Controller 1 (Intel 8237A Compatible)
$010 - $01F	16		Reserved
$020 - $021	2		Master Interrupt Controller (Intel 8259A Compatible)
$022 - $03F	30		Reserved
$040 - $043	4		Programmable Timer (Intel 8254 Compatible)
$044 - $05F	30		Reserved
$060 - $064	5		Keyboard, Speaker, System Configuration (Intel 8042 Compatible)
$065 - $06F	11		Reserved
$070 - $071	2		Real-Time Clock,
$072 - $07F	14		Reserved
$080 - $08F	16		DMA Page Registers
$090 - $091	2		Reserved
$092	1		Alt. Gate A20/Fast Reset Register
$093 - $09F	11		Reserved

Table 3-1 VMICPCI-7699 I/O Address Map (Continued)

I/O ADDRESS RANGE	SIZE IN BYTES	HW DEVICE	PC/AT FUNCTION
$0A0 - $0A1	2		Slave Interrupt Controller (Intel 8259A Compatible)
$0A2 - $0BF	30		Reserved
$0C0 - $0DF	32		DMA Controller 2 (Intel 8237A Compatible)
$0E0 - $16F	142		Reserved
$170 - $177	8	PIIX4E	Secondary Hard Disk Controller
$178 - $1EF	120		User I/O
$1F0 - $1F7	8	PIIX4E	Primary Hard Disk Controller
$1F8 - $277	128		User I/O
$278 - $27F	8	I/O Chip*	LPT2 Parallel I/O*
$280 - $2E7	104		Reserved
$2E8 - $2EE	7	UART*	COM4 Serial I/O*
$2EF - $2F7	9		User I/O
$2F8 - $2FE	7	Super-I/O Chip	COM2 Serial I/O (16550 Compatible)
$2FF - $36F	113		Reserved
$370 - $377	8	Super-I/O Chip	Secondary Floppy Disk Controller
$378 - $37F	8	Super-I/O Chip*	LPT1 Parallel I/O*
$380 - $3E7	108		Reserved
$3E8 - $3EE	7	UART*	COM3 Serial I/O*
$3F0 - $3F7	8	Super-I/O Chip	Primary Floppy Disk Controller
$3F8 - $3FE	7	Super-I/O Chip	COM1 Serial I/O (16550 Compatible)
$3FF - $4FF	256		Reserved
$500 - $503	4	82C54 Timer	Programmable Internal Timer
$504 - CFF	2043		Reserved

* While these I/O ports are reserved for the listed functions, they are not implemented on the VMICPCI-7699. They are listed here to make the user aware of the standard PC usage of these ports.

Embedded PCI Bridge

The VMICPCI-7699 uses the Intel Embedded PCI bridge to interface between the primary PCI bus and the CompactPCI (CPCI) bus. The CompactPCI bus appears as a secondary PCI bus, and all devices in the seven peripheral slots of the CPCI chassis are accessed with normal PCI accesses. Complete details of the embedded PCI Bridge are included in Volume II of the VMICPCI-7699 manual.

ISA Interrupts

In addition to an I/O port address, an I/O device has a separate hardware interrupt line assignment. Assigned to each interrupt line is a corresponding interrupt vector in the 256-vector interrupt table at $00000 to $003FF in memory. The 16 maskable interrupts and the single Non-Maskable Interrupt (NMI) are listed in Table 3-2 along with their functions. Table 3-3 on page 40 details the vectors in the interrupt vector table. The interrupt number in HEX and decimal are also defined for real and protected mode in Table 3-3 on page 40.

The interrupt hardware implementation on the VMICPCI-7699 is standard for computers built around the ISA architecture, which evolved from the IBM PC/XT. In the IBM PC/XT computers, only eight interrupt request lines exist, numbered from IRQ0 to IRQ7 at the PIC. The IBM PC/AT computer added eight more IRQx lines, numbered IRQ8 to IRQ15, by cascading a second slave PIC into the original master PIC. IRQ2 at the master PIC was committed as the cascade input from the slave PIC. This architecture is represented in Figure 3-1 on page 44.

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

Table 3-2 ISA Hardware Interrupt Line Assignments

IRQ	AT FUNCTION	COMMENTS
NMI	Parity Errors (Must be enabled in BIOS Setup)	Used by VMICPCI-7699 PCIbus Interface
0	System Timer	Set by BIOS Setup
1	Keyboard	Set by BIOS Setup
2	Duplexed to IRQ9	
3	COM2	
4	COM1	
5	Timer/I^2C	Assigned to On-Board Timer and I^2C Controller
6	Floppy Controller	
7	Unused	

Table 3-2 ISA Hardware Interrupt Line Assignments (Continued)

IRQ	AT FUNCTION	COMMENTS
8	Real-Time Clock	
9	Old IRQ2	SVGA or Network I/O
10	Not Assigned	Determined by BIOS
11	Not Assigned	Determined by BIOS
12	Mouse	
13	Math Coprocessor	
14	AT Hard Drive	
15	Flash Drive	

Table 3-3 ISA Interrupt Vector Table

INTERRUPT NO.		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
00	0		Divide Error	Same as Real Mode
01	1		Debug Single Step	Same as Real Mode
02	2	NMI	Memory Parity Error, CompactPCI 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	Timer/I^2C	Same as Real Mode
0E	14	IRQ6	Floppy Disk Controller	Page Fault
0F	15	IRQ7	Unassigned	Unassigned

Table 3-3 ISA Interrupt Vector Table (Continued)

INTERRUPT NO.		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
10	16		BIOS Video I/O	Coprocessor Error
11	17		System Configuration Check	Same as Real Mode
12	18		Memory Size Check	Same as Real Mode
13	19		XT Floppy/Hard Drive	Same as Real Mode
14	20		BIOS Comm I/O	Same as Real Mode
15	21		BIOS Cassette Tape I/O	Same as Real Mode
16	22		BIOS Keyboard I/O	Same as Real Mode
17	23		BIOS Printer I/O	Same as Real Mode
18	24		ROM BASIC Entry Point	Same as Real Mode
19	25		Bootstrap Loader	Same as Real Mode
1A	26	IRQ8	Real-Time Clock	Same as Real Mode
1B	27		Control/Break Handler	Same as Real Mode
1C	28		Timer Control	Same as Real Mode
1D	29		Video Parameter Table Pntr	Same as Real Mode
1E	30		Floppy Parm Table Pntr	Same as Real Mode
1F	31		Video Graphics Table Pntr	Same as Real Mode
20	32		DOS Terminate Program	Same as Real Mode
21	33		DOS Function Entry Point	Same as Real Mode
22	34		DOS Terminate Handler	Same as Real Mode
23	35		DOS Control/Break Handler	Same as Real Mode
24	36		DOS Critical Error Handler	Same as Real Mode
25	37		DOS Absolute Disk Read	Same as Real Mode
26	38		DOS Absolute Disk Write	Same as Real Mode
27	39		DOS Program Terminate, Stay Resident	Same as Real Mode
28	40		DOS Keyboard Idle Loop	Same as Real Mode
29	41		DOS CON Dev. Raw Output	Same as Real Mode

Table 3-3 ISA Interrupt Vector Table (Continued)

INTERRUPT NO.		IRQ LINE	REAL MODE	PROTECTED MODE
HEX	DEC			
2A	42		DOS 3.x+ Network Comm	Same as Real Mode
2B	43		DOS Internal Use	Same as Real Mode
2C	44		DOS Internal Use	Same as Real Mode
2D	45		DOS Internal Use	Same as Real Mode
2E	46		DOS Internal Use	Same as Real Mode
2F	47		DOS Print Spooler Driver	Same as Real Mode
30-60	48-96		Reserved by DOS	Same as Real Mode
61-66	97-102		User Available	Same as Real Mode
67-71	103-113		Reserved by DOS	Same as Real Mode
72	114	IRQ10	Not Assigned	
73	115	IRQ11	Not Assigned	
74	116	IRQ12	Mouse	
75	117	IRQ13	Math Coprocessor	
76	118	IRQ14	AT Hard Drive	
77	119	IRQ15	Flash Drive	
78-7F	120-127		Reserved by DOS	Same as Real Mode
80-F0	128-240		Reserved for BASIC	Same as Real Mode
F1-FF	241-255		Reserved by DOS	Same as Real Mode

PCI Interrupts

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

PCI defines one interrupt line for a single function device and up to four interrupt lines for a multifunction device or connector. For a single function device, only INTA# may be used while the other three interrupt lines have no meaning. Figure 3-1 on page 44 depicts the VMICPCI-7699 interrupt logic pertaining to CompactPCI operations and the PMC site.

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

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

The PCI-to-PCI Bridge has the capability of generating a Non-Maskable Interrupt (NMI) via the PCI SERR# line. Table 3-4 describes the register bits that are used by the NMI. The SERR interrupt is routed through logic back to the NMI input line on the CPU. The CPU reads the NMI Status Control register to determine the NMI source (bits set to 1). After the NMI interrupt routine processes the interrupt, software clears the NMI status bits by setting the corresponding enable / disable bit to 1. The NMI Enable and Real-Time Clock register can mask the NMI signal and disable / enable all NMI sources.

Table 3-4 NMI Register Bit Descriptions

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

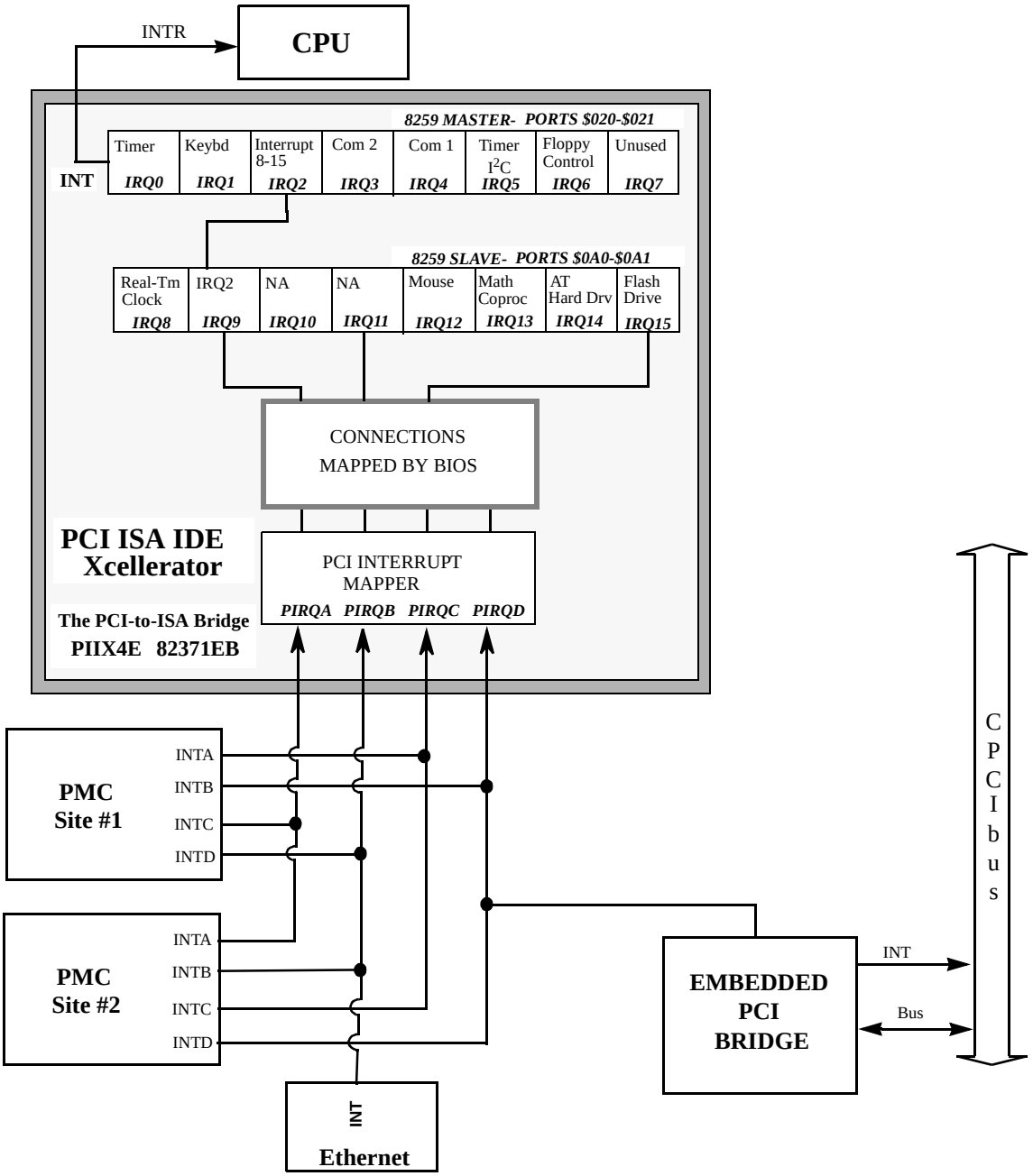


Figure 3-1 Connections for the PC Interrupt Logic Controller

Integrated Peripherals

The VMICPCI-7699 incorporates the SMC Super-I/O chip. The SMC chip provides the VMICPCI-7699 with a standard floppy drive controller and two 16550 UART-compatible serial ports. The Ultra-EIDE hard drive interface is provided by the Intel 82371EB (PIIX4E) PCI ISA IDE Xcelerator chip. All ports are located on the J3 and J5 connectors, for use with VMIC's VMIACC-0577 CompactPCI Rear Transition Utility Board.

Ethernet Controller

The network capability is provided by the Intel 82559 Ethernet Controller. This Ethernet controller is PCI bus based and is software configurable. The VMICPCI-7699 supports 10BaseT and 100BaseTx Ethernet.

10BaseT

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

100BaseTx

The VMICPCI-7699 also supports the 100BaseTx Ethernet. A network based on a 100BaseTx standard uses unshielded twisted-pair cables and a RJ-45 connector. The 100BaseTx has a maximum deployment length of 100 meters.

Remote Ethernet Booting

The VMICPCI-7699 is capable of booting from a server over the Ethernet utilizing Lanworks BootWare. BootWare gives you the ability to remotely boot the VMICPCI-7699 using Netware, TCP/IP, or RPL network protocols. The Ethernet must be connected through the LAN front panel (RJ-45) connector to boot remotely. This feature allows users to create systems without the worry of disk drive reliability, or the extra cost of adding Flash drives.

BootWare Features:

- Netware, TCP/IP, RPL compatible
- Unparalleled boot sector virus protection
- Detailed boot configuration screens
- Comprehensive diagnostics
- Optional disabling of local boots
- Dual-boot option lets users select network or local booting

Video Graphics Adapter

The SVGA port on the VMICPCI-7699 is controlled by an Intel 69000 chip with 2 Mbyte video DRAM. The video controller chip is hardware and BIOS compatible with the IBM EGA and SVGA standards supporting both VESA high-resolution and extended video modes. Table 3-5 shows the graphics video modes supported by the Trio 3D video chip.

Table 3-5 Supported Graphics Video Resolutions

Screen Resolution	Maximum Colors	Maximum Refresh Rates (Hz)
640 x 480	16 M	85
800 x 600	16 M	85
1,024 x 768	16 M	85
1,280 x 1,024	64 K	60
1600 x 1200	64 K	60

Not all SVGA monitors support resolutions and refresh rates beyond 640 x 480 at 85 Hz. Do not attempt to drive a monitor to a resolution or refresh rate beyond its capability.

CHAPTER

4

Embedded PC/RTOS Features

Contents

Introduction

VMIC's VMICPCI-7699 features additional capabilities beyond those of a typical desktop computer system. The unit provides three software-controlled, general-purpose timers along with a programmable Watchdog Timer for synchronizing and controlling multiple events in embedded applications. The VMICPCI-7699 also provides a bootable Flash Disk system and 32 Kbyte of non-volatile SRAM. These features make the unit ideal for embedded applications, particularly applications where standard hard drives and floppy disk drives cannot be used.

Access to the CompactPCI bus is provided by an Intel non-transparent bridge. The VMICPCI-7699 also supports CompactPCI Hot Swapping, geographical addressing, and I^2C by integrating specialized circuitry for these functions.

Timers

General

The VMICPCI-7699 provides a user-programmable 82C54 internal timer/counter. The 82C54 provides three independent, 16-bit timers each operating at a 1 or 2 MHz clock speed. This is determined by the configuration of jumper E8; reference Chapter 2. These timers are completely available to the user, and are not dedicated to any standard PC function. These timers may be used to generate system interrupts.

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

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

Timer Interrupt Status

A single interrupt, IRQ5, is used by all three Timers. A Timer Interrupt Status register is provided in order to determine which Timer(s) initiated an interrupt. The Interrupt Status Register is a general-purpose input register located (refer to Figure 4-1), external to the 82C54, at offset $31 from the Power Management Base I/O address. The interrupt status register address can be found by first determining the PCI Configuration Base address for Device ID $7113 and Vendor ID $8086. The Power Management Base I/O address can be found by reading offset $40 from this PCI Configuration address. The Timer Interrupt Status register bits are located at offset $31 from the Power Management Base I/O address, bits 5, 6, and 7.

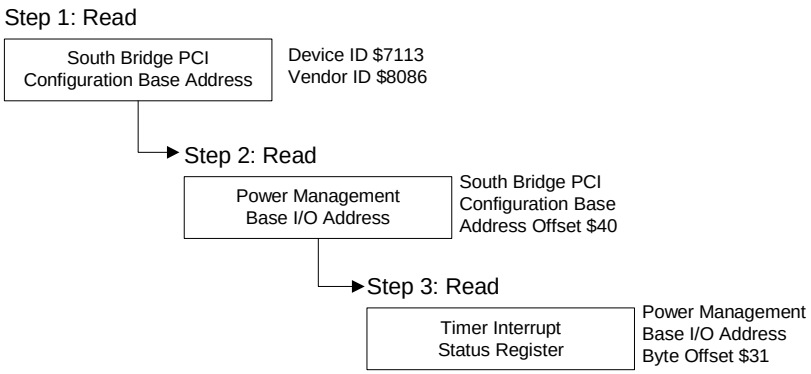


Figure 4-1 Timer Interrupt Status Register Read/Steps

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

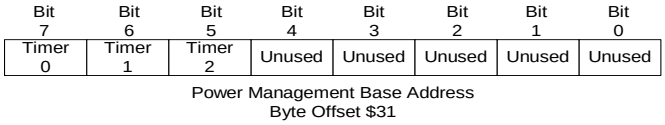


Figure 4-2 Timer Interrupt Status Register

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

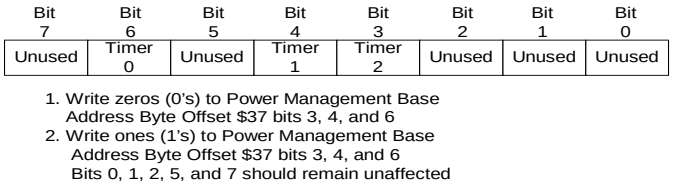


Figure 4-3 Clearing the Timer Interrupt Status Register

Clearing the Interrupt

The Timer Interrupts are cleared using the standard procedure for clearing ISA IRQ5. Refer to Appendix F for example code using the 82C54 timers.

Timer Programming

Architecture

The VMICPCI-7699 Timers are mapped in I/O address space starting at $500. See Table 4-1 below. The Timers, consisting of three 16-bit timers and a Control Word register (see Figure 4-4 on page 52), are read from/written to by way of a 8-bit data bus.

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

I/O Address	Select
$500	Timer 0
$501	Timer 1
$502	Timer 2
$503	control word Register

Table 4-1 on page 51 shows the I/O addresses of the control word Register and Timers.

The control word Register is write only. The Timer status information can be obtained from the Read-Back command (see the Section titled Reading on page 55).

