Model 617 Adapter
Hardware Manual

Connects a PCI Computer
to a VMEbus System

Disclaimer

Please read and abide by the following paragraphs. Questions and comments should be directed to:

Technical Publications Department
SBS Bit 3 Operations
1284 Corporate Center Drive
St. Paul, MN 55121-1245
612-905-4700

SBS Bit 3 Operations does not authorize the use of its components in life support applications where failure or malfunction of the component may result in injury or death. In accordance with SBS Bit 3's terms and conditions of sale, the user of SBS Bit 3 components in any and all life support applications assumes all risks arising out of such use and further agrees to indemnify and hold SBS Bit 3 harmless against any and all claims of whatsoever kind or nature (including claims of culpable conduct [strict liability, negligence or breach of warranty] on the part of SBS Bit 3) for all costs of defending any such claims.

SBS Bit 3 does not authorize the use of its components in control and process applications where failure or malfunction of the component may result in radioactive releases, explosions, environmental damage/contamination, personal injury or death. In accordance with SBS Bit 3's terms and conditions of sale, the user of SBS Bit 3 components in any and all control and process applications assumes all risks arising out of such use and further agrees to indemnify and hold SBS Bit 3 harmless against any and all claims of whatsoever kind or nature (including claims of culpable conduct [strict liability, negligence or breach of warranty] on the part of SBS Bit 3) for all costs of defending any such claims.

SBS Bit 3 makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SBS Bit 3 assumes no responsibility for any errors that may appear in this document. The information in this document is subject to change without notice.

U.S. GOVERNMENT LIMITED RIGHTS
This documentation is provided with limited rights. Use, duplication or disclosure by the Government is subject to the restrictions as set forth in subdivision (b) (3) (iii) of the Rights in Technical Data and Computer Software Clause of DFAR252.227-7013 (October 1988) and in similar clauses in the FAR and NASA FAR Supplement. Manufacturer is SBS Bit 3 Operations, 1284 Corporate Center Drive, St. Paul, MN 55121-1245.

Revision 1.4 3/98
Pub. No. 85221678

Preface

This manual describes the SBS Bit 3 Model 617 adapter that connects a PCI bus system to a VMEbus system. It includes information about the adapter's operation, installation, configuration, and control registers.

To simplify installation and eliminate operation problems, SBS Bit 3 recommends that you review this manual before beginning to install your new adapter cards. Please pay close attention to the sections on card configuration and adapter registers.

- Chapter 1 provides an overview of the adapter, product description, specifications and requirements, and supporting products.
- Chapter 2 gets you started with information about unpacking the adapter package, adapter installation, Help, and additional references.
- Chapter 3 discusses basic bus issues, features that are common to both adapter cards, and cable conflict issues.
- Chapter 4 provides a broad overview of the PCI adapter card, including how the major features fit together and how they are used. This chapter also introduces the various memory windows.
- Chapter 5 talks about how to use PCI adapter card functions, including making VMEbus accesses, allowing VMEbus accesses, handling interrupts, initiating a DMA operation from PCI, and configuration registers.
- Chapter 6 describes Control and Status Registers (CSR) accessed from the PCI bus.
- Chapter 7 is an overview of the VMEbus adapter card.
- Chapter 8 deals with using the VMEbus adapter card functions, such as: making accesses to PCI, allowing PCI accesses, handling interrupts, and initiating a DMA operation from VMEbus.
- Chapter 9 describes Control and Status Registers (CSR) accessed from the VMEbus.

- Chapter 10 contains details for setting jumpers on the VMEbus adapter card.
- Chapter 11 provides suggestions and solutions for common problems with setting up and using the adapter.
- Chapter 12 includes information about the Utilities Diskette that comes with the adapter hardware.
- Appendix A is a glossary of terms used throughout this manual.
- Appendix B provides information about VMEbus addressing, including pin assignments and address modifiers.
- Appendix C discusses PCI BIOS functions.
- Appendix D contains a jumper configuration worksheet.

➔ **Important Notes:**

- Make sure you follow proper ESD handling procedures (refer to EIA-625, ESD Association Handbook, or MIL-HDBK-263) when working with cards and components.
- Be sure power is OFF before installing adapter cards.
- There is ***no*** "Quick Start" for using the Model 617 adapter. **Please read this manual thoroughly before trying to install or use the adapter.**

Table Of Contents

Chapter 1: Introduction

1.0 Overview

The SBS Bit 3 Model 617 adapter is an easy-to-use, cost-effective way to share memory and special purpose cards between a PCI bus system and a VMEbus system. The Model 617 adapter provides high-speed data transfers between the two systems, and requires minimal software support.

Model 617 interconnects the PCI and VMEbus systems at the physical layer. Working at the lowest level, the bus, the adapter allows the two systems to share memory; memory appears to and is treated by each system as if it were its own. Therefore, a card only available on one bus may be accessed and directly controlled by a system using another bus. For example, an Array Processor board in a VMEbus chassis can be directly controlled by the processor on the PCI bus.

Model 617 supports two methods of intersystem communications: Memory Mapping and Direct Memory Access (DMA). Memory Mapping supports bi-directional random access bus mastering from either system. This allows Programmed Input/Output (PIO) access to VMEbus RAM, dual-port memory, and VMEbus I/O, and provides an easy-to-use, flexible interface with low overhead. A PCI bus master can access memory in the VMEbus system through a window in PCI memory address space. Conversely, a VMEbus bus master can access PCI memory from a window in VMEbus address space.

Memory mapping is accomplished through 16,384 Mapping Registers that are used to steer memory accesses on one bus to the appropriate address on the other bus. The Mapping Registers allow PCI devices to access up to 32M bytes of VMEbus address space and VMEbus devices to access up to 16M bytes of PCI space. In addition, the Mapping Registers allow up to 16M byte DMA transfers.

The Model 617 adapter supports two DMA techniques: Controller Mode DMA and Slave Mode DMA.

Controller Mode DMA uses the adapter's DMA Controller to provide high-speed data transfers from one system's memory directly into the other system's memory. Data transfer can be initiated in both directions by either the PCI or VMEbus processor. Each DMA cycle supports transfer lengths up to 16M bytes. The DMA Controller also allows memory-to-memory transfers between PCI memory and Dual Port RAM on the VMEbus adapter card. Controller Mode DMA can sustain data rates up to 26 Megabytes per second (M Bytes/sec).

VMEbus devices that have their own DMA controllers can use Slave Mode DMA instead of Controller Mode DMA. Slave Mode DMA allows a VMEbus DMA device to transfer data through the adapter directly into PCI memory at data rates in excess of 12M Bytes/sec.