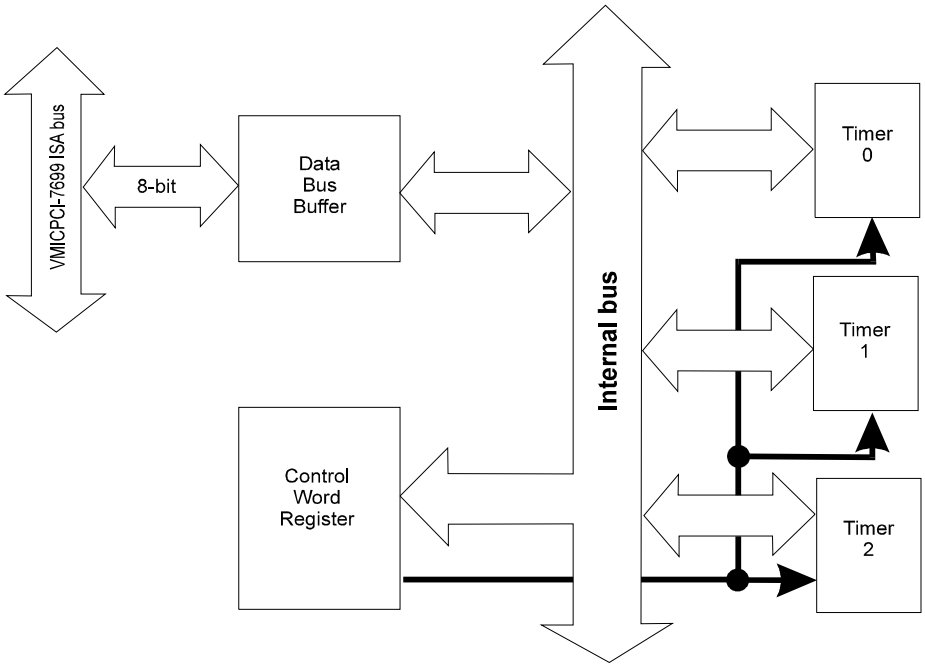


Figure 4-4 82C54 Diagram

The three timers, Timer 0, 1, and 2, are functionally equivalent, therefore only a single Timer will be described. Figure 4-5 on page 53 is a block diagram of a Timer. Each Timer is functionally independent. Although the control word is shown in the Timer block diagram, it is not a part of the Timer, but its contents directly affect how the Timer functions.

The status register, when latched (as shown in Figure 4-5 on page 53), contains the present contents of the Control Word Register and the present state of the output and load count flag. The Status Word is available via the Read-Back command (see the Section titled Reading on page 55). The Timer is labelled TE (Timer Element). It is a 16-bit synchronous presettable down counter.

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

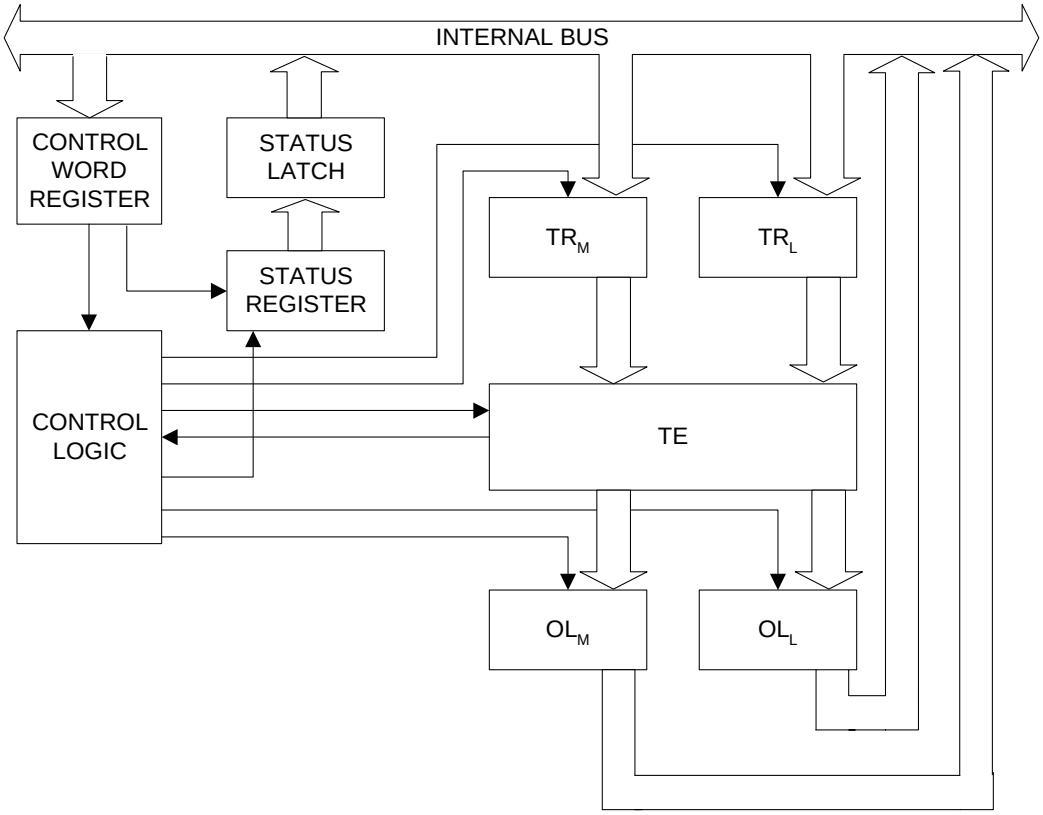


Figure 4-5 Internal Timer Diagram

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

Writing

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

Table 4-2 Control Word Format

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

Table 4-3 ST - Select Timer

ST1*	ST0*	Description
0	0	Select Timer 0
0	1	Select Timer 1
1	0	Select Timer 2
1	1	Read-Back Command (See Reading section on page 55)
* The ST bits specify which Timer (0, 1, or 2) the control word refers to and whether this is a Read-Back command		

Table 4-4 RW - Read/Write

RW1*	RW0*	Description
0	0	Timer Latch Command (see Reading section)
0	1	Read / Write least significant byte only
1	0	Read / Write most significant byte only
1	1	Read / Write least significant byte first, then most significant
* The RW bits specify whether this is a Timer Latch command or the byte ordering of the Read / Write transaction.		

Table 4-5 M - Mode

M2*	M1*	M0*	Description
0	0	0	Mode 0
0	0	1	Mode 1
X	***1***	***0***	***Mode 2***
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5
* Only Mode 2 is described in this manual.			

Table 4-6 BCD

BCD*	Description
0	Binary Timer 16-bits
1	Binary Coded Decimal (BCD) Timer (4 Decades)
* The BCD bit specifies whether the Timer count value is in Binary or BCD.	

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

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

Reading

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

Timer Latch Command

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

Like a control word, the Timer Latch Command is written to the Control Word Register (I/O Address $503, see Table 4-1 on page 51). The Select Timer bits (ST1, ST0, see Table 4-3 on page 54) select one of the three timers while the Read/Write bits (RW1, RW0, see Table 4-4 on page 54) select the Timer Latch Command, RW1 = 0 and RW0 = 0. The selected Timer's count is latched into the 0L registers at the time of the Timer Latch Command. The count is held in the 0L latches until it is read. Multiple Timer Latch Commands can be used to latch more than one Timer. Again, each Timer's count is held latched until it is read.

Read-Back Command

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

Table 4-7 Read-Back Command Format

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

Table 4-8 Read-Back Command Description

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

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

The Read-Back command can also be used to latch the timer status by setting the $\overline{\text{Status}}$ bit = 0 and selecting the Timers. Status of a Timer is accessed by a read from that Timer (see Table 4-1 on page 51). If more than one Timer Status Read-Back command is issued without reading the status, all but the first is ignored.

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

Table 4-9 Status Byte

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

Table 4-10 Status Byte Description

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

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

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

Bits D5 through D0 contain the Timer's programmed mode exactly, bit for bit, like the Timer control words bits D5 through D0. See Table 4-2 on page 54.

Table 4-11 LOAD Bit Operation

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

[1]Only the Timer specified in the control word will have its LOAD bit set to 1. LOAD bits of other Timers are not affected.
[2]If the Timer is programmed for two byte counts (least significant, then most significant), the LOAD bit will go to 1 when the second byte is written.

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

Mode Definitions

The VMICPCI-7699 utilizes an 82C54 Timer / Counter for its Timers. The 82C54 Timer / Counter can be programmed to function in six different modes (numbered Mode 0 through Mode 5). The VMICPCI-7699 Timers are hardware configured to operate using Mode 2. Only Mode 2 is supported by VMIC.

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

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

Flash Disk

The VMICPCI-7699 features an optional on-board Compact Flash mass storage system which appears to the user as an intelligent ATA (IDE) disk drive with the same functionality and capabilities as a “rotating media” IDE hard drive. The VMICPCI-7699 BIOS includes an option to allow the board to boot from the Flash Disk.

Configuration

The Flash Disk resides on the VMICPCI-7699 as the secondary IDE bus master device (the secondary IDE bus slave device is not assignable). The default setting in the Phoenix BIOS ‘STANDARD CMOS SETUP’ screen is the ‘AUTO’ setting. In the Phoenix BIOS ‘PERIPHERAL SETUP’ screen, the secondary PCI IDE interface must be enabled for the Flash Disk to be functional. Refer to Appendix C, Phoenix-Basic Input/Output System (BIOS) for additional details.

Figure 4-6 maps the configuration possibilities for a typical system consisting of the VMICPCI-7699 with a resident Flash Disk, a hard drive attached to the Primary IDE interface, and a floppy drive attached to the floppy interface.

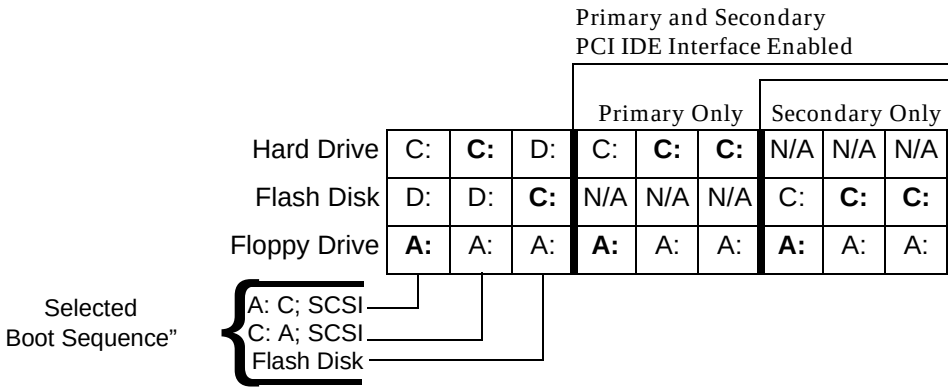


Figure 4-6 Typical System Configuration

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

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

Functionality

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

Advanced Configuration

The previous discussion is based on using the IDE disk devices formatted as one large partition per device. Some applications may require the use of multiple partitions. The following discussion of these partitions includes special procedures that must be followed to create multiple partitions on the VMICPCI-7699 IDE disk devices (including the resident Flash Disk).

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

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

As discussed earlier, a typical system consists of the VMICPCI-7699 with its resident Flash Disk configured as the Secondary IDE device, a hard drive attached to the Primary IDE interface, and a floppy drive attached to the floppy interface.

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

The DOS utility FDISK is commonly used to configure the partition structure on a hard drive. Comments on the following page pertain to partitioning efforts using FDISK.

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

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

1. Power up the VMICPCI-7699 and enter the CMOS set-up.
2. Set IDE HDD Master to “Not Installed”.

3. Set Flash Disk Master to “AUTO”.
4. Set boot device to floppy.
5. Boot DOS from the floppy, and verify that the System Configuration Screen shows only the Flash Disk.
6. Run FDISK.
7. Delete all current partitions (any data currently stored in the partitions will be lost).
8. Exit FDISK (this will cause a reboot), then run FDISK again.
9. Create an 8 Mbyte primary partition.
10. Create an 8 Mbyte extended partition.
11. Set-up a logical device for the 8 Mbyte extended partition.
12. Set the Primary partition as an active partition.
13. Exit FDISK.

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

14. Run FDISK/MBR.
15. Run FORMAT C: (use the extension /s option if you want the Flash Disk as a bootable DOS device).
16. Format D: (this is only required if two partitions were created).
17. Reset the CPU and enter the CMOS set-up.
18. Set Primary Master to “AUTO”.
19. Set boot device to desired boot source.

Drive letter assignments for a simple system were illustrated in Figure 4-6 on page 59. Understanding the order the operating system assigns drive letters is necessary for these multiple partition configurations. The operating system assigns drive letter C to the active primary partition on the first hard disk (the boot device). Drive D is assigned to the first recognized primary partition on the next hard disk. The operating system will continue to assign drive letters to the primary partitions in an alternating fashion between the two drives. The next logical partitions will be assigned drive letters starting on the first hard drive lettering each logical device sequentially until all are assigned a drive letter, then doing the same sequential lettering of each logical partition on the second hard disk.

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

Watchdog Timer

The VMICPCI-7699 utilizes a Dallas DS1384 Watchdog Timekeeping Controller as its Watchdog Timer. The device provides a Time of Day feature, a Watchdog Alarm and an Non-Volatile SRAM controller. The Time of Day feature found within the DS1384 device is explained in this section. The actual Time of Day registers used by the VMICPCI-7699 are located at the standard ISA I/O address. The Time of Day feature in the DS1384 Watchdog Timer is available for use at the user's discretion.

The Non-Volatile SRAM is explained in the battery-backed SRAM section of this manual.

The Watchdog Timer provides a Watchdog Alarm window and interval timing between 0.01 and 99.99 seconds. If enabled, and if jumper E7 pins 1 and 2 are loaded, the Watchdog Alarm will reset the CPU to a known state if not accessed during the alarm window. If pins 2 and 3 are jumpered, the watchdog alarm can be programmed to generate a Non-Maskable Interrupt (NMI) to the CPU.

Figure 4-7 shows a generalized block diagram of how the Watchdog Timer is used in the VMICPCI-7699. The Watchdog Timer registers are memory-mapped in the bottom fourteen locations of battery-backed SRAM addresses $D8000 through $D800D. Table 4-12 on page 63 shows the address, content and the range of each Watchdog Register.

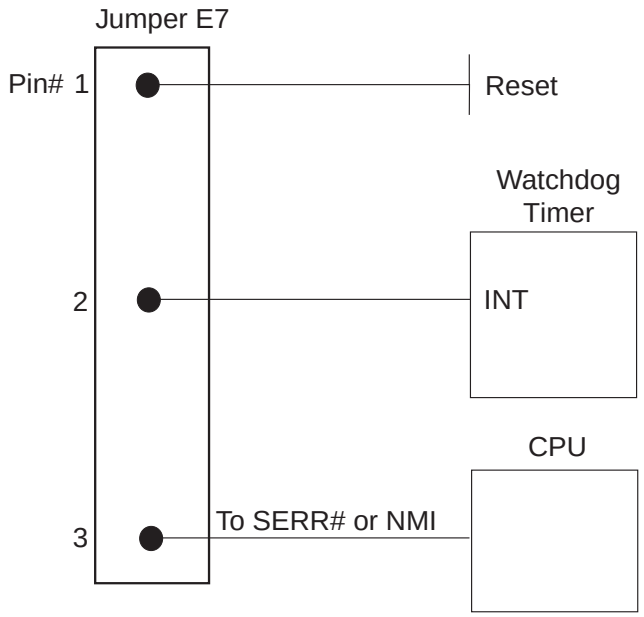


Figure 4-7 Watchdog Alarm Block

NOTE: If the Watchdog Timer is being used to reset the VMICPCI-7699, and the user only wants one reset; when the Watchdog Timer times out the user should "enable" the clearing of the Watchdog Timer in the BIOS CMOS Setup (see Appendix C, Advanced Menu). Also note that if the Watchdog Timer is being used to reset the CPU, then the minimum time that can be programmed into the Watchdog Timer, and still have the BIOS clear the timer, will be 300 ms.

Table 4-12 Watchdog Registers

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

* In the 12 hour mode Bit 5 determines AM (0) or PM (1). In the 24 hour mode Bit 5 combines with Bit 4 to represent the 10 hour value.
** Bit 6 of Register 9 must be set to a 1. If set to a 0, an unused square wave will be generated in the circuit.

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

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

Time of Day Registers

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

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

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

Register 2 contains two Time of Day values. Bits 3 - 0 contain the 1 Minute value with a range of 0 to 9 in BCD, Bits 7 - 4 contain the 10 Minutes value with a range of 0 to 9 in BCD. This register has a total range of 0 to 59 Minutes. Bit 7 of this register will always be zero regardless of what value is written to it.

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

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

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

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

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

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

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

Time of Day Alarm Registers

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

Table 4-13 Time of Day Alarm Registers

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

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

Watchdog Alarm Registers

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

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

The Watchdog Alarm Registers can be read or written in any order. When a new value is entered or the Watchdog registers are read, the Watchdog Timer will start counting down from the entered value. When zero is reached, the Watchdog Interrupt Output will go active. If jumper E7 is loaded, the CPU will reset to a known state (Refer to Figure 4-7 on page 62). The Watchdog Timer count is reinitialized back to the entered value, the Watchdog flag bit is cleared, and the Watchdog interrupt output is cleared every time either of the registers are accessed. Periodic accesses to the Watchdog Timer will prevent the Watchdog Alarm from occurring. If access does not occur, the alarm will be repetitive. The Watchdog Alarm Register always reads the entered value. The actual countdown value is internal and not accessible to the user. Writing zero's to Registers C and D will disable the Watchdog Alarm feature.

Command Register

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

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

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

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

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

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

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

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

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

Non-Volatile SRAM

The VMICPCI-7699 includes 32 Kbyte of non-volatile SRAM located at address 0xD8000 to 0xDFFFF. The lower 14 bytes (0xD8000 - 0xD800D) are used by the Watchdog Timer registers and are unavailable for general use. See the Watchdog Timer section of this chapter for more information. The next 6 bytes (0xD010 - 0xD8013) are used by the VMICPCI-7699 to support the I^2C-bus controller, geographical addressing, and a board ID register. A complete memory map of the non-volatile SRAM space is shown in Table 4-14.

Table 4-14 SRAM Memory Map

Address	Function
0xD8000 - 0xD80D	WDT Register Space
0xD800E - 0xD800F	Board ID (should read as 0x7699)
0xD8010 - 0xD8011	Geographical Address
0xD8012 - 0xD8013	I^2C Register I/O
0xD8014 - 0xDFFFF	Non-volatile SRAM

CompactPCI Bus Bridge

The VMICPCI-7699 incorporates a PCI peripheral device that performs PCI bridging functions for embedded and intelligent I/O applications. The PCI device acts as a gateway to an intelligent subsystem. It allows the local VMICPCI-7699 Pentium II/III processor to configure and control the on-board local subsystem independent from the CompactPCI bus host processor. The VMICPCI-7699 local PCI subsystem is presented to the CompactPCI bus host as a single CompactPCI device. For detailed information concerning the embedded PCI bus bridge consult the manual titled *Intel 21554 Embedded PCI-to-PCI Bridge for the VMICPCI-7699 CompactPCI CPU*.

The VMICPCI-7699 PCI bridge device provides the following features:

PCI Interface

- Full compliance with the PCI Local Bus Specification, Revision 2.1
- 3.3 V operation with 5.0 V tolerant I/O
- Concurrent local (secondary) and CompactPCI (primary) bus operation
- Compliant with CompactPCI Hot Swap specification PICMG 2.1 R1.0 for Full and High Availability Hot Swap

Buffer Architecture

- Queuing of multiple transactions in either direction
- 256 byte of posted write (data and address) buffering in each direction
- 256 byte of read data buffering in each direction
- Four delayed transaction entries in each direction

Configuration Register and Control/Status Registers (CSRs)

- Two sets of standard PCI configuration registers corresponding to the local and CompactPCI interfaces. Each set is accessible from either the local or CompactPCI interface.
- Four 32-bit base address configuration registers mapping the CSRs.

Transaction Forwarding

- Four primary interface base address configuration registers for downstream forwarding with size and prefetchable programming for all four address ranges
- Direct offset address translation for downstream memory and I/O transactions
- Three secondary interface address ranges for upstream forwarding with size and prefetchable programming for all three address ranges
- Inverse decoding above 4 Gbyte address boundary for upstream Digital-to-Analog Controllers (DACs)
- Ability to generate type 0 and type 1 configuration commands on the primary or secondary interface by way of configuration cycles or I/O cycles accessing the CSR

Hot Swap Support

The VMICPCI-7699 implements hardware to support both Full Hot Swap and High Availability Hot Swap per the CompactPCI Hot Swap Specification Rev 1.0. The hot swap circuits contain an integrated power controller, an I/O signal pre-charge circuit, and a CompactPCI hot swap controller. These hot swap functions allow the VMICPCI-7699 to be inserted into, or extracted from, a CompactPCI system while the system is still powered.

When the VMICPCI-7699 is inserted into a powered system, a physical connection will be made with the chassis in order to properly discharge the voltage potentials to ground. As the VMICPCI-7699 connector is mated with the backplane connector, the long power pins of the backplane will provide early power to the unit. As soon as power is applied; the on-board VMICPCI-7699 power controller will assert the reset outputs to the backend logic, turn off gate voltages to external power FETs, and begin outputting a reference voltage that is used to pre-charge the CompactPCI interface logic I/O pins before they mate with the bus signals. The next pins to mate are the I/O pins and the balance of the power pins. The I/O pins will have been precharged by the on-board power controller circuit. The BD_SEL# pins are the last inputs to be driven to their true state. In Basic or Full Hot Swap systems, these inputs are connected to ground when the short pins are mated. If the system supports High Availability Hot Swap, the BD_SEL# pins can be actively driven by the CompactPCI system controller. Once the board is fully inserted and power is supplied to the rest of the subsystem, the power controller may de-assert the local reset signal. The VMICPCI-7699 initialization process will begin, and once the on-board 21554 hot swap controller has been initialized by a serial ROM load, it will drive the CPCI ENUM# signal to the System Controller or Host. Host software may then clear the CPCI ENUM# status bit, configure the 21554, and load any required subsystem drivers. The board will then be fully initialized.

The card removal process operates in the opposite sequence. As the ejector handle is rotated, the ejector switch will open causing a change of state that will activate the CPCI ENUM# signal. The host may then bring the VMICPCI-7699 unit into a known software state and turn on the blue front panel LED to signify that it is now safe to proceed with card removal. In High Availability systems, the Host can actively de-assert the BD_SEL# signal, removing power from all but the backend circuitry. For all other forms of hot swap, the action of card removal disconnects the BD_SEL# (short pins) from ground.

CAUTION: In order to Hot Swap the VMICPCI-7699 the user must have a Compact PCI backplane that supports Hot Swap. If the VMICPCI-7699 is inserted into or extracted from a powered backplane that does not support Hot Swap, damage can occur to the unit.

I²C Support

The VMICPCI-7699 supports the I²C-bus and can operate as a I²C-bus master or slave per the I²C-bus specification, version 2.0, developed by Philips Semiconductor. Communication over the I²C-bus is accomplished through the use of a Philips PCF8584 I²C-bus controller. This controller is capable of communicating on the I²C-bus on a byte-wise basis using interrupt or polled handshaking. The controller supports clock rates of 90kHz, 45kHz, 11kHz, and 1.5kHz. The I²C-bus signals are routed through the VMICPCI-7699's J1 connector as shown in Table 4-15.

Table 4-15 I²C-bus Through J1

Signal Name	Pin
I2C_SCL	B17
I2C_SDA	C17

NOTE: Although the PCF8584 controller supports a 4-wire long-distance mode of operation, this mode is not supported by the VMICPCI-7699.

Communication with the I²C-bus controller is accomplished through 8-bit memory read / write cycles at address 0xD8012 - 0xD8013. Register S1 is located at address 0xD8013 and is the main control / status register for the 8584 controller. This register controls the operation of the device and the function of the register at location 0xD8012, based on the status of bits ES0, ES1, and ES2 as shown in Table 4-16.

Table 4-16 Device and Function Control

Address	Mnemonic	ES0	ES1	ES2	Description
0xD8012	S0 (Own Address)	0	0	0	Address of 8584 when operating in slave mode.
0xD8012	S3 (Interrupt Vector)	0	0	1	Interrupt vector placed on the data bus during interrupt acknowledge cycles.
0xD8012	S2 (Clock Register)	0	1	0	Register used to set operational clock rate and clock rate of I²C-bus.
0xD8012	S0 (Data register)	1	0	0	Data I/O Register during normal I²C-bus operation.
0xD8013	S1 (Control/Status Register)	X	0	X	Control/Status register for 8584 controller.

The controller can issue interrupts to the VMICPCI-7699 when handshaking on the I^2C-bus. The controller actually shares IRQ5 which is also assigned to the on-board timers. When the I^2C-bus controller drives IRQ5 active, software must service and then clear the interrupt. Software can determine the cause of the interrupt by reading the PIN bit of the PCF8584's status register, and reading the timer interrupt status register of the timers. By comparing these values, software can identify the source of the interrupt and service the interrupt appropriately. See Chapter 4 of this manual for more information on the timers use of IRQ5.

For more information related to programming the PCF8584 I^2C-bus controller, see the *PCF8584 I^2C-bus Controller Datasheet* available from Philips Semiconductor.

Geographical Addressing Support

The VMICPCI-7699 supports the ability to sense geographical addressing on the CPCI backplane. A CPCI backplane that supports geographical addressing will ground the geographical address pins in a pattern that is related to the slot number. The geographical address can be detected by reading from memory location 0xD8010. The data at this location is related to the geographical address pins on the CPCI backplane as shown in Table 4-17.

Table 4-17 Geographical Addressing

Register Bit	GA[] Signal	CPCI Pin
0 (LSB)	GA[0]	J2-E22
1	GA[1]	J2-D22
2	GA[2]	J2-C22
3	GA[3]	J2-B22
4	GA[4]	J2-A22
5-7	Reserved	None

ENUM# Hot Swap LED Monitor Function

The VMICPCI-7699 contains circuitry that monitors status of the CPCI ENUM# signal or the status of the Hot Swap controller's front panel LED. This selection is made by jumper E9 and is monitored via the PIIX4 GP11 pin. Refer to the *Intel 82371EB PCI-to-ISA/IDE Xcelerator (PIIX4)* manual (referenced in the overview section of this manual) for information on programing the general purpose I/O registers.

CHAPTER

5

Maintenance

If a VMIC product malfunctions, please verify the following:

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

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

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

Maintenance Prints

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

Appendix

A

Connector Pinouts

Contents

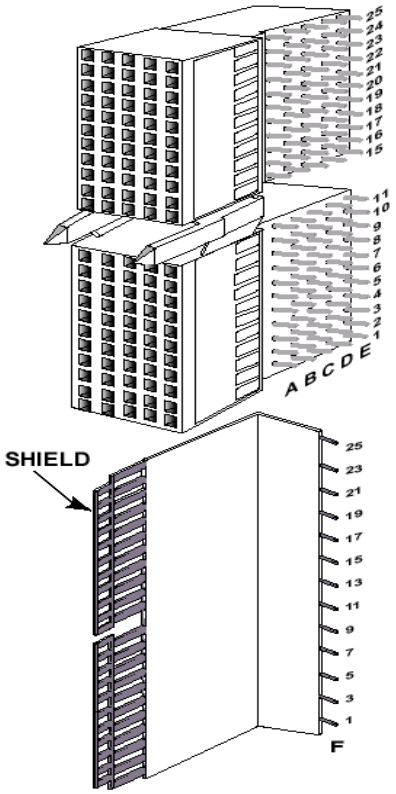
Introduction

The VMICPCI-7699 CompactPCI Peripheral SBC has several connectors for its I/O ports. Wherever possible, the VMICPCI-7699 uses connectors and pinouts typical for any desktop PC. This ensures maximum compatibility with a variety of systems.

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

J1 Connector Pinout

The VMICPCI-7699 utilitizes a high-density 110-pin, low inductance, and controlled impedance connector. This connector meets the IEC-1076 international standard for CompactPCI connectors. An additional external metal shield is required. The large number of ground pins ensures adequate shielding and grounding for low ground bounce and reliable operation in noisy environments. The key prevents misalignment of the board when installing in the chassis. Figure A-1 below depicts the J1 connector and the connector pinout.

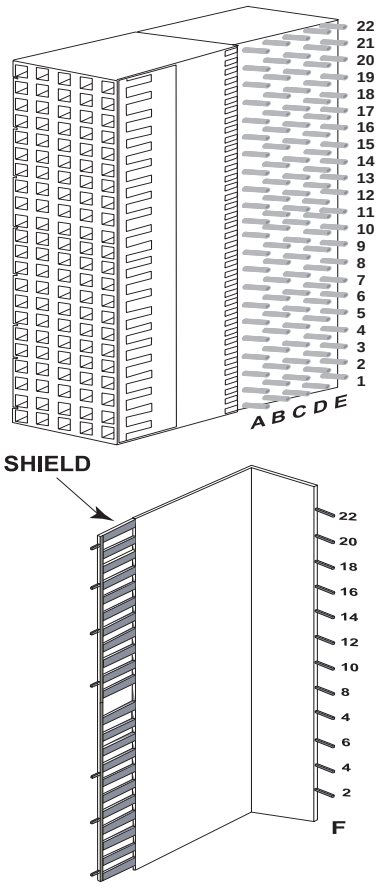


Pin No.	Row A	Row B	Row C	Row D	Row E	Row F
25	+5 V	N/C	C_ENUM#	CPCI_3.3	+5 V	GND
24	C_AD[1]	+5 V	CPCI_VIO	C_AD[0]	N/C	N/C
23	CPCI_3.3	C_AD[4]	C_AD[3]	+5 V	C_AD[2]	GND
22	C_AD[7]	GND	CPCI_3.3	C_AD[6]	C_AD[5]	N/C
21	CPCI_3.3	C_AD[9]	C_AD[8]	GND	C_C/BE0#	GND
20	C_AD[12]	GND	CPCI_VIO	C_AD[11]	C_AD[10]	N/C
19	CPCI_3.3	C_AD[15]	C_AD[14]	GND	C_AD[13]	GND
18	C_SERR#	GND	CPCI_3.3	C_PAR	C_C/BE1#	N/C
17	CPCI_3.3	I2C_SCL	I2C_SDA	GND	C_PERR#	GND
16	C_DEVSEL#	GND	CPCI_VIO	C_STOP#	C_LOCK#	N/C
15	CPCI_3.3	C_FRAME	C_IRDY#	C_BD_SEL#	C_TRDY#	GND
12 through 14 are lost to the keying area						
11	C_AD[18]	C_AD[17]	C_AD[16]	GND	C_C/BE2#	GND
10	C_AD[21]	GND	CPCI_3.3	C_AD[20]	C_AD[19]	N/C
9	C_CBE3#	C_IDSEL	C_AD[23]	GND	C_AD[22]	GND
8	C_AD[26]	GND	CPCI_VIO	C_AD[25]	C_AD[24]	N/C
7	C_AD[30]	C_AD[29]	C_AD[28]	GND	C_AD[27]	GND
6	C_REQ#	GND	CPCI_3.3	C_CLK	C_AD[31]	N/C
5	N/C	N/C	C_RST#	GND	C_GNT#	GND
4	N/C	C_HEALTHY#	CPCI_VIO	N/C	N/C	N/C
3	C_INTA#	C_INTB#	C_INTC#	+5 V	C_INTD#	GND
2	C_TCK	+5 V	C_TMS	N/C	C_TDI	N/C
1	+5 V	-12 V	C_TRST#	+12 V	+5 V	GND

Figure A-1 J1 Connector and Pinout

J2 Connector Pinout

The VMICPCI-7699 J2 connector is a 2mm “Hard Metric” CompactPCI connector, with 5 rows, of 22 pins each. J2 is required for system slot CPUs. An additional external metal shield, labeled row F, is also used. This connector’s controlled impedance minimizes unwanted signal reflections. Figure A-2 illustrates the J2 connector and the connector pinout.



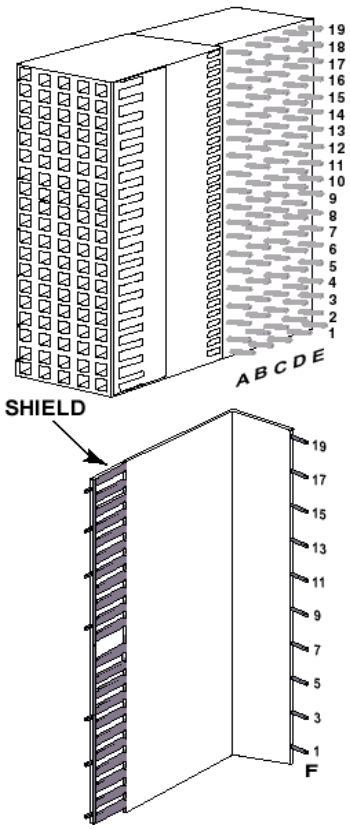
Pin No.	Row A	Row B	Row C	Row D	Row E	Row F
22	GA4	GA3	GA2	GA1	GA0	GND
21	N/C	GND	N/C	N/C	N/C	N/C
20	N/C	GND	N/C	GND	N/C	GND
19	GND	GND	N/C	N/C	N/C	N/C
18	N/C	N/C	N/C	GND	N/C	GND
17	N/C	GND	N/C	N/C	N/C	N/C
16	N/C	N/C	N/C	GND	N/C	GND
15	N/C	GND	N/C	N/C	N/C	N/C
14	N/C	N/C	N/C	GND	N/C	GND
13	N/C	GND	CPCI_VIO	N/C	N/C	N/C
12	N/C	N/C	N/C	GND	N/C	GND
11	N/C	GND	CPCI_VIO	N/C	N/C	N/C
10	N/C	N/C	N/C	GND	N/C	GND
9	N/C	GND	CPCI_VIO	N/C	N/C	N/C
8	N/C	N/C	N/C	GND	N/C	GND
7	N/C	GND	CPCI_VIO	N/C	N/C	N/C
6	N/C	N/C	N/C	GND	N/C	GND
5	N/C	GND	CPCI_VIO	N/C	N/C	N/C
4	CPCI_VIO	N/C	N/C	GND	N/C	GND
3	N/C	GND	N/C	N/C	N/C	N/C
2	N/C	N/C	GND	N/C	N/C	GND
1	N/C	GND	N/C	N/C	N/C	N/C

* CPCI_VIO - The VMICPCI-7699 is a universal VIO design.
* The VMICPCI-7699 supports a 32-bit CPCI bus.
* J2 Row C pins 15 and 16 are not supported.

Figure A-2 J2 Connector and Pinout

J3 Connector Pinout

The J3 connector is a 5 row, 19 pins each, 2mm “Hard Metric” CompactPCI connector. An additional external metal shield is also used, labeled row F. Figure A-3 illustrates the J3 connector and the connector pinout. This connector is used to route the I/O signals of the PMC Site #2, the serial, and floppy drive signals to the backplane I/O.



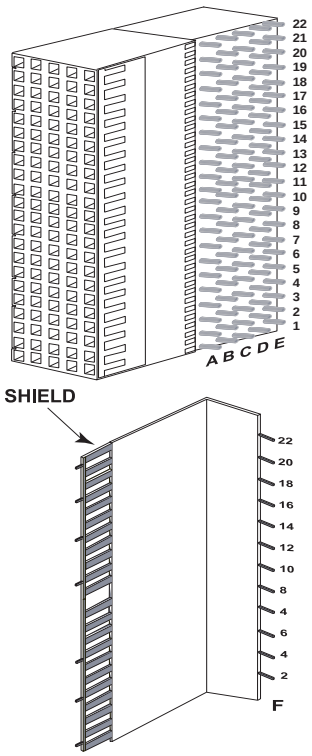
Pin No.	Row A	Row B	Row C	Row D	Row E	Row F
19	WGATE#	WDATA	STEP#	WPT#	DSKCHG#	GND
18	DRVSB#	DRATE0	TRK0#	RDATA#	DIR#	GND
17	MOTEA#	DRVSA#	INDEX#	SIDE1#	MOTEB#	GND
16	REDWC#	SP1_RX	SP1_RTS#	SP1_R1	SP1_DTR#	GND
15	VIO	SP1_TX	SP1_CTS#	SP1_DC0#	SP1_DSR#	GND
14	VCC_3.3	VCC_3.3	VCC_3.3	VCC_5.0	VCC_5.0	GND
13	J12_5	J12_4	J12_3	J12_2	J12_1	GND
12	J12_10	J12_9	J12_8	J12_7	J12_6	GND
11	J12_15	J12_14	J12_13	J12_12	J12_11	GND
10	J12_20	J12_19	J12_18	J12_17	J12_16	GND
9	J12_25	J12_24	J12_23	J12_22	J12_21	GND
8	J12_30	J12_29	J12_28	J12_27	J12_26	GND
7	J12_35	J12_34	J12_33	J12_32	J12_31	GND
6	J12_40	J12_39	J12_38	J12_37	J12_36	GND
5	J12_45	J12_44	J12_43	J12_42	J12_41	GND
4	J12_50	J12_49	J12_48	J12_47	J12_46	GND
3	J12_55	J12_54	J12_53	J12_52	J12_51	GND
2	J12_60	J12_59	J12_58	J12_57	J12_56	GND
1	VIO	J12_64	J12_63	J12_62	J12_61	GND

Figure A-3 J3 Connector and Pinout

NOTE: Backplane designs should route P3 signals straight through to rear J3. The VMIACC-0577 board can then be utilized.

J5 Connector Pinout

The VMICPCI-7699 J5 connector is a 2mm “Hard Metric” CompactPCI connector, with 5 rows of 22 pins each. An additional external metal shield is also used, labeled row F. This connector is used to route the I/O signals of PMC Site #1, and the IDE, serial, and USB signals to the backplane I/O. This connector’s controlled impedance minimizes unwanted signal reflections. Figure A-4 illustrates the J5 connector and the connector pinout.



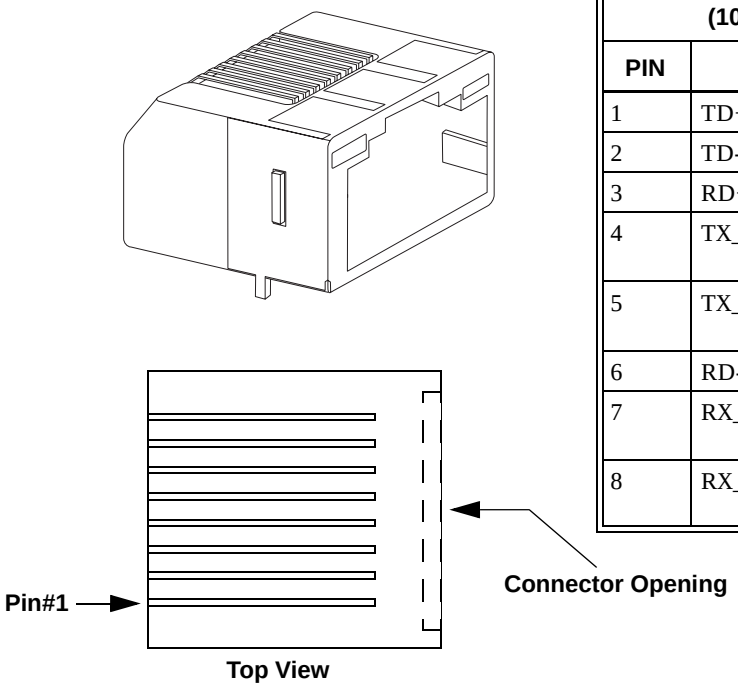
Pin No.	Row A	Row B	Row C	Row D	Row E	Row F
22	J8_5	J8_4	J8_3	J8_2	J8_1	GND
21	J8_10	J8_9	J8_8	J8_7	J8_6	GND
20	J8_15	J8_14	J8_13	J8_12	J8_11	GND
19	J8_20	J8_19	J8_18	J8_17	J8_16	GND
18	J8_25	J8_24	J8_23	J8_22	J8_21	GND
17	J8_30	J8_29	J8_28	J8_27	J8_26	GND
16	J8_35	J8_34	J8_33	J8_32	J8_31	GND
15	J8_40	J8_39	J8_38	J8_37	J8_36	GND
14	J8_45	J8_44	J8_43	J8_42	J8_41	GND
13	J8_50	J8_49	J8_48	J8_47	J8_46	GND
12	J8_55	J8_54	J8_53	J8_52	J8_51	GND
11	J8_60	J8_59	J8_58	J8_57	J8_56	GND
10	VIO	J8_64	J8_63	J8_62	J8_61	GND
9	DDP[0]	DOP[1]	DOP[2]	DOP[3]	DOP[4]	GND
8	DDP[5]	DOP[6]	DOP[7	DOP[8]	DOP[9]	GND
7	DDP[10]	DOP[11]	DOP[12]	DOP[13]	DOP[14]	GND
6	DDP[15]	IDEIOR0#	IDEIORDY0	IDEIOW0#	IDECS01#	GND
5	HD_ACTA#	IDECS03#]	IRQ[14]	IDEDACK0#	IDEDRQ0	GND
4	IDERST#	DAP1	DAP1	DAP2	USBOC0#_C	GND
3	N/C	SP0_RI	SP0_TX	C_HEALTHY	USBP0-R_C	GND
2	N/C	SP0_DC0#	SP0_RX	SP0_CTS#	USBP0+R_C	GND
1	VCC_M12.0	SP0_DTR#	SP0_DSR#	SP0_RTS#	VCC_12.0	GND

Notes: * signal can be tied to GND through Jumper E22.
** Signal can be tied to +5 V through Jumper E23.