Model 617 does not link the timing of the two buses (so that activity on one bus slows down the other). Instead, the adapter permits each bus to operate *independently*. The buses are linked only when a memory or I/O reference is made to an address on one system that translates to a reference on the other system's bus.

The Model 617 adapter consists of two cards: a short form factor PCI card and a 6U size VMEbus card. The two cards are connected by a round EMI-shielded copper-conductor cable purchased separately from SBS Bit 3.

Cable is available in standard 8-foot or 25-foot lengths. Custom lengths may be ordered. Fiber-Optic Interface Cards and Modules are also available from SBS Bit 3.

An optional Dual Port RAM card that installs on the VMEbus adapter card is available from SBS Bit 3. The Dual Port RAM can be accessed by both systems and provides an inexpensive method of expanding PCI and VMEbus memory as well as a convenient way to share memory between the two systems. The following Dual Port RAM sizes are available: 32K, 128K, 1M, 2M, 4M, and 8M bytes.

1.1 Adapter Features

Bus Communication Specifics:
Both adapter cards are capable of remote bus mastership and allow simultaneous communication between the two chassis (except when a DMA transfer is in progress).

The PCI adapter card responds to and generates A32 memory and I/O accesses and supports D32, D16, and D8 data widths (Mapping Registers support only D32 and D16; CSRs support only D16 and D8).

The VMEbus adapter card responds to and generates A32, A24, and A16 accesses and supports D32, D16, and D8 data widths (A16 space supports only D16 and D8). D32 and D16 Block Mode transfers are also supported.

The VMEbus adapter card accepts and can generate all address modifier codes except VME64.

System Controller: The VMEbus adapter card can provide the system clock, reset, and bus error timeout feature.

Bus Arbitration: Provides Single-Level (SGL) or four-level Priority/Round-Robin (PRI/RRS) arbitration.

VMEbus: Release-On-Request (ROR);
Release-On-Bus-Clear.

Access Times:
PCI bus read/write access to remote RAM: about 2.2 µsec.
PCI bus read/write access to remote Dual Port RAM: about 2.1 µsec.
VMEbus read/write access to local Dual Port RAM: 400 nsec.

Slave Mode DMA Transfer Rate:
VMEbus Block Mode access to PCI RAM --
VMEbus writes to PCI bus: about 14M Bytes/sec.
VMEbus reads from PCI bus: about 12M Bytes/sec.

DMA Controller Transfer Rate*:

** This rate may vary significantly depending on PCI chip set implementation.*

The adapter is capable of sustained DMA data rates of:

26M Bytes/sec with 20 nsec Block Mode VMEbus memory cards;
20M Bytes/sec with 50 nsec Block Mode VMEbus memory cards;
10M Bytes/sec with 200 nsec Block Mode VMEbus memory cards;
13M Bytes/sec DMA from PCI bus to Dual Port RAM.

The maximum Burst Rate over the interface is 32M Bytes/sec.

Actual data rates measured from the application software level are also dependent on the clock frequency of the PCI I/O channel controller and system software overhead.

DMA data transfer block length: 2 bytes to 16M bytes.

Interrupt Passing:
All seven VMEbus interrupts can be passed to the PCI system.

Two types of programmed interrupts (PT and PR Interrupts) can be exchanged between the PCI adapter card and the VMEbus.

Interrupt Acknowledgment:
RORA (Release On Register Access).

PCI acknowledgment of VMEbus interrupts and VMEbus vector passing is provided through an adapter card control register.

PCI - VMEbus Timeout:
PCI bus to VMEbus transfer cycles timeout within 30 µsec. This results in an error bit being set and an interrupt generated if enabled.

Read-Modify-Write:
Read-modify-write from the PCI to the VMEbus system is supported via the PCI interface signal LOCK# and a CSR bit.

Read-modify-write from the VMEbus system into PCI memory is not supported.

Read-modify-write transactions to dual-port memory are also supported.

Conformance:
The VMEbus adapter card meets IEEE 1014C specifications.

The PCI card meets the PCI Local Bus specification version 2.0.

Power Requirements:
The VMEbus adapter card draws 3.5A at 5V.

The PCI adapter card draws 1.5A at 5V.

Environment:

Temperature:	0º to 60º C operating; -40º to 85º C storage.
Humidity:	0% to 90% non-condensing.

1.2 Supporting Products

Cables to connect the two adapter cards, Dual Port RAM cards, Fiber-Optic Interface Cards and Modules, and 6U-9U Holders are also available from SBS Bit 3. Please call 612-905-4700 for more information.

1.2.1 Cables

Standard shielded cables to connect the two adapter cards are available in 8-foot and 25-foot lengths. Custom-length cables and bulkhead options are also available from SBS Bit 3. *All cables are purchased separately.*

1.2.2 Dual Port RAM

Dual Port RAM is an optional memory card that attaches to the VMEbus adapter card and appears to both systems as simply more memory. The address of the Dual Port RAM is independently set on each adapter card, and may be set to respond to one address range in one system and a different range in the other. Both systems can access the memory at the same time with the VMEbus adapter card arbitrating simultaneous accesses.

SBS Bit 3's Dual Port RAM is a printed circuit card that plugs into the VMEbus adapter card as a daughter card. The following memory sizes are currently available: 32K, 128K, 1M, 2M, 4M, and 8M bytes.

1.2.3 Fiber-Optic Interface Cards And Modules

SBS Bit 3 Fiber-Optic Interfaces are available in two- and four-fiber cards and modules. Each card or module is one half of a high-performance fiber-optic link between the two SBS Bit 3 adapter cards. Fiber-optic cable lengths to 2 kilometers are supported.

Model 400-5 (Two Fiber) and 400-6 (Four Fiber) Fiber-Optic Interface cards are size 6U VMEbus printed circuit cards that install in VMEbus chassis. Models 400-50 (Two Fiber) and 400-60 (Four Fiber) are external Fiber-Optic Interface Modules with a chassis, power supply and installed fiber-optic interface card.

SBS Bit 3's Fiber-Optic Interfaces convert the electrical signals that are normally sent over a copper wire cable to light signals that travel over fiber-optic cables. In addition to greatly extending the distance between chassis, the Fiber-Optic Interfaces provide virtual immunity to electromagnetic interference.

Fiber-Optic Interfaces work transparently with SBS Bit 3 adapters and fully support the Model 617 adapter's feature set.

1.2.4 6U-9U Holder

A 6U-9U Holder for VMEbus systems is available from SBS Bit 3. When the 6U size VMEbus adapter card is installed in the Holder, it can be installed in a 9U sized slot in the VMEbus chassis.

1.3 System Controller Operation

The Model 617 adapter can act as a link between a PCI chassis and a VMEbus chassis even when the VMEbus chassis has no processor or system controller present. This form of operation is called System Controller Mode.