Figure A-4 J5 Connector and Pinout

Ethernet Connector Pinout (J6)

The pinout diagram for the Ethernet 10BaseT and 100BaseTx connector is shown in Figure A-5.

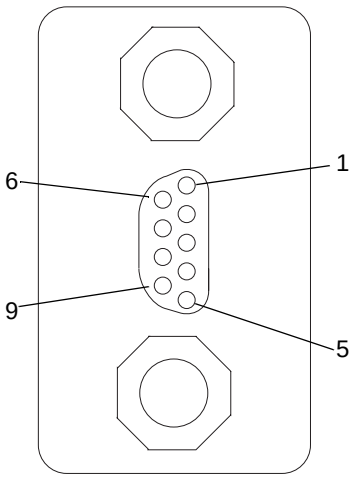


ETHERNET CONNECTOR (10BaseT, 100BaseTx)		
PIN	**Signal Name**	
1	TD+	Transmit Data
2	TD-	Transmit Data
3	RD+	Receive Data
4	TX_CT_OUT	Transmit Center Tap Out
5	TX_CT_OUT	Transmit Center Tap Out
6	RD-	Receive Data
7	RX_CT_OUT	Receive Center Tap Out
8	RX_CT_OUT	Receive Center Tap Out

Figure A-5 Ethernet Connector and Pinout

Video Connector Pinout (P3)

The video port uses a micro DB9 connector. Figure A-6 illustrates the connector and pinout.



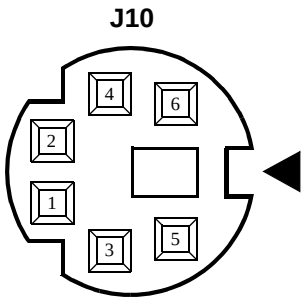
Video Connector		
Pin	**Direction**	**Function**
1	Out	Red
2	Out	Green
3	Out	Blue
4	Out	Horizontal Sync
5	Out	Vertical Sync
6		5V
7	I/O	DDC Data
8	I/O	DDC Clock
9		Ground

Figure A-6 Video Connector and Pinout

NOTE: An adapter to convert the micro DB9 connector to a standard female high-density DB15 connector is available as an option (order # 360-000168-004).

Keyboard and Mouse Connectors and Pinout (J7)

The keyboard and mouse connectors are standard 6-pin female mini-DIN PS/2 connectors as shown in Figure A-7 and Table A-1.



Keyboard/Mouse Connector*		
Pin	**Dir**	**Function**
1	In/Out	Mouse Data
2	In/Out	Keyboard Data
3		Ground
4		+5 V
5	Out	Mouse Clock
6	Out	Keyboard Clock
Shield		Chassis Ground
An adapter cable is included with the VMICPCI-7699 to separate the keyboard and mouse connector.		

Figure A-7 Keyboard/Mouse Connector and Pinout

Table A-1 Keyboard/Mouse Y Splitter Cable

Keyboard			Mouse		
Pin	**Dir**	**Function**	**Pin**	**Dir**	**Function**
1	In/Out	Keyboard Data	1	In/Out	Mouse Data
2		Unused	2		Unused
3		Ground	3		Ground
4		+5 V	4		+5 V
5	Out	Keyboard Clock	5	Out	Mouse Clock
6		Unused	6		Unused
Shield		Chassis Ground	Shield		Chassis Ground

PMC Connector Pinout

PMC #1 (J11)/PMC #2 (J15) Connector and Pinout

The PCI Mezzanine Card (PMC) carries the same signals as the PCI standard; however, the PMC standard uses a completely different form factor. Table A-2 through A-5 are the pinouts for the PMC connectors (J8, J10, J11, J12, J14, and J15).

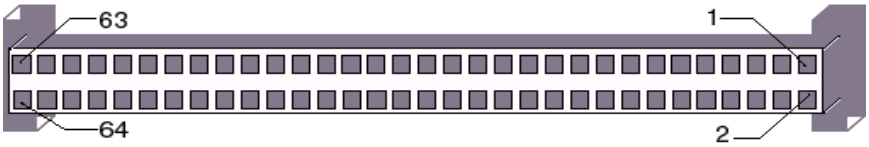


Table A-2 PMC #1 (J11)/PMC #2 (J15) Connector Pinout

PMC Connector (J11/J15)				PMC Connector (J11/J15)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	GND	2	-12	33	FRAME#	34	GND
3	GND	4	INTC#	35	GND	36	IRDY#
5	INTD#	6	INTA#	37	DEVSEL#	38	+5 V
7	BMODE1A	8	+5 V	39	GND	40	LOCK#
9	INTB#	10	NC	41	SDONE#	42	NC
11	GND	12	NC	43	PAR	44	GND
13	CLK	14	GND	45	+5 V	46	AD[15]
15	GND	16	GNT#	47	AD[12]	48	AD[11]
17	REQ#	18	+5 V	49	AD[9]	50	+5 V
19	+5 V	20	AD[31]	51	GND	52	C/BE0#
21	AD[28]	22	AD[27]	53	AD[6]	54	AD[5]
23	AD[25]	24	GND	55	AD[4]	56	GND
25	GND	26	C/BE3#	57	+5 V	58	AD[3]
27	AD[22]	28	AD[21]	59	AD[2]	60	AD[1]
29	AD[19]	30	+5 V	61	AD[0]	62	+5 V
31	+5 V	32	AD[17]	63	GND	64	REQ64#

PMC #1 (J10)/ PMC #2 (J14) Connector and Pinout

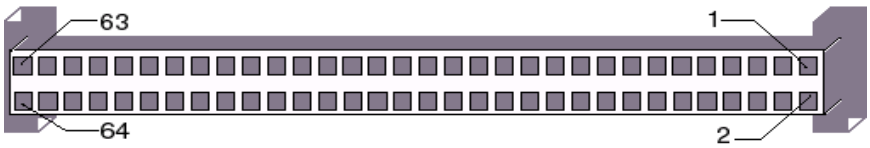


Table A-3 PMC #1 (J10)/PMC #2 (J14) Connector Pinout

PMC Connector (J10/J14)				PMC Connector (J10/J14)			
Left Side		Right Side		Left Side		Right Side	
Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	+12 V	2	+5 V	33	GND	34	NC
3	GND	4	NC	35	TRDY	36	+3.3 V
5	+5 V	6	GND	37	GND	38	STOP#
7	GND	8	NC	39	PERR#	40	GND
9	NC	10	NC	41	+3.3 V	42	SERR#
11	PRSNT2	12	+3.3 V	43	C/BE1#	44	GND
13	RST#	14	GND	45	AD[14]	46	AD[13]
15	+3.3 V	16	GND	47	GND	48	AD[10]
17	NC	18	GND	49	AD[8]	50	+3.3 V
19	AD[30]	20	AD[29]	51	AD[7]	52	NC
21	GND	22	AD[26]	53	+3.3 V	54	NC
23	AD[24]	24	+3.3 V	55	NC	56	GND
25	IDSEL	26	AD[23]	57	NC	58	NC
27	+3.3 V	28	AD[20]	59	GND	60	NC
29	AD[18]	30	GND	61	ACK64#	62	+3.3 V
31	AD[16]	32	C/BE2#	63	GND	64	NC

PMC #1 (J8) Connector and Pinout

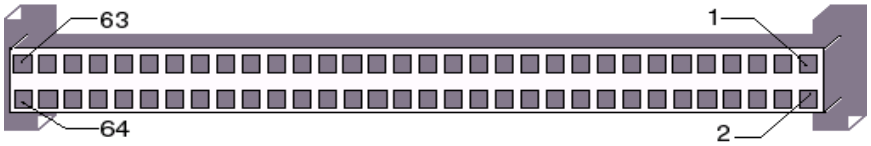


Table A-4 PMC #1 J8 Connector Pinout

PMC Connector (J8)						PMC Connector (J8)					
Left Side			Right Side			Left Side			Right Side		
Pin	Name	Connected To	Pin	Name	Connected To	Pin	Name	Connected To	Pin	Name	Connected To
1	J8-1	J5 pin E22	2	J8-2	J5 pin D22	33	J8-33	J5 pin C16	34	J8-34	J5 pin B16
3	J8-3	J5 pin C22	4	J8-4	J5 pin B22	35	J8-35	J5 pin A16	36	J8-36	J5 pin E15
5	J8-5	J5 pin A22	6	J8-6	J5 pin E21	37	J8-37	J5 pin D15	38	J8-38	J5 pin C15
7	J8-7	J5 pin D21	8	J8-8	J5 pin C21	39	J8-39	J5 pin B15	40	J8-40	J5 pin A15
9	J8-9	J5 pin B21	10	J8-10	J5 pin A21	41	J8-41	J5 pin E14	42	J8-42	J5 pin D14
11	J8-11	J5 pin E20	12	J8-12	J5 pin D20	43	J8-43	J5 pin C14	44	J8-44	J5 pin B14
13	J8-13	J5 pin C20	14	J8-14	J5 pin B20	45	J8-45	J5 pin A14	46	J8-46	J5 pin E13
15	J8-15	J5 pin A20	16	J8-16	J5 pin E19	47	J8-47	J5 pin D13	48	J8-48	J5 pin C13
17	J8-17	J5 pin D19	18	J8-18	J5 pin C19	49	J8-49	J5 pin B13	50	J8-50	J5 pin A13
19	J8-19	J5 pin B19	20	J8-20	J5 pin A19	51	J8-51	J5 pin E12	52	J8-52	J5 pin D12
21	J8-21	J5 pin E18	22	J8-22	J5 pin D18	53	J8-53	J5 pin C12	54	J8-54	J5 pin B12
23	J8-23	J5 pin C18	24	J8-24	J5 pin B18	55	J8-55	J5 pin A12	56	J8-56	J5 pin E11
25	J8-25	J5 pin A18	26	J8-26	J5 pin E17	57	J8-57	J5 pin D11	58	J8-58	J5 pin C11
27	J8-27	J5 pin D17	28	J8-28	J5 pin C17	59	J8-59	J5 pin B11	60	J8-60	J5 pin A11
29	J8-29	J5 pin B17	30	J8-30	J5 pin A17	61	J8-61	J5 pin E10	62	J8-62	J5 pin D10
31	J8-31	J5 pin E16	32	J8-32	J5 pin D16	63	J8-63	J5 pin C10	64	J8-64	J5 pin B10

PMC #2 (J12) Connector and Pinout

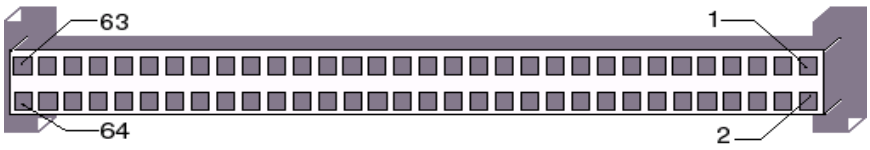


Table A-5 PMC #2 J12 Connector Pinout

PMC Connector (J12)						PMC Connector (J12)					
Left Side			Right Side			Left Side			Right Side		
Pin	Name	Connected To	Pin	Name	Connected To	Pin	Name	Connected To	Pin	Name	Connected To
1	J12-1	J3 pin E22	2	J12-2	J3 pin D22	33	J12-33	J3 pin C16	34	J12-34	J3 pin B16
3	J12-3	J3 pin C22	4	J12-4	J3 pin B22	35	J12-35	J3 pin A16	36	J12-36	J3 pin E15
5	J12-5	J3 pin A22	6	J12-6	J3 pin E21	37	J12-37	J3 pin D15	38	J12-38	J3 pin C15
7	J12-7	J3 pin D21	8	J12-8	J3 pin C21	39	J12-39	J3 pin B15	40	J12-40	J3 pin A15
9	J12-9	J3 pin B21	10	J12-10	J3 pin A21	41	J12-41	J3 pin E14	42	J12-42	J3 pin D14
11	J12-11	J3 pin E20	12	J12-12	J3 pin D20	43	J12-43	J3 pin C14	44	J12-44	J3 pin B14
13	J12-13	J3 pin C20	14	J12-14	J3 pin B20	45	J12-45	J3 pin A14	46	J12-46	J3 pin E13
15	J12-15	J3 pin A20	16	J12-16	J3 pin E19	47	J12-47	J3 pin D13	48	J12-48	J3 pin C13
17	J12-17	J3 pin D19	18	J12-18	J3 pin C19	49	J12-49	J3 pin B13	50	J12-50	J3 pin A13
19	J12-19	J3 pin B19	20	J12-20	J3 pin A19	51	J12-51	J3 pin E12	52	J12-52	J3 pin D12
21	J12-21	J3 pin E18	22	J12-22	J3 pin D18	53	J12-53	J3 pin C12	54	J12-54	J3 pin B12
23	J12-23	J3 pin C18	24	J12-24	J3 pin B18	55	J12-55	J3 pin A12	56	J12-56	J3 pin E11
25	J12-25	J3 pin A18	26	J12-26	J3 pin E17	57	J12-57	J3 pin D11	58	J12-58	J3 pin C11
27	J12-27	J3 pin D17	28	J12-28	J3 pin C17	59	J12-59	J3 pin B11	60	J12-60	J3 pin A11
29	J12-29	J3 pin B17	30	J12-30	J3 pin A17	61	J12-61	J3 pin E10	62	J12-62	J3 pin D10
31	J12-31	J3 pin E16	32	J12-32	J3 pin D16	63	J12-63	J3 pin C10	64	J12-64	J3 pin B10

Appendix

B

System Driver Software

Contents

Introduction

The VMICPCI-7699 provides high-performance video and Local Area Network (LAN) access by means of on-board PCI-based adapters and associated software drivers. The PCI-based video adapter used on the VMICPCI-7699 is the Chips and Technology 69000. High-performance LAN operation, including 10BaseT and 100BaseTx, is provided by the Intel GD82559ER PCI Ethernet Controller chip.

To optimize performance of each of these PCI-based subsystems, the VMICPCI-7699 is provided with software drivers compatible with Windows 98 SE and Windows NT operating systems. The following paragraphs provide instructions for loading and installing the adapter software.

Driver Software Installation

In order to properly use the Video and LAN adapters of the VMICPCI-7699, the user must install the driver software located on the distribution diskettes provided with the unit. Detailed instructions for installation of the drivers during installation of Windows 98 SE or Windows NT (Versions 4.0) operating systems are described in the following sections.

Windows 98 SE (Second Edition)

1. Begin installation of Windows 98 SE, following the instructions provided by the Windows 98 SE manual.
2. When you reach the 'Windows 98 SE Setup Wizard Screen', choose 'Typical' under 'Setup Options' and then click 'Next'.
3. Under the 'Windows Components Screen', select 'Install The Most Common Components' and then click 'Next'.
4. Continue with the installation until Windows 98 SE is completely installed and has rebooted.
5. Right-click the 'My Computer' icon, then click 'Properties'.
6. Click the 'Device Manager' tab. Highlight 'PCI Ethernet Controller' under 'Other Devices' by clicking on it once. Click 'Remove.' Confirm the removal of the device by clicking on OK. Click 'Close' in the 'System Properties' window.
7. Ensure that the Windows 98 SE CD-ROM is in the CD-ROM drive. Click the 'Start' button, then click 'Settings' then 'Control Panel.'
8. Double-click the 'Network' icon. Click 'Add.' Highlight 'Adapter' by clicking on it once, then click 'Add.'
9. Insert the 320-500051-004 diskette in the floppy drive, then click 'Have disk....'
10. Ensure A:\ is entered in the box labeled 'Copy Manufacturer's Files from:'. Click OK.
11. Highlight 'Intel GD82559ER PCI Adapter' by clicking on it once, then click OK.
12. When asked to insert the disk labeled 'Intel GD82559ER PCI Adapter', simply leave the 320-500051-004 diskette in the floppy drive and click OK.
13. In the next window you will be told that the file 'e100ent.sys' cannot be found. Type A:\ in the 'Copy files from:' box, click OK.
14. A message will appear stating that you have selected a Plug-n-Play adapter. The message will ask that you turn the computer off, install the adapter and restart the computer. Click OK to disregard this message, then click OK in the 'Network' window.
15. Windows 98 SE will copy a few files from the CD-ROM and ask if you want to reboot. Remove the 320-500051-004 diskette and click 'Yes'. Allow the system to restart.
16. Windows 98 SE will now recognize the Intel GD82559ER PCI Adapter and install the proper drivers. When asked again if you want to restart, click 'Yes.' Allow the system to restart. The Intel GD82559ER PCI Adapter driver is now installed.

Installing the Chips and Technology 69000 Video Driver

To install the 69000 Video driver follow the steps below.

1. Insert disk 320-500051-002 into the floppy drive.
2. Click ‘Start’, then click ‘Run’. In the ‘Open:’ box, type A:\w98600 and click OK.
3. Follow the on-screen instructions until you are asked to restart the computer. Remove the 320-50051-002 diskette from the floppy drive and click ‘Finish.’ Allow the system to restart. The Chips and Technology 69000 driver is now installed.

Windows NT (Version 4.0)

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

1. Follow the normal Windows NT 4.0 installation until you reach the **Windows 'NT** Workstation Setup' window which states that 'Windows NT Needs To Know How This Computer Should Participate On A Network'.
2. Click 'This Computer Will Participate On A Network'.
3. Click 'Wired To The Network' and click 'Next'.
4. In the next screen, click 'Select From List'.
5. Click 'Have Disk'.
6. Insert disk **320-500051-004** into drive A.
7. Click OK.
8. In the 'Select OEM Option', choose 'Intel GD82559ER Fast Ethernet Adapter**'**, then click OK.
9. Select the above entry on the displayed list, click Next.
10. Select the 'NetBEUI Protocol', click 'Next'.
11. Click 'Next' to install selected components.
12. Click 'Next' to start the network connection.
13. Step through the remaining screens, providing the data pertinent to your network.
14. Continue through the setup procedure until the 'Detected Display' window appears, click 'OK' to continue.
15. In the 'Display Properties' window, click 'Test'.

NOTE: Windows NT 4.0 does not allow the selection of the Chips and Technology video drivers during initial setup.

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

16. Continue with the procedure to the 'Windows NT Setup' window. Click 'Restart Computer'.
17. When the computer reboots, double-click 'My Computer' icon.
18. Double-click the 'Control Panel' icon in the 'My Computer' window.
19. Double-click the 'Display' icon in the 'Control Panel'.

20. Select the ‘Settings’ tab in the ‘Display Properties’ window, then click ‘Display Type’.
21. In the ‘Display Type’ window, click ‘Change’.
22. In the ‘Change Display’ window, click ‘Have Disk’.
23. Insert disk 320-500051-003 into drive A**.**
24. Click OK.
25. Chips Video Accelerator will be displayed in the ‘Change Display’ window. Click OK.
26. Proceed as directed, removing the driver disk from the floppy drive, and the computer will restart to activate the new settings. When the system reboots, the ‘Invalid Display Settings’ screen will be displayed. Click OK.
27. On the ‘Display Properties’ screen click **Settings**, then click ‘Test’.
28. The ‘Testing Mode’ screen will be displayed. Click Ok. If the bitmap test image is displayed correctly, click Yes, then click OK.

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

Appendix

C

Phoenix BIOS

Contents

Introduction

The VMICPCI-7699 utilizes the BIOS (Basic Input / Output System) in the same manner as other PC / AT compatible computers. This appendix describes the menus and options associated with the VMICPCI-7699 BIOS.

System BIOS Setup Utility

During system bootup, press the F2 key to access the Phoenix BIOS Main screen. From this screen, the user can select any section of the Phoenix (system) BIOS for configuration, such as floppy drive configuration or system memory. The parameters shown throughout this section are the default values.

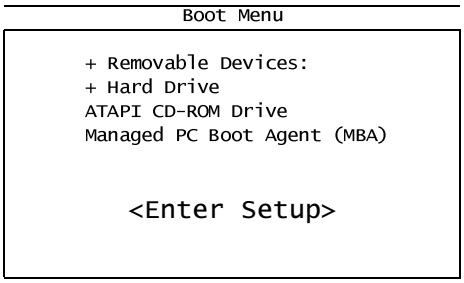
Help Window

The help window on the right side of each menu displays the help text for the currently selected field. It updates as you move the cursor to each field. Pressing F1 or ALT-H on any menu brings up the General Help window that describes the legend keys and their alternates. The scroll bar on the right of any window indicates that there is more than one page of information in the window. Use PGUP and PGDN to display all each page. Pressing HOME and END displays the first and last page. Pressing ENTER displays each page and then exits the window. Press ESC to exit the current window.

First Boot

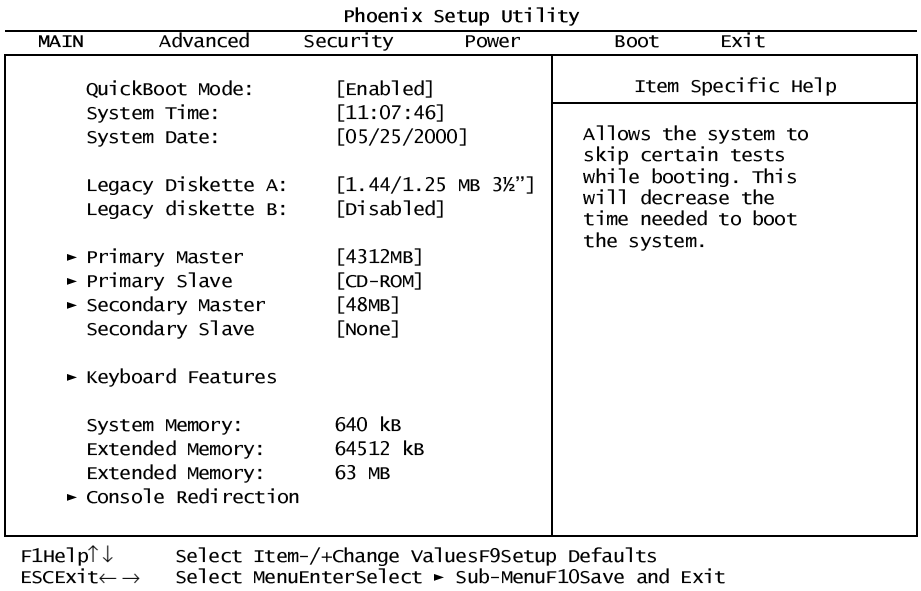
The VMICPCI-7699 has a First Boot menu enabling the user to, on a one time basis, select a drive device to boot from. This feature is useful when installing from a bootable disk. For example, when installing Windows NT from a CD, enter the First Boot menu and using the arrows keys highlight ATAPI CD-ROM Drive. Press ENTER to continue with system boot.

This feature is accessed by pressing the ESC at the very beginning of the boot cycle. The selection made from this screen applies to the current boot only, and will not be used during the next boot-up of the system. If you have trouble accessing this feature, disable the QuickBoot Mode in the Main BIOS setup screen. Exit, saving changes and retry accessing the First Boot menu.



Main Menu

The Main menu allows the user to select QuickBoot, set the system clock and calendar, record disk drive parameters, and set selected functions for the keyboard.



QuickBoot

When enabled, certain checks normally performed during the POST are omitted, decreasing the time required to run the POST. The default is Enabled.

Setting The Time

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

Setting The Date

Press the left or right arrow key to move the cursor to the desired field (month, day, year). Press the PGUP or PGDN key to step through the available choices, or type in the information.

Legacy Diskette

Floppy Drive A

The VMICPCI-7699 supports one floppy disk drive. The options are:

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

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

Floppy Drive B

The VMICPCI-7699 does not support a second floppy drive. The default is Disabled.

Primary Master/Slave

The VMICPCI-7699 is capable of utilizing one IDE hard disk drive on the Primary Master bus. The default setting is Auto. The Primary Slave is assigned to the CD-ROM (if installed).

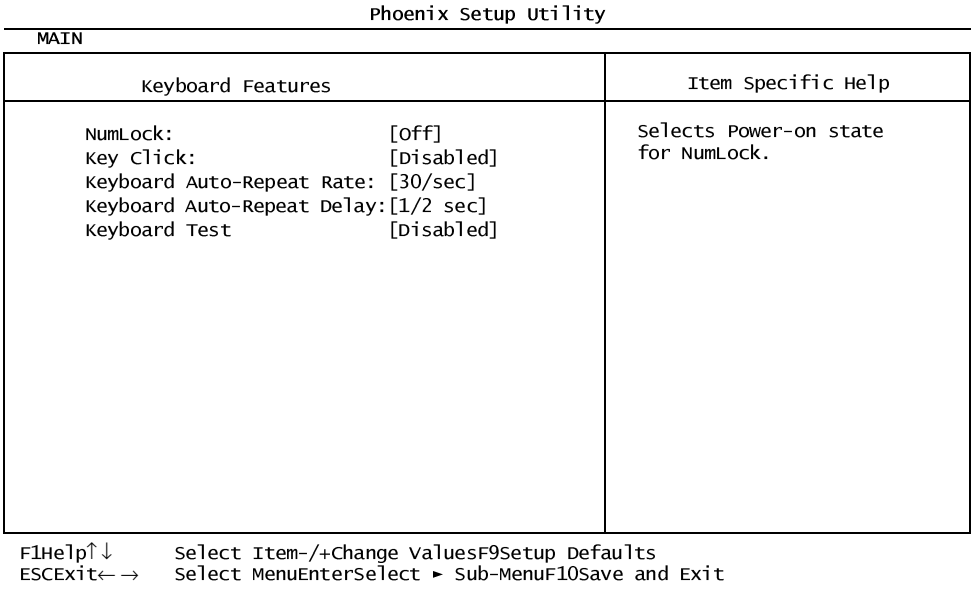
```
                              Phoenix Setup Utility
 MAIN
 ┌──────────────────────────────────────┬──────────────────────────────┐
 │      Primary Master [1350]           │      Item Specific Help      │
 ├──────────────────────────────────────┴──────────────────────────────┤
 │     Type:                    [Auto]     User = you enter            │
 │                                         parameters of hard-disk     │
 │     Multi-Sector Transfers   [16 Sectors] drive installed at this   │
 │     LBA Mode Control         [Enabled]  connection.                 │
 │     32 Bit I/O               [Disabled] Auto = autotypes            │
 │     Transfer Mode:           [FPIO 4 / DMA 2] hard-drive            │
 │     Ultra DMA Mode:          [Mode 2]   installed here.             │
 │                                         1-39 = you select           │
 │                                         pre-determined type of      │
 │                                         hard-drive                  │
 │                                         installed here.             │
 │                                         CD-ROM = a CD-ROM drive     │
 │                                         is installed here.          │
 │                                         ATAPI Removable =           │
 │                                         Removable disk-drive is     │
 │                                         installed here.             │
 └─────────────────────────────────────────────────────────────────────┘
 F1Help↑↓      Select Item-/+Change ValuesF9Setup Defaults
 ESCExit←→     Select MenuEnterSelect ► Sub-MenuF10Save and Exit
```

Secondary Master

The Secondary Master is the resident Flash Disk (if installed). The default setting is Auto.

Keyboard Features

The Keyboard Features allows the user to set several keyboard functions.



NumLock

The NumLock can be set to Auto, On, or Off to control the state of the NumLock key when the system boots. When set to Auto or On, the numeric keypad generates numbers instead of controlling the cursor operations. The default is On.

Key Click

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

Keyboard Auto-Repeat Rate (Chars/Sec)

If the Key Click is enabled this determines the rate a character is repeated when a key is held down. The options are: 30, 26.7, 21.8, 18.5, 13.3, 10, 6, or 2 characters per second. The default is 30.

Keyboard Auto-Repeat Delay (sec)

If the Key Click is enabled, this determines the delay before a character starts repeating when a key is held down. The options are: 1/4, 1/2, 3/4, or 1 second. The default is 1/2.

Keyboard Test

When enabled, this feature will test the keyboard during boot-up.

System Memory

The System Memory field is for informational purposes only and cannot be modified by the user. This field displays the base memory installed in the system.

Extended Memory

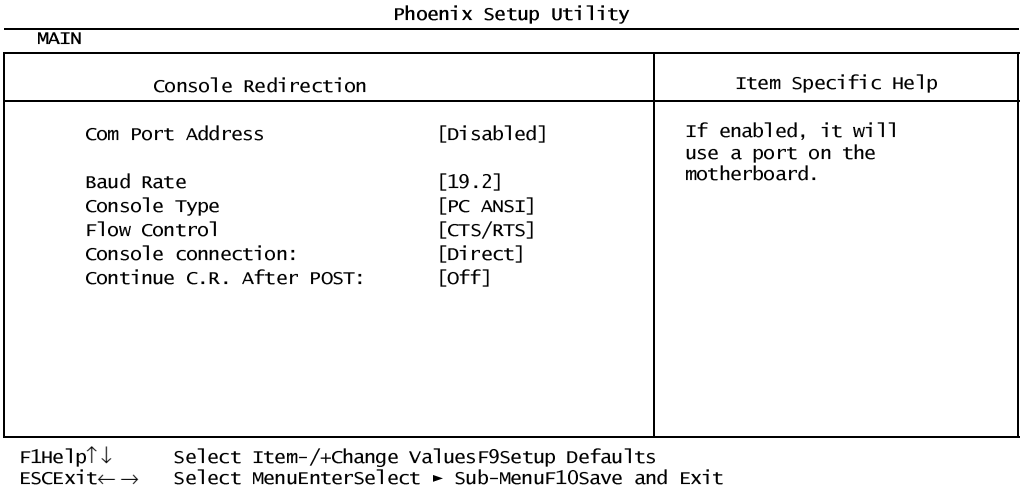
The Extended Memory field is for informational purposes only and cannot be modified by the user. This field displays the total amount of memory installed in the system in Kbytes.

Extended Memory

The Extended Memory field is for informational purposes only and cannot be modified by the user. This field displays the total amount of memory installed in the system in Mbytes.

Console Redirection

Console Redirection allows for remote access and control of the PC functions to a remote terminal via the serial port. Selecting Console Redirection provides additional menus used to configure the console.



Com Port Address

If enabled, it will allow remote access through the serial port. The options are: Disabled, Motherboard Com A, and Motherboard Com B. The default is Disabled.

Baud Rate

Selects a baud rate for the serial port. The options are: 600, 1200, 2400, 4800, 9600, 19.2, 38.4, and 115.2. The default is 19.2.

Console Type

Selects the type of console to be used. The options are: PC ANSI or VT100. The default is PC ANSI.

Flow Control

Enables or disables Flow Control. The options are No Flow Control, XON/XOFF, or CTS/RTS. The default is CTS/RTS.

Console Connection

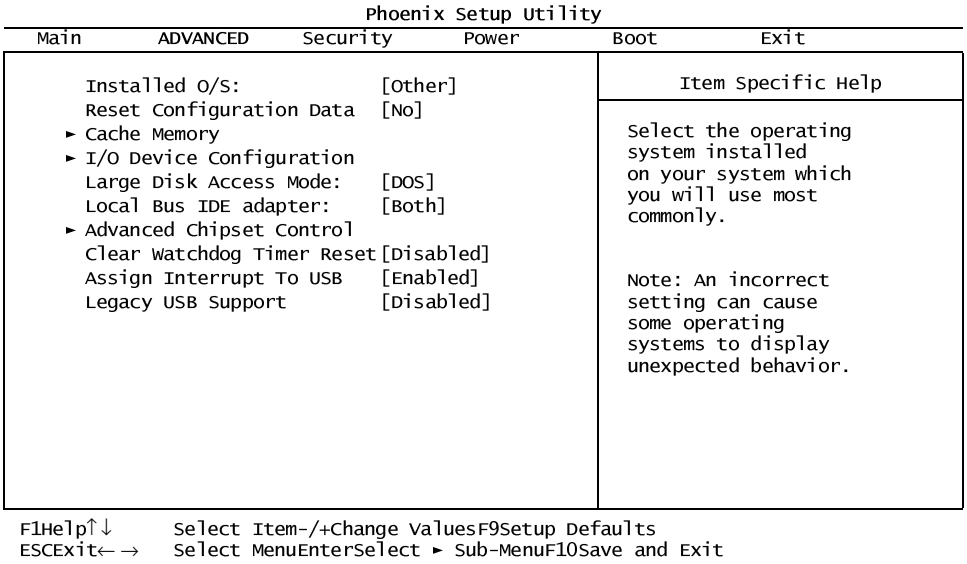
Indicates whether the console is connected directly to the system, or if a modem is being used to connect. The options are: Direct or Via Modem. The default is Direct.

Continue C.R. After POST

This enables console redirection after the operating system has loaded. The options are OFF or ON. The default setting is OFF.

Advanced Menu

Selecting Advanced from the Main menu will display the screen shown below.



Installed O/S

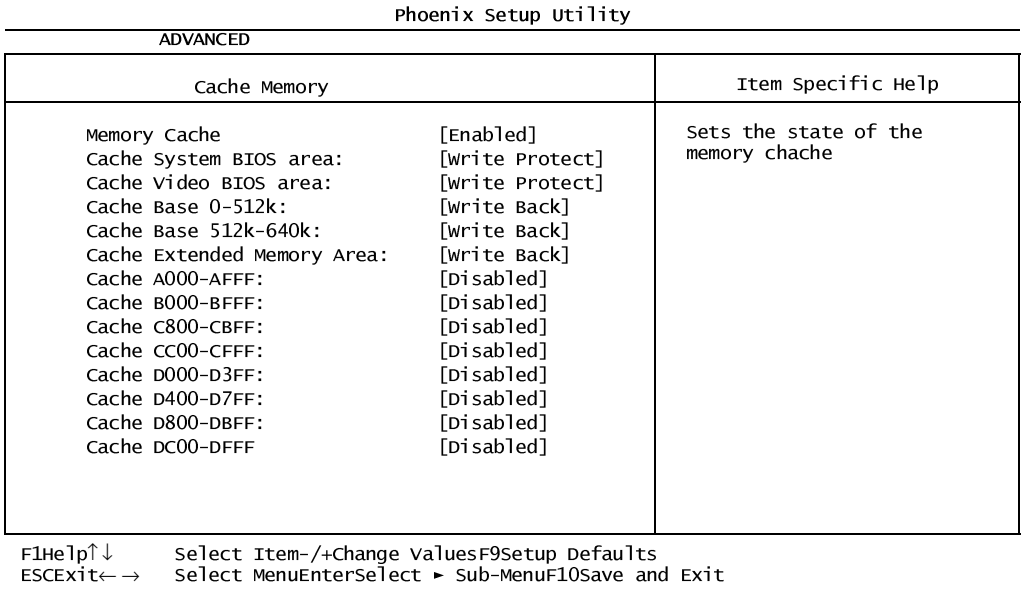
Use this feature to select the operating system to use with your system.

Reset Configuration Data

Select Yes if you want to clear the extended system configuration data. The default is No.

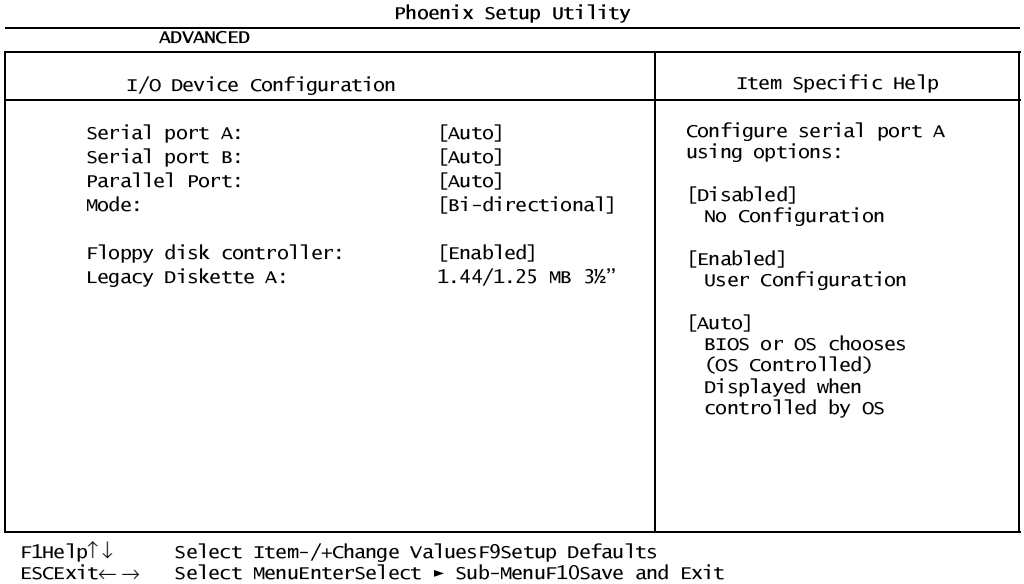
Cache Memory

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



I/O Device Configuration

Select this menu to configure your I/O devices, if required.



Large Disk Access Mode

The options for the Large Disk Access Mode are UNIX Novell Netware or Other.

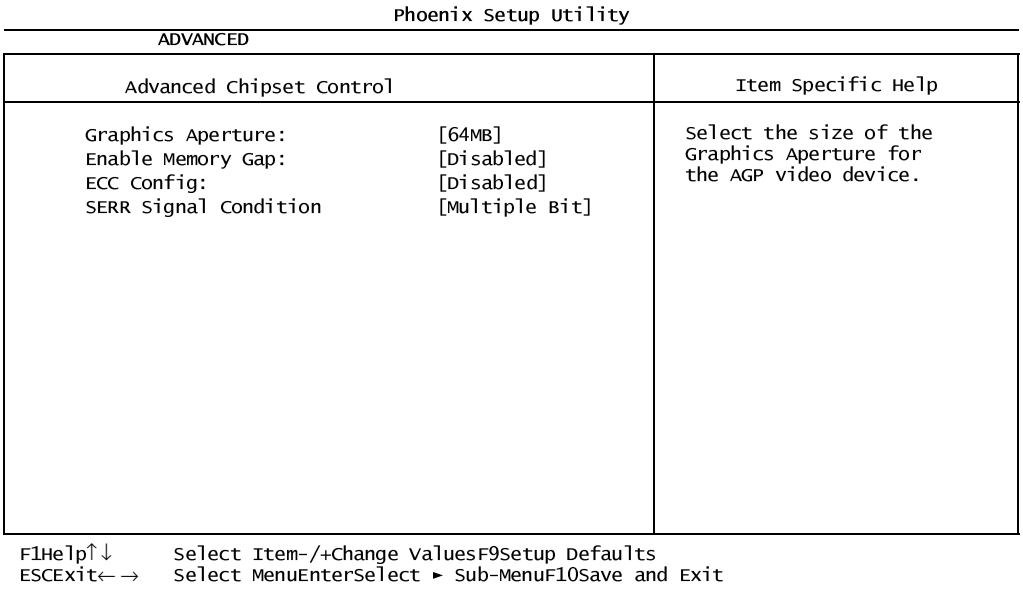
If you are installing new software and the drive fails, change this selection and try again. Different operating systems require different representations of drive geometries. The default is Other.

Local Bus IDE Adapter

This enables or disables the intergrated local bus IDE adapter. The options are: Disabled, Primary, Secondary, or Both. The default is Both.

Advanced Chipset Control

Selecting Advanced Chipset Control opens the menu below. Use this menu to change the values in the chipset register for optimizing your system's performance.



Graphics Aperture

Select the size of the graphics aperture for the AGP video device. The options are: 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, or 256MB. The default is 64MB.

Enable Memory Map

If enabled, turn system RAM off to free address space for use with an option card. Either a 128KB conventional memory gap, starting at 512KB, or a 1MB extended memory gap, starting at 15MB, will be created in the system RAM. The options are Disabled, Conventional, or Extended. The default is Disabled.

ECC Config

If all memory in the system supports ECC (x72), the ECC Config selects from No ECC (Disabled), Checking Only (EC), Checking and Correction (ECC), or Checking, Correction with Scrubbing (ECC Scrub). The default is Disabled.