System Controller Mode is configured by setting the VMEbus adapter card's SYS jumper block to drive the system clock (SYSCLK) and the Bus Error (BERR) global timeout. The SBS Bit 3 VMEbus adapter card may be configured to be a Single-Level (SGL) bus arbiter or a four-level bus arbiter in Priority (PRI) or Round-Robin (RRS) mode.

1.4 Adapter Control And Status Registers (CSRs)

The CSRs allow PCI and VMEbus bus masters to control and obtain status information about the adapter. Each system has 32 CSRs that the adapter can access; half the CSRs are located on the local adapter card and half on the remote adapter card. VMEbus processors can access the VMEbus CSRs through 32 bytes of VMEbus I/O space. PCI processors can use either I/O space or PCI memory space to access their 32 CSRs.

1.5 Direct Memory Access (DMA)

DMA is the transfer of data from one memory address to another without processor intervention once the transfer starts. DMA logic is usually employed when large files or sections of data need to be moved.

The Model 617 adapter supports two types of DMA operations: Controller Mode DMA and Slave Mode DMA. In Controller Mode DMA, the adapter becomes a bus master on both the PCI bus and the VMEbus, and transfers data from memory on one system to memory on the other system. Controller Mode DMA transfers require very little processor attention and are a very fast means of transferring data.

Slave Mode DMA is performed when a VMEbus DMA device, such as a disk controller card, does a block transfer through the adapter directly into the PCI bus. In this case, the VMEbus adapter card looks like a slave memory card.

1.6 Interrupts

The adapter has four sources of interrupts; three can be generated by either adapter card, one is unique to the PCI adapter card.

Common interrupt sources:

- Programmed Interrupts - Each adapter card can receive and send a programmed interrupt from the other adapter card. Programmed interrupts are the basic method processors on the two dissimilar buses use to communicate and synchronize data transfers. There are two types of programmed interrupts: Programmed interrupt to Transmitter (PT) and Programmed interrupt to Receiver (PR).
- Interface Error Interrupt - Each adapter card can assert an interrupt when it detects that an error occurred.
- DMA Done Interrupt - Each adapter card can interrupt a local processor when the DMA has completed.

PCI adapter card specific interrupt source:

- VMEbus Backplane Interrupt - The PCI adapter card can interrupt the PCI bus whenever any of seven VMEbus interrupt levels are asserted. The adapter also allows a PCI processor to acknowledge the VMEbus interrupt and retrieve the interrupt acknowledgment vector.

1.7 Mapping Registers

The PCI adapter card is equipped with Mapping Registers that allow the adapter to take a large contiguous section of local memory and map it to many small non-contiguous sections of remote memory. This feature is very desirable because most of today's system architectures use virtual and paged memory management schemes. These schemes often satisfy an application's memory request with small sections of memory that are scattered throughout the memory space. The Mapping Registers allow a bus-to-bus bridge to remap these non-contiguous memory areas into a contiguous areas on the remote bus. This mapping works both for PCI to VMEbus accesses and VMEbus to PCI accesses.

Chapter 2: Getting Started

➔ Make sure you follow proper ESD handling procedures (refer to EIA-625, ESD Association Handbook, or MIL-HDBK-263) when working with cards and components.

2.0 Unpacking

The SBS Bit 3 Model 617 adapter package contains the following items. Please identify each item and notify SBS Bit 3 (612-905-4700) if any are missing:

- PCI circuit card - Part Number: 85221510
- VMEbus circuit card - Part Number: 85154558
- Utilities Diskette - Part Number: 85221680
- Model 617 manual - Pub. Number: 85221678
- One I/O cable to connect the two cards (purchased separately)
- Warranty card (please complete and mail)

➔ Eight-digit part numbers with card revision level are printed on white labels affixed to the adapter cards.

SBS Bit 3 adapter cards are shipped in static-safe packages to protect the components on the cards. Make sure you follow proper ESD handling procedures (refer to EIA-625, ESD Association Handbook, or MIL-HDBK-263) when working with cards and components.

2.1 No Fast Start

SBS Bit 3 strongly recommends that you read this manual completely before attempting to configure, install or use the Model 617 adapter.

2.2 Help!

Please have the following items and information handy when calling SBS Bit 3 for technical support:

- Model number and revision level of the adapter, or the serial number located on the white bar code label on the adapter cards.
- Size of Dual Port RAM, if any.
- Configuration information including jumper settings on the VMEbus adapter card. Record the jumper settings on the diagram in Appendix D.
- This manual.

Technical support is available from 9:00 a.m. - 5:00 p.m. (Central Time) Monday - Friday, excluding holidays.

Contact SBS Bit 3 at:

Mailing Address:	SBS Bit 3 Operations 1284 Corporate Center Drive St. Paul, MN 55121-1245
Phone:	612-905-4700
Fax:	612-905-4701
Email:	support@bit3.com

2.3 Installation

➔ Observe static safety precautions to prevent damage to the cards.

➔ Make sure power is *off* before installing cards.

2.3.1 Configure The Adapter Cards

Any required jumper configuration takes place before the adapter cards are installed. Refer to Chapter 10 for information about configuring the VMEbus adapter card. There are no jumpers to configure on the PCI adapter card.

2.3.2 Installing The PCI Adapter Card

1. Locate a vacant PCI card slot in the PCI chassis that supports a bus master.
2. Remove the metal plate that covers the cable exit at the rear of the chassis.
3. Insert the PCI adapter card into the connector.
4. Fasten the adapter card in place with the mounting screw.

2.3.3 Installing The VMEbus Adapter Card

➔ VMEbus backplanes have jumpers to connect the daisy-chained, bus grant and interrupt acknowledge signals around unused card locations. Make sure these jumpers are removed from the slot in which the adapter card will be installed.

1. Decide if the VMEbus adapter card is the system controller. If it is, it must be installed in slot 1.
2. Locate an unoccupied 6U slot in the VMEbus card cage if the adapter card is not the system controller.
3. Insert the card into the connector of the selected slot.

2.3.4 Connecting The Adapter Cable

The cable connectors on each adapter card are connected via I/O cable (ordered separately from SBS Bit 3).

To connect the I/O cable:

1. Match the A/B label on the cable connector to the label above the VMEbus adapter card faceplate. The connectors are keyed so that the cables cannot be installed incorrectly.

2. Connect the cable shield wire lug to the VMEbus chassis. (When removing the connectors, lift the retaining clamps -- *do not* pull on the cable.)

3. Plug the cable connector into its mate on the PCI adapter card.

4. Secure the cable with the two screws on the cable connector body.

5. Turn power on to both PCI and VMEbus systems.

➔ After installation, make sure the READY LEDs on both adapter cards are lit. They must be on for the adapter to operate.

2.4 Additional References

- The *VMEbus Specification Manual* is available from VITA (VMEbus International Trade Association), 7825 E. Gelding Drive, Suite 104, Scottsdale, AZ 85260-3415.

- *IEEE Standard 1014* is available from The Institute of Electrical and Electronics Engineers (IEEE), 445 Hoes Lane, Piscataway, NJ 08855-1331.

- *The PCI Local Bus Specification* is available from the PCI Special Interest Group, JF2-51, 5200 NE Elam Young Parkway, Hillsboro, OR 97124-6497.

- *PCI BIOS Specification* is available from the PCI Special Interest Group, JF2-51, 5200 NE Elam Young Parkway, Hillsboro, OR 97124-6497.

- *DOS Protected Mode Interface (DPMI) Specification Version 1.0* is available from Intel Corp.

- *Data Format and Bus Compatibility in Multiprocessors*, IEEE Micro, August 1983, is available from IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Chapter 3: The Model 617 Adapter

3.0 Introduction

The Model 617 adapter functions as a bridge between the PCI bus and VMEbus. The adapter allows PCI bus masters to become masters on the VMEbus, and VMEbus master to become masters on the PCI bus.

Because the adapter interconnects two dissimilar buses, it is important to understand both buses. Chapter 3 discusses basic bus issues, features that are common to both adapter cards, and cable conflict issues.

3.1 PCI Bus

The PCI bus is a 32-bit architecture designed for high-data throughput, self-configuration, and multiple bus mastership.

The PCI bus is self-configuring; therefore, all PCI devices are automatically configured by the startup firmware at system boot. The startup firmware sets up interrupt level routing and other hardware parameters, and resolves all addressing conflicts. Simply plug the PCI device into an open PCI slot and switch on the power; the system takes care of configuration.

The PCI specification supports multiple bus masters on the PCI bus. Consequently, the adapter allows VMEbus masters to become bus masters on the PCI bus, and supports Direct Memory Access (DMA) data transfers.

3.2 VMEbus

The VMEbus is a 32-bit bus that is widely used in industrial, commercial and military applications worldwide. An abundance of VMEbus cards are available to perform a wide range of tasks, from digital image processing to disk controllers. Like the PCI bus, the VMEbus supports multiple bus masters and high data transfer rates. Unlike the PCI bus, the VMEbus is not self-configuring.

A VMEbus consists of card cage with 1 - 21 slots, a backplane with two connectors and, normally, five jumpers per slot. The slots are numbered from 1 - 21. Slot 1 is the system controller slot. Cards with different functions are inserted in the slots to form a customized VMEbus chassis.

Sections 3.2.1 - 3.2.4 discuss several important unique VMEbus features: system controller operation, backplane jumpers, interrupt acknowledgment cycle, and address spaces.

3.2.1 System Controller Operation

A card that provides system controller functions must be installed in slot 1 of the VMEbus card cage. The system controller (usually a processor card) provides bus arbitration, checks for timeouts, and drives the system clock and system reset signals. The SBS Bit 3 VMEbus adapter card may be configured to be a Single-Level (SGL) bus arbiter or a four-level bus arbiter in Priority (PRI) or Round-Robin (RRS) mode.

The Model 617 adapter has the ability to act as a link between a PCI chassis and a VMEbus chassis even when the VMEbus chassis has no processor present. This form of operation is called System Controller Mode. System Controller Mode is selected on the VMEbus adapter card by configuring the SYS and BGO-BGI jumper blocks. These jumpers tell the VMEbus adapter card that it must provide the system controller functions, including: bus arbitration, timeout detection, and system level signals. See Chapter 10 for detailed descriptions of VMEbus adapter card jumper blocks.

A priority arbiter provides requesters preferential control of the data transfer bus over the other levels. By definition, BR3 is the highest priority, and BR0 is the lowest. When two or more requests are pending, the arbiter assigns control of the bus in the appropriate order by granting the bus in this sequence.

The priority arbiter must assert BCLR when a bus master of higher priority than the one in control of the bus initiates a request. When BBSY is asserted and a request is pending, the arbiter will drive BCLR if the pending request is of higher priority than the bus grant of the previous arbitration. Although the current bus master is not required to relinquish control of the bus in any prescribed time limit, it can continue transferring data until it reaches an appropriate stopping point.

A round-robin arbiter gives equal priority to all bus request levels. It grants control of the bus on a rotating basis. Upon release of the bus, the arbiter steps one level and tests for an active request and asserts a bus grant. If no request is active, it continues stepping through the levels until a request is found.

The RRS arbiter can drive the BCLR signal. In RRS mode BCLR is asserted whenever a master requests the bus on a level other than the last one granted. It does not assert BCLR if a master on the same level requests the bus.

➔ There must be one and only one card with System Controller Mode enabled. This card must be installed in slot 1.

➔ When the VMEbus adapter card is in System Controller Mode, except in special situations, the SYSCLK and SYSRESET jumpers must be installed. In most cases, the Detect Bus Timeout jumper should also be installed.

3.2.2 Backplane Jumpers

VMEbus chassis have five jumpers associated with each slot except slot 1. The five jumpers pass the bus grant and interrupt acknowledge signals to the next slot. If a slot is empty, the jumpers must be installed in order to pass the daisy-chained signals to the next slot. For example, if slots 1, 5 and 7 have cards installed, slots 2, 3, 4, and 6 must have the backplane jumpers installed or the system will not function properly.

➔ If a slot has no card installed and a card is installed in higher number slot, the backplane jumpers must be installed in the empty slot.

➔ Backplane jumpers should never be installed in slots in which cards are installed.

3.2.3 VMEbus Address Modifiers

The VMEbus specification defines three types of address spaces: extended (A32), standard (A24) and short (A16). Extended (A32) addressing uses 32 address bits. Standard (A24) addressing uses 24 address bits. Short (A16) addressing uses 16 address bits. The VMEbus uses special lines, the address modifier lines, to select which type of address space is being referenced.

Although the adapter does not respond to VME64 address modifiers, it can still be used in VME64 systems to do D32, D16, or D8 transfers with A32, A24, or A16 address modifiers.

Each address space is independent of the other address spaces and can be thought of as a logically separate address bus. A32 addressing uses address lines A31-01. A24 uses A23-01, and A31-24 are unused. A16 uses A15-A01, and A31-16 are unused. The VMEbus does not have an address 0 (A0) line. Byte addressing is controlled by the signals DS0 and DS1. Address lines A03-A01 are used during Interrupt ACKnowledge cycles (IACK cycles).

The following table summarizes the use of the address bus.