SERR Signal Configuration

Select ECC error conditions that SERR# will be asserted. The options are: None, Single Bit, Multiple Bit, or Both. The default is Both.

Clear Watchdog Timer Reset

If enabled, after power-up or reset, the BIOS will clear the on-board Watchdog Timer so it will not generate a Watchdog Reset or NMI. The default is Disabled.

Assign Interrupt To USB

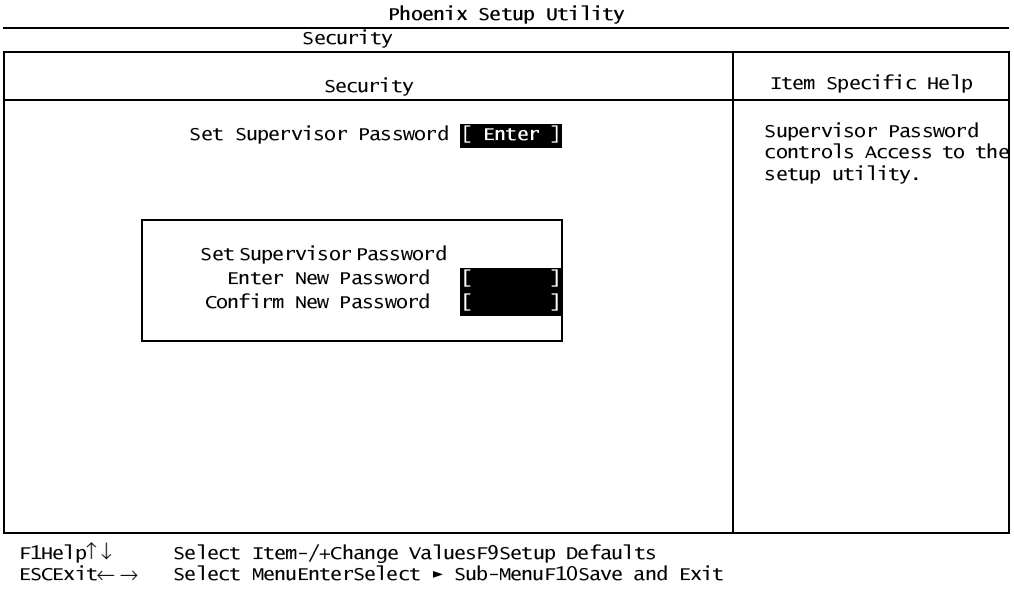
Enabled assigns an interrupt to the USB. The default is Disabled.

Legacy Keyboard/Mouse

Enabled supports a legacy keyboard and mouse on the USB. The default is Disabled.

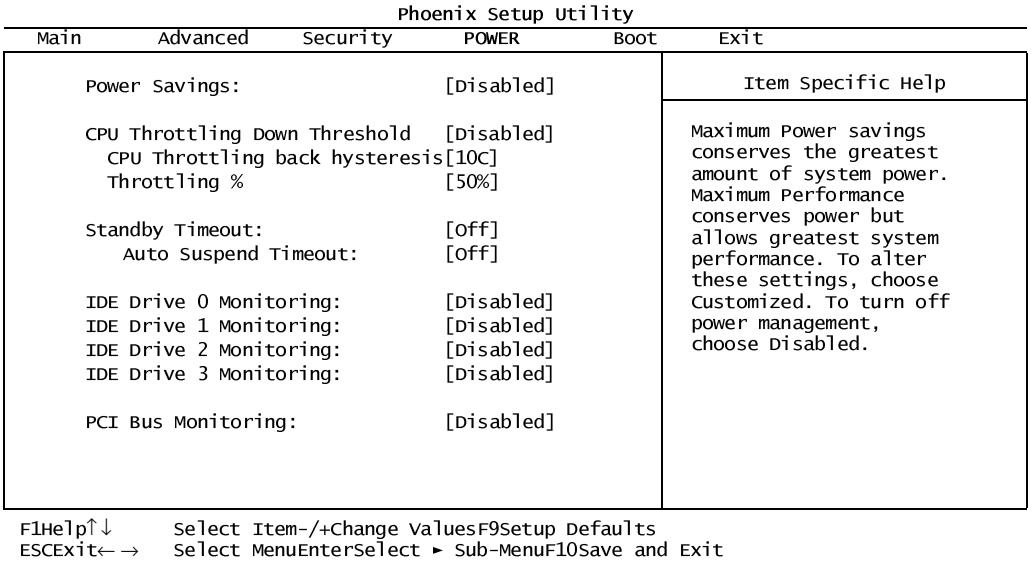
Security

Utilize this screen to set a user password.



Power

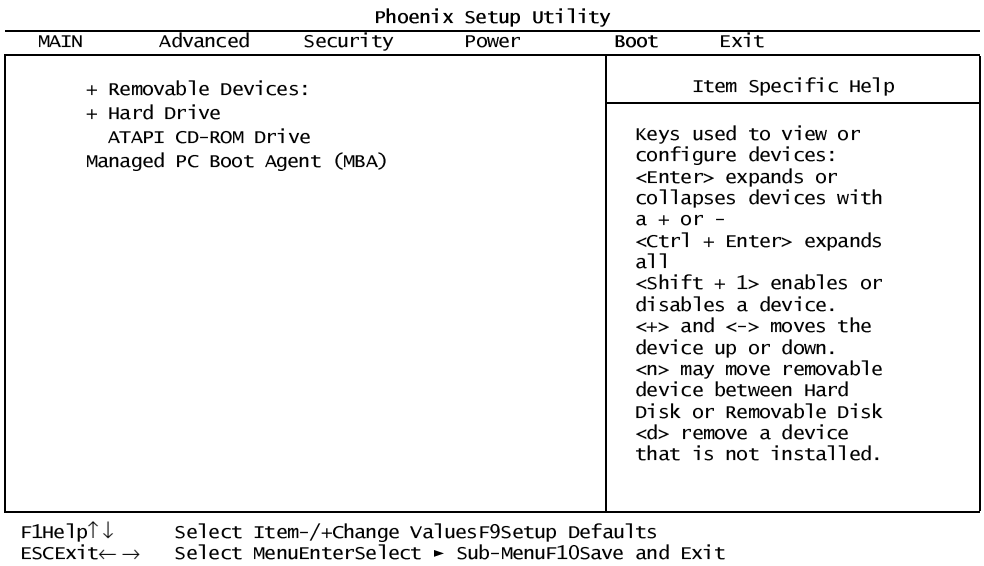
This screen, selected from the Main screen, allows the user to configure power saving options on the VMICPCI-7699.



The Throttling and CPU Throttling back hysteresis selections will only be shown if CPU DIE Temp is enabled. The default for Throttling is 50%. The default for CPU Throttling back hysteresis is 5c.

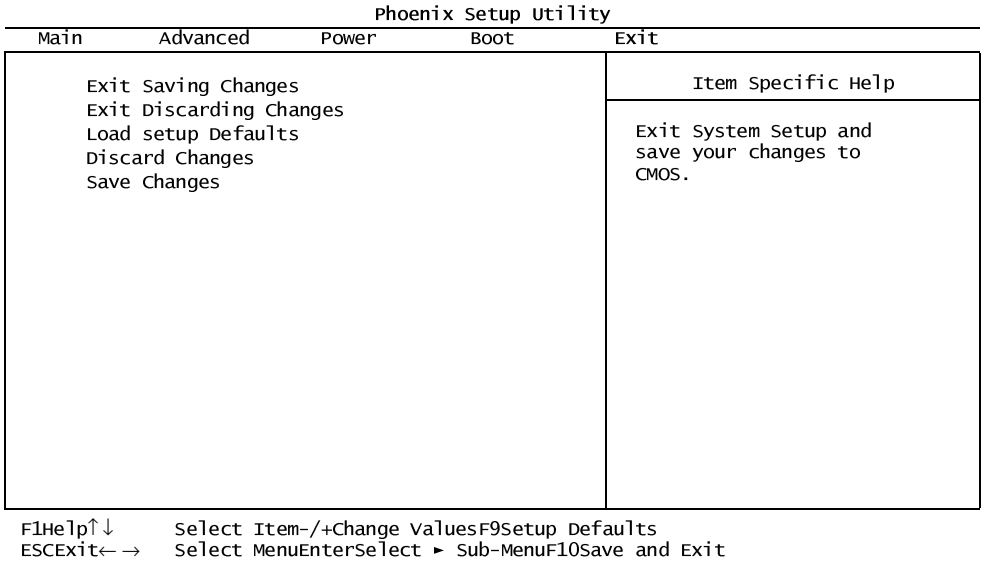
Boot Menu

The Boot priority is determined by the stack order, with the top having the highest priority and the bottom the least. The order can be modified by highlighting a device and, using the <+> or <-> keys, moving it to the desired order in the stack. A device can be boot disabled by highlighting the particular device and pressing <Shift 1>. <Enter> expands or collapses devices with a + or - next to them.



Exit Menu

The Exit menu allows the user to exit the BIOS program, while either saving or discarding any changes. This menu also allows the user to restore the BIOS defaults if desired.



Exit Saving Changes

Exit System Setup and save your changes to CMOS.

Exit Discarding Changes

Exit System Setup, discarding any changes to CMOS.

Load Setup Defaults

Load System defaults as defined at the factory.

Discard Changes

Discard any changes without exiting the Setup program.

Save Changes

Save any changes made without exiting the Setup program.

Appendix

D

LANWorks BIOS

Contents

Introduction

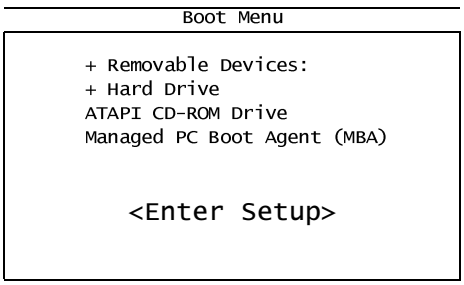
The VMICPCI-7699 includes a LANWorks option which allows the VMICPCI-7699 to be booted from a network. This appendix describes the procedures to enable this option and the LANWorks BIOS Setup screens.

Boot Menus

There are two methods of enabling the LANWorks BIOS option. The first method is the *First Boot* menu. The second is the *Boot* menu from the BIOS Setup Utility.

First Boot Menu

Press ESC at the very beginning of the boot cycle, which will access the *First Boot* menu. Selecting “Managed PC Boot Agent (MBA)” to boot from the LAN in this screen applies to the current boot only, at the next reboot the VMICPCI-7699 will revert back to the setting in the Boot menu.

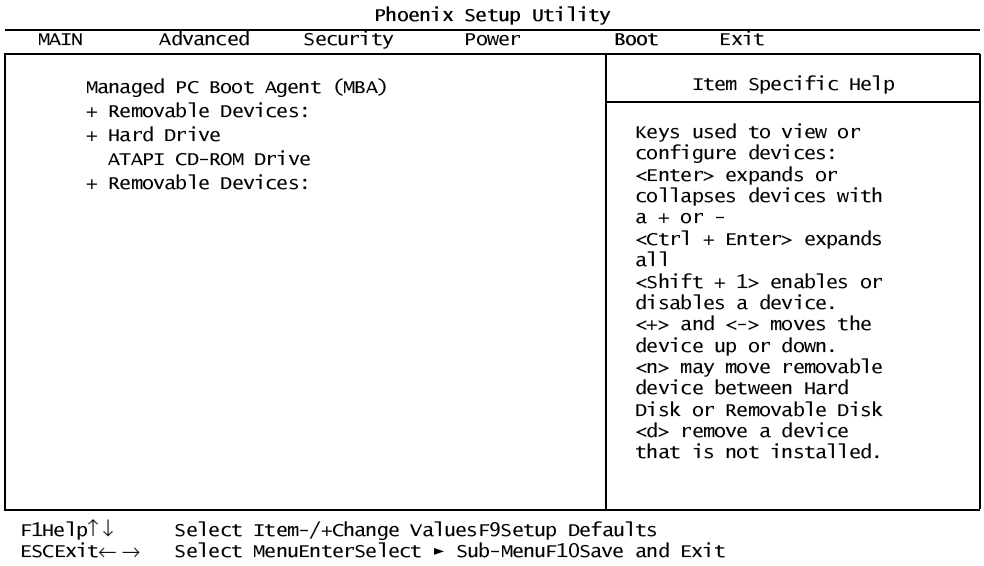


Using the arrow keys, highlight *Managed PC Boot Agent (MBA)*, and press the ENTER key to continue with the system boot.

Boot Menu

The second method of enabling the LANWorks BIOS option is to press the F2 key during system boot. This will access the BIOS Setup Utility. Advance to the Boot menu and, using the arrow keys, highlight the Managed PC Boot Agent (MBA) option. Then, using the <+> or <-> keys move the MBA option to the top of the stack.

Advance to the Exit menu and select “Exit Saving Changes” press ENTER. When the system prompts for confirmation, press “Y” for yes. The computer will then restart the system boot-up.


Phoenix Setup Utility
MAIN
Advanced
Security
Power
Boot
Exit
Managed PC Boot Agent (MBA)
+ Removable Devices:
+ Hard Drive
ATAPI CD-ROM Drive
+ Removable Devices:
Item Specific Help
Keys used to view or configure devices:
<Enter> expands or collapses devices with a + or -
<Ctrl + Enter> expands all
<Shift + 1> enables or disables a device.
<+> and <-> moves the device up or down.
<n> may move removable device between Hard Disk or Removable Disk
<d> remove a device that is not installed.
F1Help↑↓ Select Item-/+Change ValuesF9Setup Defaults
ESCExit← → Select MenuEnterSelect ► Sub-MenuF10Save and Exit

BIOS Features Setup

After the Managed PC Boot Agent has been enabled, there are several boot options available to the user. These options are RPL (default), TCP/IP, Netware, and PXE. The screens below show the defaults for each boot method.

RPL

```
Managed PC Boot Agent (MBA) v3.20 (BIOS Integrated)
(c) Copyright 1998 LANWorks Technologies Co. a subsidiary of 3Com Corporation
All rights reserved
```

```
                              Configuration

     Boot Method:             RPL

     Config Message:          Enabled
     Message Timeout:         6 Seconds
     Boot failure Prompt:     Wait for key
     Boot Failure:            Next BBS device
```

```
Use cursor keys to edit: Up/Down change field, Left/Right change value
ESC to quit, F9 restore previous settings, F10 to save
```

TCP/IP

```
Managed PC Boot Agent (MBA) v3.20 (BIOS Integrated)
(c) Copyright 1998 LANWorks Technologies Co. a subsidiary of 3Com Corporation
All rights reserved
```

```
                              Configuration

     Boot Method:             TCP/IP

     Protocol                 BootP
     Config Message:          Enabled
     Message Timeout:         6 Seconds
     Boot failure Prompt:     Wait for key
     Boot Failure:            Next BBS device
```

```
Use cursor keys to edit: Up/Down change field, Left/Right change value
ESC to quit, F9 restore previous settings, F10 to save
```

Netware

```
Managed PC Boot Agent (MBA) v3.20 (BIOS Integrated)
(c) Copyright 1998 LANWorks Technologies Co. a subsidiary of 3Com Corporation
All rights reserved
```

```
                                Configuration

     Boot Method:               Netware

     Protocol                   802.2
     Config Message:            Enabled
     Message Timeout:           6 Seconds
     Boot failure Prompt:       Wait for key
     Boot Failure:              Next BBS device
```

```
Use cursor keys to edit: Up/Down change field, Left/Right change value
ESC to quit, F9 restore previous settings, F10 to save
```

PXE

```
Managed PC Boot Agent (MBA) v3.20 (BIOS Integrated)
(c) Copyright 1998 LANWorks Technologies Co. a subsidiary of 3Com Corporation
All rights reserved
```

```
                                Configuration

     Boot Method:               PXE
```

```
Use cursor keys to edit: Up/Down change field, Left/Right change value
ESC to quit, F9 restore previous settings, F10 to save
```

Appendix

E

Device Configuration: I/O and Interrupt Control

Contents

Introduction

This appendix provides the information needed to develop custom applications for the VMICPCI-7699. The CPU board on the VMICPCI-7699 is unique in that the BIOS cannot be removed; it must be used in the initial boot cycle. A custom application, like a revised operating system for example, can only begin to operate after the BIOS has finished initializing the CPU. The VMICPCI-7699 will allow the user to either maintain the current BIOS configuration or alter this configuration to be more user specific, but this alteration can only be accomplished after the initial BIOS boot cycle has completed.

BIOS Operations

When the VMICPCI-7699 is powered on, control immediately jumps to the BIOS. The BIOS initiates a Power-on Self-Test (POST) program which instructs the microprocessor to initialize system memory as well as the rest of the system. The BIOS establishes the configuration of all on-board devices by initializing their respective I/O and Memory addresses and interrupt request lines. The BIOS then builds an interrupt vector table in main memory, which is used for interrupt handling. The default interrupt vector table and the default address map is described in Chapter 3 of this manual. Finally, the BIOS jumps to the selected boot device. This is the point at which a custom operating system could take over control of the board and proceed with a custom configuration and/or custom application. A user application could override the configuration set by the BIOS and reconfigure the system, or it could accept what the BIOS initialized.

BIOS Control Overview

There are two areas on the VMICPCI-7699 in which the user must be familiar with in order to override the initial BIOS configuration. These areas include the device addresses and the device interrupts. This appendix reviews the details of these addresses and interrupts, and provides a reference list for the individual devices used on the board.

The VMICPCI-7699 utilizes the high-performance Peripheral Component Interconnect (PCI) bus along with the Industrial Standard Architecture (ISA) bus. In general, the PCI bus is plug-and-play compatible. The components that are connected to the PCI bus are not always placed at a standard I/O or Memory address, nor are they connected to a standard interrupt request line as is the case with ISA bus devices. These PCI bus devices are re-established by the BIOS during every boot cycle, meaning these devices will not always be located at the same address or connected to the same interrupt request line every time the CPU is booted. This appendix lists the defaults that are found by powering up a specific VMICPCI-7699.

Functional Overview

The block diagram included in Figure E-1 on page 119 illustrates the VMICPCI-7699 emphasizing the I/O features, including the Embedded PCI bridge.

The circled numbers reference the appropriate data book necessary for the programming of the function block.