ACTIVE PORTION OF ADDRESS BUS - ADDRESS ROUTING				
A31 - A24	A23 - A16	A15 - A04	A03 - A01	Address Modifier Codes (hex)
A31 ----------A01				Extended (32-bit) 08 - 0F
	A23----------A01			Standard (24-bit) 38 - 3F
		A15----------A01		Short I/O (16-bit) 29, 2D
			A03 - A01	Interrupt Acknowledge

= Unused portion of address bus.

The value of the address modifier lines, AM[0..5], determines which address space is used. The VMEbus master is responsible for supplying the proper address modifier at the same time it drives the address lines. Slaves are designed to respond to cycles with a particular address modifier; however, most slaves are capable of responding to several address modifiers.

The address modifier generated when the PCI bus accesses the VMEbus is determined by the value in the selected Mapping Register. The programmer should determine which address modifier the target slave responds to, then program the Mapping Registers with that value.

The most commonly used address modifier codes are listed below. For a complete list of address modifiers, see Appendix B.

ADDRESS MODIFIER (HEX)	NUMBER OF ADDRESS BITS	TRANSFER TYPE
2D	16	Short supervisory access
3D	24	Standard supervisory access
3F	24	Standard supervisory block transfer
0D	32	Extended supervisory data access
0F	32	Extended supervisory block transfer

➔ The VMEbus adapter card can generate and respond to all address modifiers except D64.

3.2.4 VMEbus Interrupts And The IACK Cycle

Hardware devices use interrupts to indicate that they need attention or that some event has occurred. The VMEbus supports seven interrupt levels labeled IRQ1 to IRQ7. Any number of devices can use the same interrupt; however, only one VMEbus device can respond to an interrupt level.

Interrupts are responded to with an IACK cycle. The basic interrupt process is as follows:

1. A VMEbus device asserts one of the seven interrupt lines. For this example, we assume IRQ1.

2. The VMEbus device acknowledging IRQ1 receives control of the VMEbus and starts a byte, word, or longword read with the IACK signal asserted. This read should have address bits A3 - A1 equal to the level of the interrupt to be acknowledged. For example, IRQ1 is acknowledged by reading from location 2 (A3=0, A2=0, A1=1).

3. The system controller starts the IACK daisy-chain by sending IACKIN to slot 2. This daisy-chained IACK signal is passed up the slots until it reaches the card asserting IRQ1.

4. The device asserting IRQ1 responds to the IACK read cycle with the appropriate size data (the status/id value or IACK read vector).

5. The VMEbus device uses the IACK read vector to determine which device is interrupting and makes any necessary accesses to the interrupting device's registers to acknowledge the interrupt.

Two types of interrupting devices are supported by the VMEbus: Release On AcKnowledgment (ROAK) and Release On Register Access (RORA). A ROAK device removes its interrupt near the end of the IACK cycle (see step 4 above). A RORA device removes its interrupt after the IACK cycle occurred and one of its registers is accessed (see step 5).

For RORA devices, software must make a register access to switch off the interrupt before exiting the Interrupt Service Routine (ISR). Otherwise, the interrupting device will not remove its interrupt and the processor will go into an endless IACK loop.

3.3 Bridging PCI And VMEbus

The PCI and VMEbus adapter cards are connected by a cable that carries commands from one card to the other. Unless there is a software cable control scheme, it is possible for both cards to send commands at the same time. Since the PCI bus supports retry and the VMEbus (Rev. C) does not, the PCI adapter card usually gives precedence to VMEbus commands. The adapter responds to the PCI bus master with a target retry response and allows the VMEbus command to proceed. Ideally, the retried PCI bus operation follows. The single exception, a local register access, momentarily preempts all VMEbus commands.

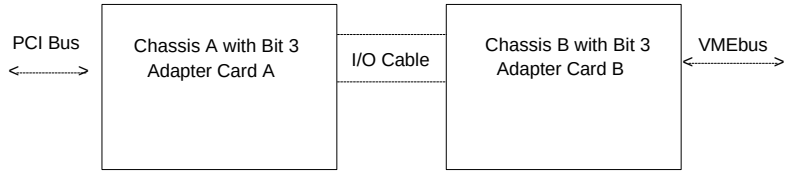
Even though the PCI bus supports retry, resolution of command collisions or cable conflicts may not be achieved on all PCI systems. This is not a design flaw with either the Model 617 adapter or the PCI system. The PCI specification is not defined to fully accommodate add-in cards that bridge to buses that do not have retry. Signals that resolve collisions are not available on the PCI connector. These signals, sideband signals, exist only between PCI bus components soldered directly to the motherboard. Sideband signals' functional definition is beyond the control of the PCI specification.

Command collisions to add-in board bridges can be accommodated by PCI systems if all processor initiated PCI bus requests are designed to back-off in the presence of a request from an add-in board. PCI systems that do not function in this manner exhibit bus livelock. Some workstations function free of bus livelock. Conversely, most personal computers exhibit bus livelock.

PCI bus livelock takes on two separate forms. The first occurs when the CPU becomes a PCI bus master and attempts to access the PCI adapter card. The PCI adapter card issues a target retry response because of a pending VMEbus command. When the PCI adapter card becomes bus master, it too receives a target retry response. Bus traffic momentarily consists of continuous retry responses from the PCI adapter card and the PCI system memory bridge. The second situation begins with the CPU becoming a bus master, and attempting to access the PCI adapter card. The PCI adapter card activates the bus request signal, attempting to become bus master, but is never granted access to the PCI bus. Bus traffic momentarily consists of continuous retry responses from the PCI adapter card. In either case, the PCI adapter detects bus livelock and returns an error response to the VMEbus adapter card. There is no alternative in either case but to activate the BERR* signal on the VMEbus. The CPU initiated operation to the PCI adapter will complete.

Installation of the PCI adapter card behind a PCI-to-PCI Bridge (PPB) presents additional potential for bus livelock. Specifically, if write posting is enabled in the PPB and the posted write buffer contains data, any read command from the PCI adapter card will be continually retried until the buffers are flushed. If the destination of the posted write buffer is the PCI adapter card, momentary bus livelock will occur. Upon detection of this condition, the PCI adapter card returns an error status back to the VMEbus adapter card, ultimately activating the BERR* signal on the VMEbus.

Two terms that deal with cable control are used throughout this manual: Transmitter and Receiver. When a card sends a command, it is a Transmitter. When a card receives a command, it is a Receiver.



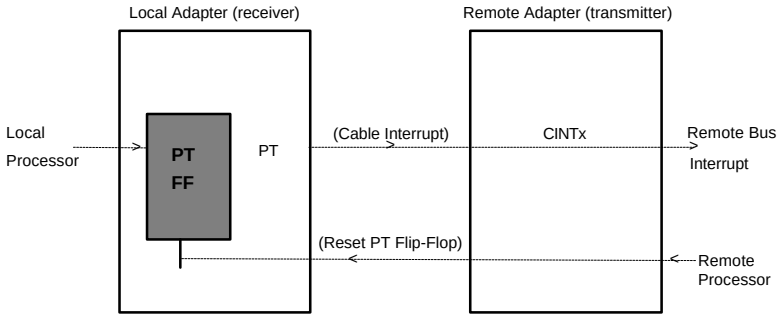
For example (see diagram above) --
Assume a device in chassis A performs reads or writes to memory or I/O on chassis B, but devices in chassis B never perform reads or writes to cards inside chassis A. In this case, adapter card A would be the *transmitter* and adapter card B would be the *receiver*.

The Model 617 adapter uses two methods of passing software or programmed interrupts between adapter cards: Programmed interrupts to Transmitter (PT Interrupts) and Programmed interrupts to Receiver (PR Interrupts). Because the Model 617 adapter has hardware cable conflict resolution, either method may be used without concern for cable conflicts.

3.3.1 Programmed Interrupt To Transmitter (PT)

The PT Interrupt allows an adapter card to *generate* an interrupt on the other adapter card's bus without making a remote cable access.

A local processor sets the Send PT Interrupt bit in a local adapter CSR. Adapter hardware then sends this interrupt request to the remote adapter card using a Cable Interrupt (CINT) line. The remote processor can then acknowledge the PT Interrupt by writing a remote adapter CSR.



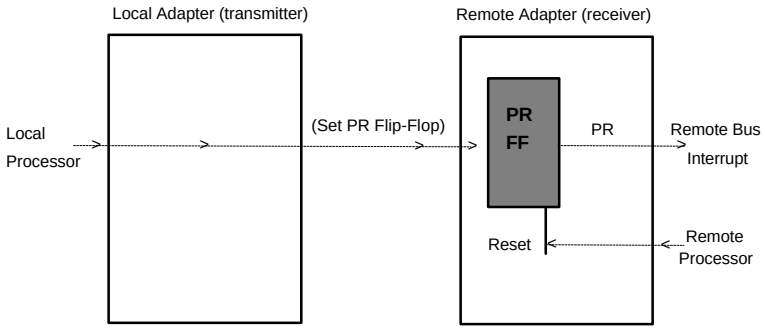
Programmed Interrupt To Transmitter
-- PT Interrupt --

See sections 5.5 and 8.5 for information on sending, receiving and acknowledging programmed interrupts.

3.3.2 Programmed Interrupt To Receiver (PR)

The PR Interrupt allows an adapter card to *receive* an interrupt and acknowledge it without a remote cable access.

A local processor sets the Send PR Interrupt bit in a remote adapter CSR. This causes an interrupt to be asserted on the remote bus. The remote processor can then acknowledge the PR Interrupt by writing a local adapter CSR.



Programmed Interrupt To Receiver
-- PR Interrupt --

See sections 5.5 and 8.5 for information on sending, receiving and acknowledging programmed interrupts.

3.3.3 Direct Memory Access (DMA)

Direct Memory Access, DMA, is the transfer of data from one memory address to another *without* processor intervention once the transfer starts. DMA logic is usually employed when large files or sections of data need to be moved. DMA is a highly efficient way to move data because it does not require processor overhead.

The SBS Bit 3 Model 617 adapter supports two types of DMA operations: Controller Mode DMA and Slave Mode DMA. In Controller Mode DMA, the adapter becomes a bus master on both the PCI bus and the VMEbus. Controller Mode DMA is used when the programmer wants the adapter to move a large block of data from one system to the other system. Transfer sizes from 2 bytes to 16M bytes are supported.

Slave Mode DMA is performed when a VMEbus DMA device, such as a disk controller card, does a Block Transfer through the adapter directly into the PCI bus. In this case, the VMEbus adapter card looks like a slave memory card.

The DMA Controller is accessible from either a VMEbus or PCI processor through the adapter node I/O space. Consequently, a single processor can perform all DMA setup and start commands on both the local *and* remote adapter cards.

➔ When a DMA is in progress, neither system may use any adapter features that require use of the cable; for example, remote node registers, remote Dual Port RAM, or remote memory.

3.3.4 How Controller Mode DMA Transfers Occur

Each adapter card has a DMA Controller. The DMA Controllers are somewhat independent of each other. While they are synchronized to pass data across the interface cable, they also independently perform reads and writes in their respective chassis.

The basic operations in a DMA transfer from local to remote memory (DMA write), once the DMA transfer has started, are as follows:

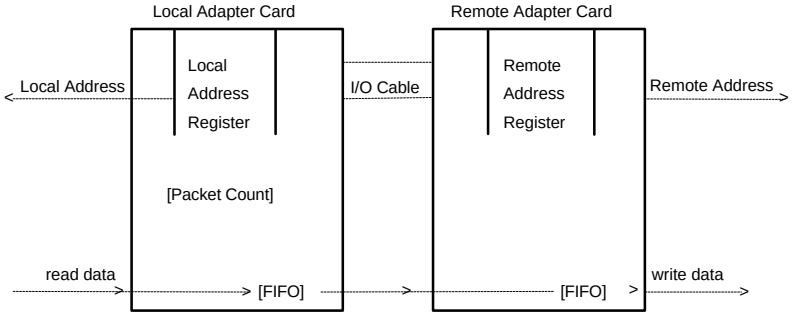
1. The local adapter card presents an address to its bus from the local address register and signals a read.

2. The locally read data are stored in the local FIFO and the address register (a counter) increments.

3. When the FIFO has a complete packet of information, the data are sent to the remote FIFO and the packet counter decrements.

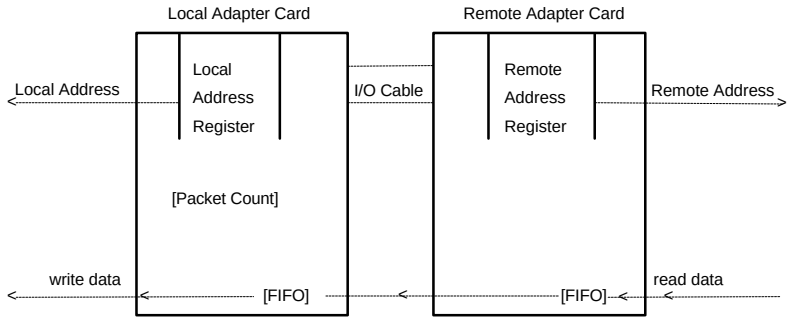
4. When the remote FIFO has a complete packet of information, the remote adapter card presents an address to the remote bus from the remote address register (also a counter) and signals a write. The data in the remote FIFO are written to its bus and the remote address register increments.