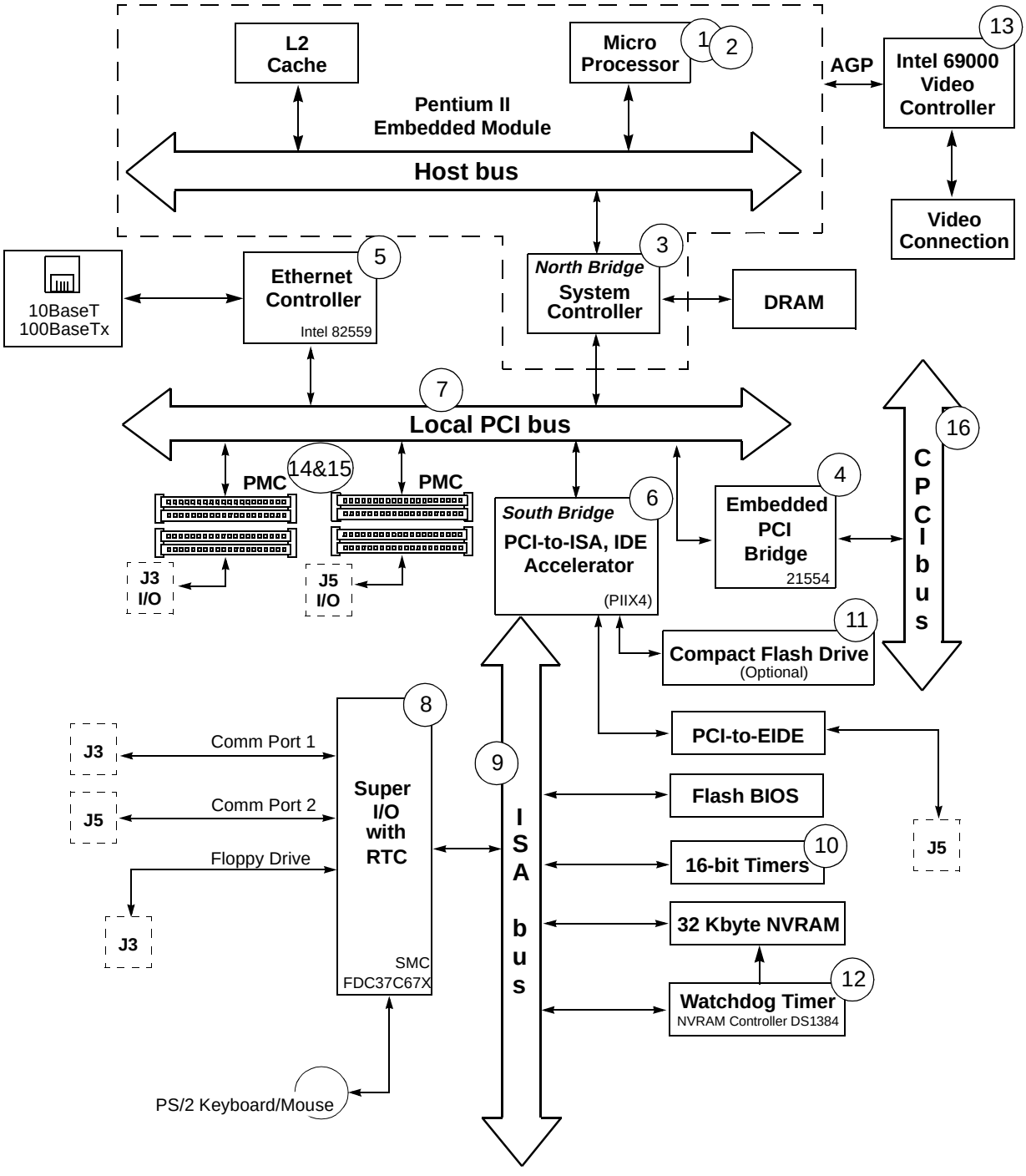


Figure E-1 VMICPCI-7699 Block Diagram

Data Book References

1. Pentium II Processor with Mobile Module

   Order Number 243668-001
   Intel Corporation
   2200 Mission College Blvd.
   P.O. Box 58119
   Santa Clara, CA 95052-8119
   (408) 765-8080
   www.intel.com

2. Pentium III Processor with Mobile Module

   Order Number 245304-002
   Intel Corporation
   2200 Mission College Blvd.
   P.O. Box 58119
   Santa Clara, CA 95052-8119
   (408) 765-8080
   www.intel.com

3. *Intel 440BX AGP set: 82443BX Host Bridge/Controller*

   April 1998, Order Number: 290633-001
   Intel Corporation
   2200 Mission College Blvd.
   P.O. Box 58119
   Santa Clara, CA 95052-8119
   (408) 765-8080
   www.intel.com

4. *Intel 21554 Embedded PCI Bridge*

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

5. *Intel 82559 10/100 Mb/s Ethernet LAN Controller*
   Intel
   www.intel.com

6. *Intel 82371EB PCI ISA IDE Xcellerator (PIIX4E)*

   2200 Mission College Blvd.
   P.O. Box 58119
   Santa Clara, CA 95052-8119

7. *PCI Local Bus Specification, Rev. 2.1*

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

8. *SMC FDC37C67X Enhanced Super I/O Controller*

   SMC Component Products Division
   300 Kennedy Dr.
   Hauppauge, NY 11788
   (516) 435-6000
   (516) 231-6004 (FAX)

9. *ISA & EISA, Theory and Operation*

   Solari, Edward
   Annabooks
   15010 Avenue of Science, Suite 101
   San Diego, CA 92128 USA
   ISBN 0-929392 -15-9

10. *82C54 CMOS Programmable Internal Timer*

    Intel Corporation
    2200 Mission College Blvd.
    P.O. Box 58119
    Santa Clara, CA 95052-8119

11. *SanDisk CompactFlash and Motorola 8 bit Microcontroller Interface Design Reference Example*
    SanDisk Corporation
    140 Caspian Court
    Sunnyvale, CA 94089-9820
    www.sandisk.com

12. *DS 1384 Watchdog Timekeeping Controller*

    Dallas Semiconductor
    4461 South Beltwood Pwky.
    Dallas, TX 75244-3292

13. *Intel 69000 Technical Reference Manual*
Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080
www.intel.com

14. *CMC Specification, P1386/Draft 2.0*

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

15. *PMC Specification, P1386.1/Draft 2.0*

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

16. *CompactPCI™ Specification PICMG 2.0 R2.1*

PCI Industrial Computer Manufactures' Group
301 Edgewater Place
Suite 220
Wakefield, MA 01880
(617) 224-1100
(617) 224-1239 (FAX)
www.picmg.org

Device Address Definition

The standard ISA architecture defines two distinctive types of address spaces for the devices and peripherals on the CPU board. These spaces have typically been named Memory address space and I/O address space. The boundaries for these areas are limited to the number of address bus lines that are physically located on the CPU board. The VMICPCI-7699 has 32 address bus lines located on the board, thereby defining the limit of the address space as 4 Gbyte. The standard ISA architecture defines Memory address space from zero to 4 Gbyte and the separate I/O address space from zero to 64 Kbyte.

ISA Devices

The ISA devices on the VMICPCI-7699 are configured by the BIOS at boot-up. They are mapped in I/O address space within standard addresses and their interrupts are mapped to standard interrupt control registers. However, all of the ISA devices with the exception of the real-time clock, keyboard, and programmable timer are relocatable to almost anywhere within the standard 1 Kbyte of I/O address space. Table E-1 below defines the spectrum of addresses available for reconfiguration of ISA devices.

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

Table E-1 ISA Device Mapping Configuration

Device	Memory Space	I/O Address Space	PIC Interrupt Options	Byte Address Boundary	Default
Floppy	N/A	[0x100 - 0xFF8]	IRQ1 - IRQ15	8	$3F0
Serial Port 1	N/A	[0x100 - 0xFF8]	IRQ1 - IRQ15	8	$3F8
Serial Port 2	N/A	[0x100 - 0xFF8]	IRQ1 - IRQ15	8	$2F8
Real-Time Clock	Nonrelocatable				
Keyboard	Nonrelocatable				
Auxiliary I/O	N/A	- Primary I/O [0x000 - 0xFFF] - Secondary I/O [0x000 - 0xFFF]	IRQ1, IRQ3-IRQ15	1 1	
Programmable Timer		Nonrelocatable 0x500	IRQ5	4	0x500
Watchdog Timer	Nonrelocatable 0xD8000	N/A	N/A	0xD	0xD8000
SRAM	Nonrelocatable 0xD8014	N/A	N/A	(32K-0xD)	0xD800E

PCI Devices

PCI devices are fully configured under I/O and/or Memory address space. Table E-1 describes the on-board PCI bus devices along with each device's configuration spectrum.

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

Table E-2 PCI Device Mapping Configuration

Device	Memory Space	I/O Address Space	PIC Interrupt Options
Embedded PCI Bridge	Anywhere	Anywhere	PCI Defined
Ethernet	Anywhere	Anywhere	PCI Defined
Timer Interrupt Registers	N/A	Fixed	N/A
PMC	Anywhere	Anywhere	PCI Defined

Device Interrupt Definition

PC/AT Interrupt Definition

The interrupt hardware implementation on the VMICPCI-7699 is built around the ISA standard which evolved from the IBM PC / XT architecture. In the IBM PC / XT systems, only eight Interrupt Request (IRQ) lines exist, numbered from IRQ0 to IRQ7. These interrupt lines were included originally on a 8259A Priority Interrupt Controller (PIC) chip.

The IBM PC / AT computer added eight more IRQx lines, numbered IRQ8 to IRQ15, by cascading a second slave 8259A PIC into the original master 8259A PIC. The interrupt line IRQ2 at the master PIC was committed as the cascade input from the slave PIC. This master / slave architecture, the standard PC / AT interrupt mapping, is illustrated in Figure E-2 on page 126 within the PCI-to-ISA Bridge PIIX4 82371AB section of the diagram.

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

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

ISA Device Interrupt Map

The VMICPCI-7699 BIOS maps the IRQx lines to the appropriate device per the standard ISA architecture. Reference Figure E-2 on page 126. This initialization operation cannot be changed; however, a custom application could reroute the interrupt configuration after the BIOS has completed the initial configuration cycle.

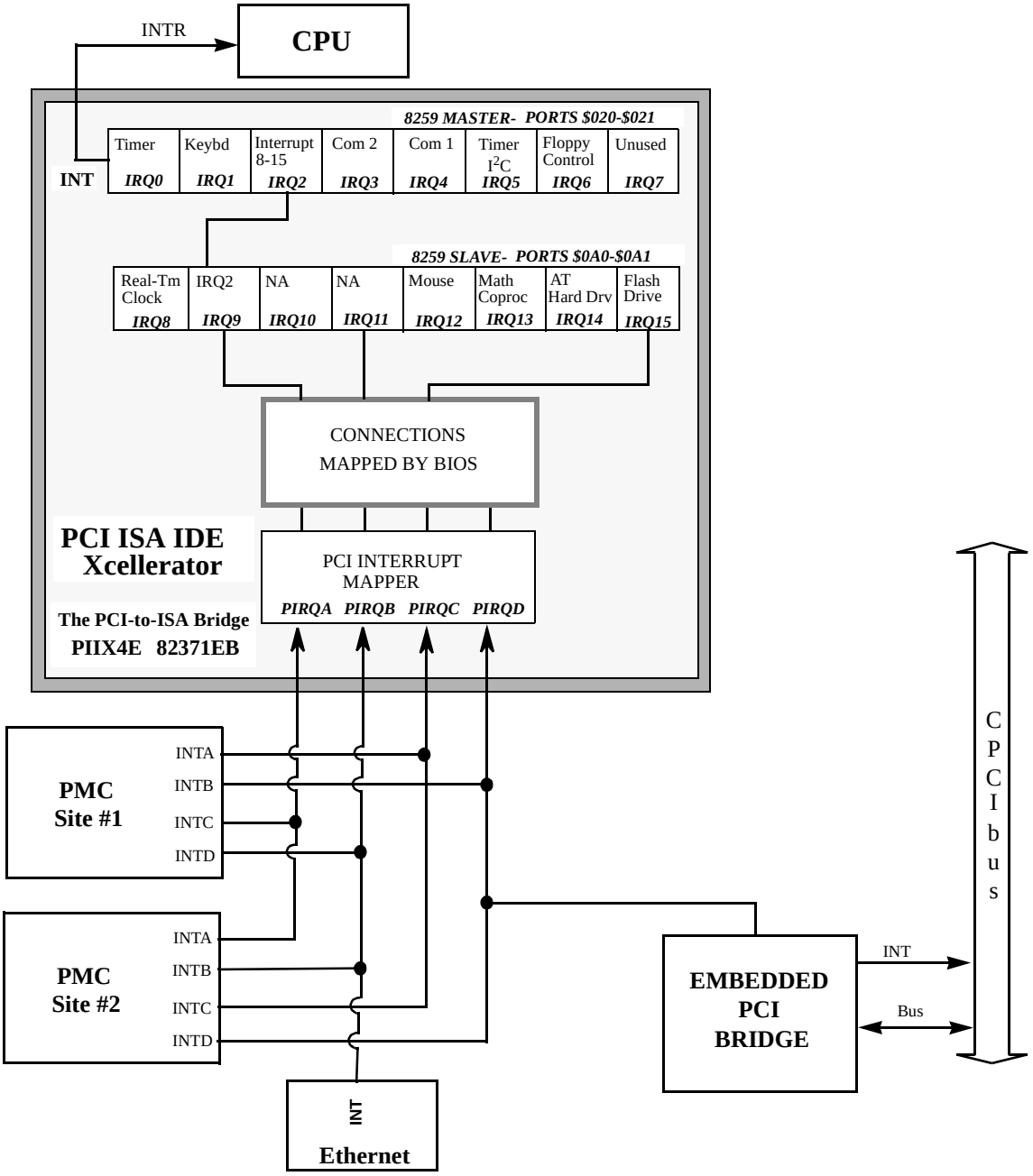


Figure E-2 BIOS Default Connections for the PC Interrupt Logic Controller

NOTE: See Table E-3 for the PCI Device Interrupt Mapping by the BIOS.

PCI Device Interrupt Map

The PCI bus-based external devices include the PMC site, Ethernet Controller, and the PCI-to-PCI bridge. The default BIOS maps these external devices to the PCI Interrupt Request (PIRQx) lines of the PIIX4. This mapping is illustrated in Figure E-2 on page 126 and is defined in Table E-3.

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

Table E-3 PCI Device Interrupt Mapping by the BIOS

DEVICE	COMPONENT	VENDOR ID	DEVICE ID	CPU ADDRESS MAP ID SELECT	DEVICE PCI INTERRUPT	MOTHER-BOARD PCI INTERRUPT MAPPER	DATA BOOK REF. #	Arbitration Level
Power Management	PIIX4 82371EB Function	0x8086	0x7113	AD18	N / A	N / A	5	N / A
PCI-to-ISA Bridge	PIIX4 82371EB Function 00	0x8086	0x7000	AD18	N / A	N / A	5	N / A
Ethernet Controller	Intel 82559	0x1011	0x0019	AD22	INTA	PIRQ1	4	REQ1
PCI IDE Controller	PIIX4 82371EB Function 01	0x8086	0x7111	AD18	N / A	N / A	5	N / A
PCI Host Bridge	Intel 440BX	0x8086	0x7190	N / A	N / A	N / A	2	N / A
PMC	N / A	N / A	N / A	AD31	INTA	PIRQ2	14	REQ3
Compact PCI Bridge	DEC 21554	0x1011	0x0046	AD26	INTA	PIRQ3	3	REQ0

NOTE: PIRQ4 interrupt is not enabled by the BIOS.

The motherboard accepts these PCI device interrupts through the PCI Interrupt Mapper function. The BIOS default maps the PCI Interrupt Request (PIRQx) external device lines to one of the available slave PIC Interrupt Request lines, IRQ (9, 10, 11, or 15). The BIOS default mapping of the PIRQx to the slave PIC is defined in Table E-4.

Table E-4 Default PIRQx to IRQx BIOS Mapping with all Devices Loaded

PCI INTx	PIC IRQx
PIRQ0	IRQ11
PIRQ1	IRQ10
PIRQ2	IRQ9
PIRQ3	IRQ9

Using the interrupt steering registers of the 82371EB PIIX4E, the user can override the BIOS defaults and map any of the PCI interrupts (PIRQ0-3) to any of the following PIC IRQx (ISA) interrupts: IRQ15, 14, 12-9, or 7-3.

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

Appendix

F

Sample C Software

Contents

Introduction

This appendix provides listings of a library of sample code that the programmer may utilize to build applications. These files are provided to the VMICPCI-7699 user on disk 320-500045-007, Sample Application C Code for the VMICPCI-7699, included in the distribution disk set.

Because of the wide variety of environments in which the VMICPCI-7699 operates, the samples provided in this appendix are not necessarily intended to be verbatim boilerplates. Rather, they are intended to give the end user an example of the standard structure of the operating code.

Directory SRAM

**File: T_SRAM.C

```
/*************************************************************************/
/* FILE: T_SRAM.C                                                        */
/*                                                                       */
/* Test battery backed SRAM with patterns and data=address.              */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
#include    <stdlib.h>
#include    <stdio.h>
#include    <dos.h>
unsigned char far * b_ptr;
unsigned int  far * w_ptr;
unsigned long far * l_ptr;
unsigned int  far * buf_ptr;
static unsigned long pat[4] = {
  0x55555555,
  0xCCCCCCCC,
  0x66666666,
  0xFFFFFFFF
};
void main( void ) {
unsigned long i, x;
unsigned char bdat;
unsigned char brd;
unsigned int  wdat;
unsigned int  wrd;
unsigned long ldat;
unsigned long lrd;
  printf("\nTesting 32K SRAM DATA/~DATA B/W/L/ADDR .............");
  buf_ptr = (unsigned int far *) MK_FP( 0xD800, 0x18 );
  /* fill and test buf with DATA/~DATA BYTES 4 patterns */
  for( x = 0; x < 4; x++ ) {
    b_ptr = (unsigned char far *) buf_ptr;
    bdat = (unsigned char) pat[x];
    for( i = 0x18; i < 0x8000; i++ ) {
      *b_ptr++ = bdat;
      bdat = ~bdat;
    }
    b_ptr = (unsigned char far *) buf_ptr;
```

```
  bdat = (unsigned char) pat[x];
  for( i = 0x18; i < 0x8000; i++ ) {
    brd = *b_ptr++;
    if( bdat != brd ) {
      printf("FAILED\nBYTE DATA @ ADDR: %Fp  WR: %.2X  RD: %.2X\n",
              --b_ptr, bdat, brd );
      exit( 1 );
    }
    bdat = (~bdat) & 0xFF;
  }
}
/* fill and test buf with DATA/~DATA WORDS 4 patterns */
for( x = 0; x < 4; x++ ) {
  w_ptr = (unsigned int far *) buf_ptr;
  wdat = (unsigned int) pat[x];
  for( i = 0x18; i < 0x8000; i+=2 ) {
    *w_ptr++ = wdat;
    wdat = ~wdat;
  }
  w_ptr = (unsigned int far *) buf_ptr;
  wdat = (unsigned int) pat[x];
  for( i = 0x18; i < 0x8000; i+=2 ) {
    wrd = *w_ptr++;
    if( wdat != wrd ) {
      printf("FAILED\nWORD DATA @ ADDR: %Fp  WR: %.4X  RD: %.4X\n",
              --w_ptr, wdat, wrd );
      exit( 1 );
    }
    wdat = ~wdat;
  }
}
/* fill and test buf with DATA/~DATA LONGS 4 patterns */
for( x = 0; x < 4; x++ ) {
  l_ptr = (unsigned long far *) MK_FP( 0xD800, 0x18 ); ;
  ldat = (unsigned long) pat[x];
  for( i = 0x18; i < 0x8000; i+=4 ) {
    *l_ptr++ = ldat;
    ldat = ~ldat;
  }
  l_ptr = (unsigned long far *) MK_FP( 0xD800, 0x18 );
  ldat = (unsigned long) pat[x];
  for( i = 0x18; i < 0x8000; i+=4 ) {
    lrd = *l_ptr++;
    if( ldat != lrd ) {
```

```
        printf("FAILED\nLONG DATA @ ADDR: %Fp  WR: %.8X  RD: %.8X\n",
                --l_ptr, ldat, lrd );
        exit( 1 );
      }
      ldat = ~ldat;
    }
  }
  /* fill and test buf with DATA = ADD LONG */
  for( x = 0; x < 4; x++ ) {
    l_ptr = (unsigned long far *) MK_FP( 0xD800, 0x18 );
    for( i = 0x18; i < 0x8000; i+=4 ) {
      *l_ptr++ = i;
    }
    l_ptr = (unsigned long far *) MK_FP( 0xD800, 0x18 );
    for( i = 0x18; i < 0x8000; i+=4 ) {
      lrd = *l_ptr++;
      if( lrd != i ) {
        printf("FAILED\nDATA=ADDR @ ADDR: %Fp  WR: %.8X  RD: %.8X\n",
                --l_ptr, i, lrd );
        exit( 1 );
      }
    }
  }
  printf("PASSED\n\n");
  exit( 0 );
} /* end main */
```

Directory Timers

This directory contains sample code useful in the creation of applications involving the VMICPCI-7699's three software controlled 16-bit timers. The code is written for the control of a single timer, but can be utilized in generating code for any timer configuration. The timers are described in Chapter 4 of the manual.