5. The packet transfer from the local adapter card to the remote adapter card continues until the local DMA packet count reaches zero.



DMA Controller Transfer From Local To Remote
-- DMA Write --

The Model 617 adapter runs DMA transfers from remote to local (DMA read) in much the same way as described above. The remote bus controller begins by reading remote data and sending them from the remote FIFO to the local FIFO, where the local adapter card writes local FIFO data to local memory.



DMA Controller Transfer From Remote To Local
--DMA Read --

This form of DMA provides a very efficient, high-speed transfer because each bus is allowed to run at maximum speed. All data are moved without the intervention of a processor on either side of the adapter.

A DMA transfer (read or write) can be initiated from either system.

3.3.5 Slave Mode DMA

Slave Mode DMA is performed when a VMEbus DMA device, such as a disk controller card, does a Block Transfer through the adapter directly into the PCI bus. In this case, the VMEbus adapter card looks like a slave memory card.

If the VMEbus DMA device does not use VMEbus Block Mode transfer protocol, the operation defaults to a standard random access read or write. The maximum speed for this type of transfer is 2M Bytes/sec.

If the DMA device (i.e. the disk controller card) uses VMEbus Block Mode transfers, transfer rates through the adapter of 12-14M Bytes/sec are possible.

Slave Block Mode DMA data transfers are supported only from VMEbus to PCI bus.

3.4 Accessing Windows

A window is a range of addresses that the adapter responds to for a specific function. The Model 617 adapter uses windows to access adapter functions.

Each of the adapter's windows has two properties: a unique starting address and size. The window's starting address is called its base address and is the lowest numerical address that hits the window. The window's size is the number of bytes past the base address that the window extends.

When discussing the base address of a window, the address is given as a physical address, the address that appears on the bus. The physical address may or may not be the same as the address (virtual address) the programmer sees. Most of today's Operating Systems and machine architectures use some type of memory management system that translates virtual addresses into physical addresses. Most Operating Systems provide functions that, given a physical address and length, will map a virtual address to this space.

Throughout this manual, there are references to accessing a specific offset within a window. For example, *write 0x80 to Remote Command Register 1 at offset 8*. This instruction translates to *write 0x80 to the physical address equal to the base of the Node Register Window plus 8*.

➔ Become familiar with the method your specific Operating System uses to map physical addresses to virtual addresses. To use the Model 617 adapter, you will need to map many PCI adapter windows.

3.5 Byte And Word Swapping

VMEbus and PCI systems store and transfer data differently. VMEbus systems store data in big endian format. PCI systems store data in little endian format. These differences cause data moved between the two systems to appear scrambled. For example, a PCI processor stores an integer differently than a VMEbus processor, therefore, an integer transferred from one system to the other will be misinterpreted by the receiving processor. To resolve this problem, the Model 617 adapter provides several methods of byte swapping.

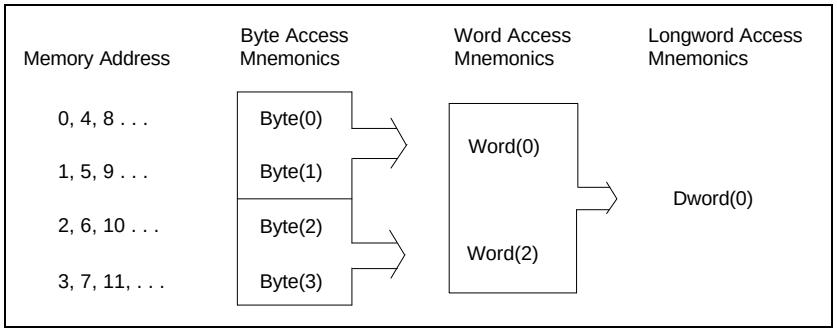
Byte swapping incorporates three primary issues:

- The access size -- 8-bit, 16-bit or 32-bit;
- Little endian versus big endian;
- The data size -- 8-bit, 16-bit, or 32-bit.

3.5.1 Data Accesses

Most buses support multiple transfer widths. Both VMEbus and PCI systems support byte, word, and longword accesses. This manual uses shorthand notations for describing a particular byte, word or longword. Byte(0) refers to bytes stored at addresses that are multiples of four. Byte (1) is defined as the byte after Byte(0), and so forth up to Byte(3). Word(0) is a word with an even starting address; it is composed of Byte(0) and Byte(1). Word(1) has an odd starting address and is composed of Byte(1) and Byte(2). Dword(0) is composed of Word(0), Word(2) and Byte(0) - Byte(3).

Data stored in a specific memory location is also described using a shorthand notation. For example, instead of *0x55 is stored in memory address 0*, the shorthand notation is *Byte(0) is 0x55*.



Mnemonics for Data Accesses

3.5.2 Little Endian Versus Big Endian

There are two popular ways to store multiple byte data in memory: big endian and little endian.

Big endian architectures store multiple byte data in consecutive memory locations with the most significant byte at the lowest numerical addresses. In this case, Byte(0) holds the *most significant* byte.

Little endian architectures store multiple byte data with the least significant byte at the lowest numerical address. Byte(0) holds the *least significant* byte.

The VMEbus is oriented towards big endian processors. The PCI bus is oriented towards little endian processors.

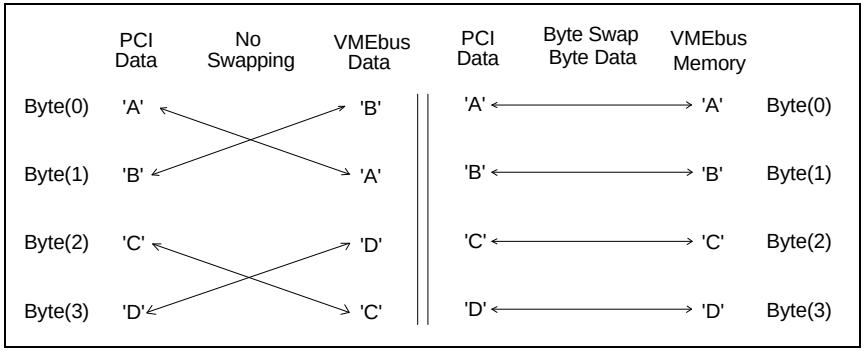
Data Type	Memory Location	Little Endian (PCI)	Big Endian (VMEbus)
String "ABCD"	Byte(0)	'A'	'A'
	Byte(1)	'B'	'B'
	Byte(2)	'C'	'C'
	Byte(3)	'D'	'D'
32 Bit Integer 12345678	Byte(0)	78	12
	Byte(1)	56	34
	Byte(2)	34	56
	Byte(3)	12	78
2 16 Bit Integers 1234, and 5678	Byte(0)	34	12
	Byte(1)	12	34
	Byte(2)	78	56
	Byte(3)	56	78

Big Endian Versus Little Endian

3.5.3 Swapping For Byte Accesses

For byte accesses across the adapter, the byte order is swapped. For example, a PCI processor performs a byte write of Byte(0) in PCI memory to VMEbus. If no swapping bits are active, a VMEbus processor looks in Byte(1) for the byte that was transferred.