**File: CPU.H

```
/*
**  FILE: T7699T.H
**
*/
#define     DID_PWR_MGM     0x7113
#define     VID_PWR_MGM     0x8086
#define     IRQ5            0x0D
#define     GPI_T1          0x80    /* PIX General Purpose Input 15 (tmr 1)     */
#define     GPI_T2          0x40    /* PIX General Purpose Input 14 (tmr 2)     */
#define     GPI_T3          0x20    /* PIX General Purpose Input 13 (tmr 3)     */
#define     GPO_T1          0x40    /* PIX General Purpose Output 30 (tmr 1)    */
#define     GPO_T2          0x10    /* PIX General Purpose Output 28 (tmr 2)    */
#define     GPO_T3          0x08    /* PIX General Purpose Output 27 (tmr 3)    */
#define     GPO_CLR         0xA7    /* PIX General Purpose Output CLR TMRS      */
#define     TIMER_CNTR1     0x00    /* Timer counter 1 offset                   */
#define     TIMER_CNTR2     0x01    /* Timer counter 2 offset                   */
#define     TIMER_CNTR3     0x02    /* Timer counter 3 offset                   */
#define     TIMER_CNTL      0x03    /* Timer control offset                     */
/*************************************************************************/
/* 8254 Control word                                                     */
/*************************************************************************/
#define     CW_SC0          0x00    /* W Selcect counter 0                  */
#define     CW_SC1          0x40    /* W Selcect counter 1                  */
#define     CW_SC2          0x80    /* W Selcect counter 2                      */
#define     CW_RBC          0xC0    /* W Read back command                      */
#define     CW_CLC          0x00    /* W Cntr latch command (cnt/stat)          */
#define     CW_SLC          0x00    /* W Status latch command           */
#define     CW_LSB          0x10    /* W LSB only                               */
#define     CW_MSB          0x20    /* W MSB only                               */
#define     CW_LSBMSB       0x30    /* W LSB first then MSB             */
#define     CW_M0           0x00    /* W Mode 0                                 */
#define     CW_M1           0x02    /* W Mode 1                                 */
#define     CW_M2           0x04    /* W Mode 2                                 */
#define     CW_M3           0x06    /* W Mode 3                                 */
#define     CW_M4           0x08    /* W Mode 4                                 */
#define     CW_M5           0x0A    /* W Mode 5                                 */
#define     CW_BCD          0x01    /* W Binary Coded Decimal           */
```

```
#define     CW_RB_CNT       0x00    /* W Read back count                    */
#define     CW_RB_STAT      0x00    /* W Read back status              */
#define     CW_RB_C0        0x02    /* W Read back counter 0           */
#define     CW_RB_C1        0x04    /* W Read back counter 1           */
#define     CW_RB_C2        0x08    /* W Read back counter 2           */
```

**File: PCI.H

```
define TRUE  1
#define FALSE 0

#define CARRY_FLAG              0x01

/* PCI Functions */
#define PCI_FUNCTION_ID         0xB1
#define PCI_BIOS_PRESENT        0x01
#define FIND_PCI_DEVICE         0x02
#define FIND_PCI_CLASS_CODE     0x03
#define READ_CONFIG_BYTE        0x08
#define READ_CONFIG_WORD        0x09
#define READ_CONFIG_DWORD       0x0A
#define WRITE_CONFIG_BYTE       0x0B
#define WRITE_CONFIG_WORD       0x0C
#define WRITE_CONFIG_DWORD      0x0D

/* PCI Return codes */

#define SUCCESSFUL              0x00
#define NOT_SUCCESSFUL          0x01

/* PCI Config Space Regs */
#define PCI_CS_VENDOR_ID
#define PCI_CS_DEVICE_ID
#define PCI_CS_COMMAND
#define PCI_CS_STATUS
#define PCI_CS_REVISION_ID
#define PCI_CS_CLASS_CODE
#define PCI_CS_BASE_ADDRESS_0
#define PCI_CS_BASE_ADDRESS_1
#define PCI_CS_BASE_ADDRESS_2
#define PCI_CS_BASE_ADDRESS_3
#define PCI_CS_BASE_ADDRESS_4
#define PCI_CS_BASE_ADDRESS_5
#define PCI_CS_INTERRUPT_LINE
#define PCI_CS_INTERRUPT_PIN

/* Prototypes */

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

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

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

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

void outpd(unsigned short port, unsigned long value);

unsigned long inpd(unsigned short port);
```

**File: PCI.C

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

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

int find_pci_device(unsigned short device_id,
            unsigned short vendor_id,
            unsigned short index,
            unsigned char *bus_number,
            unsigned char *device_and_function)
{
   int ret_status;
   unsigned short ax, bx, flags;
   _CX = device_id;
   _DX = vendor_id;
   _SI = index;
   _AH = PCI_FUNCTION_ID;
   _AL = FIND_PCI_DEVICE;

   geninterrupt(0x1a);

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

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

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

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

   geninterrupt(0x1a);

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

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

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

   geninterrupt(0x1a);

   ax = _AX;
   flags = _FLAGS;
   if ((flags & CARRY_FLAG) == 0)
   {
     ret_status = HIGH_BYTE(ax);
   }
   else
   {
     ret_status = NOT_SUCCESSFUL;
   }
   return(ret_status);
}
void outpd(unsigned short port, unsigned long value)
{
   _DX = port;
   _EAX = value;
   __emit__(0x66, 0xEF);
}
unsigned long inpd(unsigned short port)
{
   _DX = port;
   __emit__(0x66, 0xED);
   return(_EAX);
}
/*
```

**File: T_Timers.C

```
/*****************************************************************************/
/*  FILE:   T_TIMERS.C                                                       */
/*          Test Timers (TIC is jumper selectable for 500 ns or 1 us)        */
/*                                                                           */
/*****************************************************************************/
#include  <stdlib.h>
#include  <stdio.h>
#include  <string.h>
#include  <conio.h>
#include  <ctype.h>
#include  <dos.h>
#include  "pci.h"
#include  "cpu.h"
/* TT.C function prototypes */
void do_exit( int );
/* TIMERS.C function prototypes */
void far interrupt irq_rcvd( void );
void init_timer_int( void );
void restore_orig_int( void );
void load_counter( int, unsigned int );
void read_counter( int, unsigned int *, unsigned char * );
/* global variables */
unsigned char bus, dev_func;
/* the following globals are used in other files as 'extern' variables */
unsigned char tmr_status, t1_stat, t2_stat, t3_stat;
unsigned int tmr_cnt, t1_cnt, t2_cnt, t3_cnt;
unsigned long t1_count;      /* counts no. of times timer 1 ISR entered */
unsigned long t2_count;      /* counts no. of times timer 2 ISR entered */
unsigned long t3_count;      /* counts no. of times timer 3 ISR entered */
unsigned int pwr_mgm_base;
unsigned int gpi_base;
unsigned int gpo_base;
unsigned int timer_base;
unsigned char pic1_org;
unsigned char gpo_org;
void main( int argc, char * argv[] )
{
  unsigned long t1, t2, t3;
  int test_int;
  unsigned long temp_dword;
  char user[80];
  timer_base = 0x500;
```

```
 /* try to locate the power management device on the PCI bus */
 test_int = find_pci_device(DID_PWR_MGM, VID_PWR_MGM, 0,
                &bus, &dev_func);
 if(test_int != SUCCESSFUL)
 {
    printf("\nUnable to locate power management device on PCI bus\n");
    do_exit( 1 );
 }
 /* get base address from config area */
 test_int = read_configuration_area(READ_CONFIG_DWORD,
                                    bus, dev_func, 0x40, &temp_dword);
 if(test_int != SUCCESSFUL)
 {
  printf("\nUnable to read POWER MGM. BASE ADDRESS @ 0x40 in config space\n");
    do_exit( 1 );
 }
 pwr_mgm_base = temp_dword & 0x0000FFC0;
 gpi_base = pwr_mgm_base + 0x31; /* PIX general purpose input bits 8-15 */
 gpo_base = pwr_mgm_base + 0x37; /* PIX general purpose output bits 24-31*/
 disable();
/* disable timers by reloading the control word */
 outp( timer_base + TIMER_CNTL, (CW_SC0 | CW_LSBMSB | CW_M2) );
 outp( timer_base + TIMER_CNTL, (CW_SC1 | CW_LSBMSB | CW_M2) );
 outp( timer_base + TIMER_CNTL, (CW_SC2 | CW_LSBMSB | CW_M2) );
 /* Read 8259 slave Programmable Interrupt controller */
 pic1_org = inp(0x21) & 0xFF; /* slave mask bits */
 /* disable interrupt 5 */
 outp(0x21, (pic1_org | 0x20) ); /* 0 = enable 1 = disable */
 enable();
 gpo_org = inp( gpo_base ) & 0xFF;
 /* setup timers interrupt service routine */
 init_timer_int();
 /*
 ** verify all three counters can generate an interrupt (counters 1,2,3)
 */
 printf("\nTesting all three 16 bit counters for interrupt ....");
 /* setup for interrupts to occur */
 t1_count = 0;
 t2_count = 0;
 t3_count = 0;
 t1 = 0;
 t2 = 0;
 t3 = 0;
 tmr_status = 0;
```

```
test_int = 100;
/* load counters */
load_counter( 1, 0xFFFF );
do
{
  if( t1_count ) {
    t1++;
    break;
  }
  test_int--;
  delay( 1 );
} while( test_int );
outp( timer_base + TIMER_CNTL, (CW_SC0 | CW_LSBMSB | CW_M2) );
outp( timer_base + TIMER_CNTL, (CW_SC1 | CW_LSBMSB | CW_M2) );
outp( timer_base + TIMER_CNTL, (CW_SC2 | CW_LSBMSB | CW_M2) );
 tmr_status = 0;
test_int = 100;
load_counter( 2, 0xFFFF );
do
{
  if( t2_count ) {
    t2++;
    break;
  }
  test_int--;
  delay( 1 );
} while( test_int );
/* disable timers by reloading the control word */
outp( timer_base + TIMER_CNTL, (CW_SC1 | CW_LSBMSB | CW_M2) );
tmr_status = 0;
test_int = 100;
load_counter( 3, 0xFFFF );
do
{
  if( t3_count ) {
    t3++;
    break;
  }
  test_int--;
  delay( 1 );
} while( test_int );
/* disable timers by reloading the control word */
outp( timer_base + TIMER_CNTL, (CW_SC2 | CW_LSBMSB | CW_M2) );
/* clear all three status bits in GPI */
```

```
  outp( gpo_base, ( gpo_org & GPO_CLR ) );
  /* set all three GPO outputs to 1 to allow int status registers to function */
  outp( gpo_base, ( gpo_org | GPO_T1 | GPO_T2 | GPO_T3 ) );
  if( t1 && t2 && t3 )
  {
    printf("PASSED\n");
  }
  else
  {
    printf("FAILED\n");
    if( !t1 ) printf("TIMER 1 failed\n");
    if( !t2 ) printf("TIMER 2 failed\n");
    if( !t3 ) printf("TIMER 3 failed\n");
    do_exit( 2 );
  }
  /* do orderly exit */
  do_exit( 3 );
} /* end main */
void do_exit( int xit_code )
{
  if( xit_code > 1 )  restore_orig_int();
  outp( gpo_base, gpo_org );
  if( xit_code == 3 ) xit_code = 0;
  exit( xit_code );
} /* do_exit */
```

**File: Timer.C

```
/*
**  FILE: TIMERS.C
**
*/
#include  <stdlib.h>
#include  <stdio.h>
#include  <dos.h>
#include  <ctype.h>
#include  <conio.h>
#include  "cpu.h"
/* function prototypes */
void far interrupt irq_rcvd( void );
void init_timer_int( void );
void restore_orig_int( void );
void load_counter( int, unsigned int );
void read_counter( int, unsigned int *, unsigned char * );
/* global variables */
extern unsigned long t1_count;    /* timer 1 count */
extern unsigned long t2_count;    /* timer 2 count */
extern unsigned long t3_count;    /* timer 3 count */
extern unsigned char tmr_status;
extern unsigned int gpi_base;
extern unsigned int gpo_base;
extern unsigned int timer_base;
extern unsigned char pic1_org;
extern unsigned char gpo_org;
void far interrupt (* old_vect)(void);
/*****************************************************************/
/*  init_timer_int()                                             */
/*                                                               */
/*  purpose: Using the interrupt assigned, the original vector is  */
/*           saved and the vector to the new ISR is installed. The */
/*           programmable-interrupt-controller (PIC) is enabled.   */
/*                                                               */
/*  Prerequisite: The interrupt line to be used must have          */
/*                already been loaded in the global variable.      */
/*                                                               */
/*****************************************************************/
/*  parameters: none                                             */
/*****************************************************************/
/*  return value: none                                           */
/*****************************************************************/
```

```
void init_timer_int( void )
{
  disable();
  old_vect = getvect( IRQ5 ); /* save vector for IRQ5 */
  setvect( IRQ5, irq_rcvd );
  /* enable interrupt 5 */
  outp(0x21, (pic1_org & 0xDF) ); /* 0 = enable 1 = disable */
  /* clear all three GPO inputs */
  outp( gpo_base, ( gpo_org & GPO_CLR ) );
  /* set all three GPO outputs to 1 to allow int status registers to function */
  outp( gpo_base, (gpo_org | GPO_T1 | GPO_T2 | GPO_T3 ) );
  enable();
} /* init_timer_int */
/*********************************************************************/
/*  restore_orig_int()                                               */
/*                                                                   */
/*  purpose: Using the interrupt assigned, the original vector is  */
/*           restored and the programmable-interrupt-controller    */
/*           is disabled.                                            */
/*                                                                   */
/*  Prerequisite: The interrupt line to be used must have          */
/*                already been loaded in the global variable.      */
/*                                                                   */
/*********************************************************************/
/*  parameters: none                                                 */
/*********************************************************************/
/*  return value: none                                               */
/*********************************************************************/
void restore_orig_int( void )
{
  disable();
  outp(0x21, pic1_org);
  setvect( IRQ5, old_vect );
  enable();
} /* restore_orig_int */
/*********************************************************************/
/*  load_counter()                                                   */
/*                                                                   */
/*  purpose: Loads the appropriate counter with the count passed   */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
/*  parameters: int counter = 1, 2, 3 for COUNTER 1, 2, or 3       */
/*              unsigned int count = count to be loaded            */
/*********************************************************************/
/*  return value: none                                               */
/*********************************************************************/
```

```
void load_counter( int counter, unsigned int count )
{
  int lsb, msb;
  lsb = count & 0xff;
  msb = count >> 8;
  switch( counter )
  {
    case 1: /* select counter 1, LSB then MSB, mode 2 */
      outp( timer_base + TIMER_CNTL, (CW_SC0 | CW_LSBMSB | CW_M2) );
      outp( timer_base + TIMER_CNTR1, (unsigned char) lsb );
      outp( timer_base + TIMER_CNTR1, (unsigned char) msb );
    break;
    case 2: /* select counter 2, LSB then MSB, mode 2 */
      outp( timer_base + TIMER_CNTL, (CW_SC1 | CW_LSBMSB | CW_M2) );
      outp( timer_base + TIMER_CNTR2, (unsigned char) lsb );
      outp( timer_base + TIMER_CNTR2, (unsigned char) msb );
    break;
    case 3: /* select counter 3, LSB then MSB, mode 2 */
      outp( timer_base + TIMER_CNTL, (CW_SC2 | CW_LSBMSB | CW_M2) );
      outp( timer_base + TIMER_CNTR3, (unsigned char) lsb );
      outp( timer_base + TIMER_CNTR3, (unsigned char) msb );
    break;
  }
} /* load_counter */
/*****************************************************************/
/*  read_counter()                                               */
/*                                                               */
/*  purpose: Reads the appropriate counter in the appropriate    */
/*           bank with the remaining count and status.            */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*  parameters: int counter = 1, 2, 3 for COUNTER 1, 2, or 3     */
/*              unsigned int * count = remaining count           */
/*              unsigned char * status = counter status          */
/*****************************************************************/
/*  return value: none                                           */
/*****************************************************************/
void read_counter( int counter,
                   unsigned int * count, unsigned char * status )
{
  int lsb, msb;
  switch( counter )
  {
    case 1: /* select counter 1, LSB then MSB */
      outp( timer_base + TIMER_CNTL, ( CW_RBC | CW_RB_CNT | CW_RB_STAT | CW_RB_C0 )
);
```

```
      *status = inp( timer_base + TIMER_CNTR1 ) & 0xFF;
      lsb = inp( timer_base + TIMER_CNTR1 ) & 0xFF;
      msb = inp( timer_base + TIMER_CNTR1 ) & 0xFF;
      msb = msb << 8;
      *count = ( lsb | msb );
    break;
    case 2: /* select counter 2, LSB then MSB */
      outp( timer_base + TIMER_CNTL, ( CW_RBC | CW_RB_CNT | CW_RB_STAT | CW_RB_C1 )
);
      *status = inp( timer_base + TIMER_CNTR2 ) & 0xFF;
      lsb = inp( timer_base + TIMER_CNTR2 ) & 0xFF;
      msb = inp( timer_base + TIMER_CNTR2 ) & 0xFF;
      msb = msb << 8;
      *count = ( lsb | msb );
    break;
    case 3: /* select counter 3, LSB then MSB */
      outp( timer_base + TIMER_CNTL, ( CW_RBC | CW_RB_CNT | CW_RB_STAT | CW_RB_C2 )
);
      *status = inp( timer_base + TIMER_CNTR3 ) & 0xFF;
      lsb = inp( timer_base + TIMER_CNTR3 ) & 0xFF;
      msb = inp( timer_base + TIMER_CNTR3 ) & 0xFF;
      msb = msb << 8;
      *count = ( lsb | msb );
    break;
  }
} /* read_counter */
/*********************************************************************/
/*  irq_rcvd()                                                       */
/*                                                                   */
/*  purpose: Interrupt service routine used to service any of the  */
/*           counters on the 7699.                                   */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
/*  parameters: none                                                 */
/*********************************************************************/
/*  return value: none                                               */
/*********************************************************************/
void interrupt irq_rcvd(void)
{
  disable();
  asm {
    .386P
    push  eax
```

```
    push  ebx
  }
  tmr_status = inp( gpi_base ) & 0xFF;
  /* increment counts and clear status */
  if( tmr_status & GPI_T1 ) {
    t1_count++;
    outp( gpo_base, (gpo_org & (~GPO_T1)) );  /* clear timer 1 status bit */
  }
  if( tmr_status & GPI_T2 ) {
    t2_count++;
    outp( gpo_base, (gpo_org & (~GPO_T2)) );  /* clear timer 2 status bit */
  }
  if( tmr_status & GPI_T3 ) {
    t3_count++;
    outp( gpo_base, (gpo_org & (~GPO_T3)) );  /* clear timer 3 status bit */
  }
  outp( gpo_base, (gpo_org | GPO_T1 | GPO_T2 | GPO_T3) ); /* enable status */
  /* Non specific end of interrupt to PIC */
  outp(0x20, 0x20);  /* Master end of irq command */
  asm {
    .386P
    pop  ebx
    pop  eax
  }
  enable();
}
```

Directory WATCHDOG

This directory contains sample code useful in the creation of applications involving the VMICPCI-7699's watchdog timer function as described in Chapter 4.

**File: Watchdog.H

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

/*
** DS1384 COMMAND REGSITER BIT DEFINITIONS
*/

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

**File: WDT0_RST.C

```
/*************************************************************************/
/* FILE: WDTO_RST.C                                                      */
/*                                                                       */
/* Setup watchdog to issue reset on time out.                            */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
#include    <stdlib.h>
#include    <stdio.h>
#include    <dos.h>
#include    <time.h>
#include    <conio.h>
#include    <ctype.h>
#include    "watchdog.h"
unsigned char far * wd_ptr;
time_t t;
char usr[80];
char reg_b;
void main( void ) {
  /* Install Jumper E6 */
  wd_ptr = (unsigned char far *) MK_FP( 0xD800, 0 );
  /* set WatchDog Alarm Mask 1 - deactivated and update with 0 time */
  *(wd_ptr + WD_CMD) = ( WD_TE | WD_WAM );
  *(wd_ptr + WD_MSEC) = 0; /* load with 0 to disable */
  *(wd_ptr + WD_SEC) = 0;  /* load with 0 to disable */
  *(wd_ptr + WD_CMD) = ( WD_TE | WD_WAM ); /* allow update with 0 time */
  *(wd_ptr + WD_CMD) = WD_WAM; /* set watchdog alarm mask to 1 */
  *(wd_ptr + WD_MSEC) = 0x99; /* 00.99 seconds */
  *(wd_ptr + WD_SEC) = 0x99;  /* 99.00 seconds */
  *(wd_ptr + WD_CMD) = ( WD_TE | WD_PU ); /* set for 3 ms pulse */
  printf("Reset time out in 99.99 seconds\n\n");
  time(&t);
  printf("START DATE & TIME: %.24s\n\n", ctime(&t) );
  do {
    time(&t);
    printf("CURRENT DATE & TIME: %.24s\r", ctime(&t) );
    delay( 250 );
  } while( !kbhit() );
} /* end main */
```

Index