The Model 617 adapter provides a Byte Swap On Byte Data bit that is used to correct the byte ordering problem when byte data are transferred across the adapter. When this bit is set, byte accesses across the cable are automatically swapped. For example, with the Byte Swapping On Byte Data bit set, a byte write of Byte(0) is stored in the remote memory's Byte(0).



Swapping for Byte Transfers of the String "ABCD"

➔ Set the Byte Swap Byte Data bit when transferring byte data as bytes between the two systems.

➔ Word swap and byte swap on non-byte data have no effect on byte accesses through the adapter.

➔ Byte data can be transferred as words or longwords if the Byte Swap On Non-Byte Data bit is set.

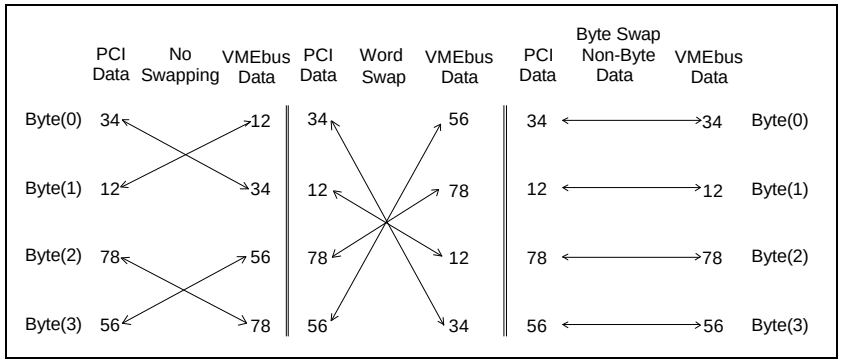
3.5.4 Swapping For Word Accesses

When words are transferred via the Model 617 adapter, the byte order is swapped just like for byte transfers. Because the VMEbus is big endian and the PCI is little endian, this byte swapping is desirable. For example, PCI memory contains the two byte number 1234: Byte(0) is 0x34 and Byte(1) is 0x12. The PCI processor then writes this word to VMEbus memory. If no byte swapping bits are set, the PCI adapter stores the data on the VMEbus as follows: Byte(0) is 0x12 and Byte(1) is 0x34.

If the default swapping is not acceptable to your application, the Model 617 adapter provides two other swapping methods for word accesses: swapping adjacent words and byte swap on non-byte data. These two methods can be used alone or combined.

Word swap causes adjacent words to be swapped as they pass through the adapter. For example, if the PCI writes Byte(0) and Byte(1) as a word to the VMEbus, PCI Byte(0) is stored in VMEbus Byte(3) and PCI Byte(1) is stored in VMEbus Byte(2).

In Byte Swap On Non-Byte Data Mode, the individual bytes within the word are swapped as they pass through the adapter. For example, if PCI writes Byte(0) and Byte(1) as a word to the VMEbus, PCI Byte(0) is stored in VMEbus Byte(0) and PCI Byte(1) is stored in VMEbus Byte(1).



Swapping for Word Transfers of the Numbers 1234 and 5678

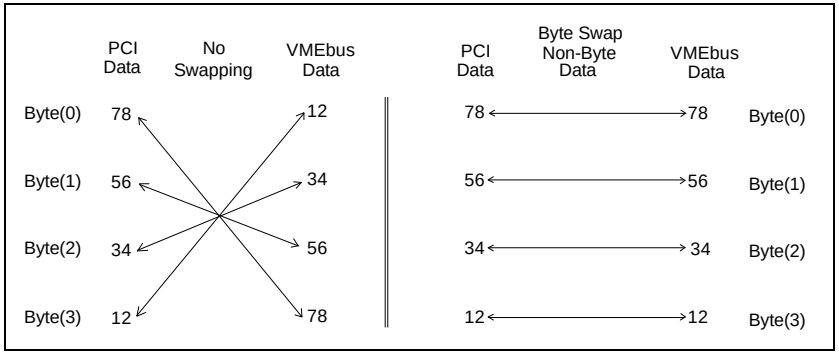
➔ No swapping bits need to be set if word data are transferred by bytes or words.

➔ Byte swap on byte data has no effect on word accesses.

3.5.5 Longword Accesses

When longword accesses are made through the Model 617 adapter, the byte order within the longword is reversed. For example, the longword 1234678 is stored in PCI memory as: Byte(0) is 0x78, Byte(1) is 0x56, Byte(2) is 0x34, and Byte(3) is 0x12. When this longword is transferred across the adapter, the bytes are completely swapped in VMEbus memory. VMEbus Byte(0) is 0x12, Byte(1) is 0x34, Byte(2) is 0x56, and Byte(3) is 0x78. Because the PCI bus and VMEbus are oriented towards different endian formats, this swapping is desirable.

Byte Swap on Non-Byte Data Mode is also available for longword accesses. In this mode, the byte ordering within each longword is maintained. A PCI value of 78563412 in PCI Byte(0) through Byte(3) is stored as 78563412 in VMEbus Byte(0) through Byte(3).



Swapping for Longword Transfers of the Number 12345678

➔ No swapping bits need to be set if longword data are transferred by longwords.

➔ Byte swap on byte data and word swap have no effect on longword accesses.

3.5.6 Access Width Versus Data Width

The following table shows how to set the Byte Swapping bit for different transfer lengths and different data sizes. This table assumes the PCI processor is little endian-based, and the VMEbus processor is big endian-based.

	SIZE OF DATA		
	Byte	Word	Longword
SIZE OF ACCESS			
Byte	Set Byte Swap Byte Data bit	No swapping	Software swapping required
Word	Set Byte Swap On Non-Byte Data bit	No swapping	Set Word Swap bit
Longword	Set Byte Swap On Non-Byte Data bit	Software swapping required	No swapping

Table of Common Byte Swapping Combinations

For more information about byte ordering and its history, we recommend the following article:

Data Format and Bus Compatibility in Multiprocessors, IEEE Micro, August 1983, PO Box 3014, Los Alamitos, CA 90720-1264.

Chapter 4: The PCI Adapter Card

4.0 Introduction

The Model 617 adapter has three distinct parts: the PCI adapter card, the VMEbus adapter card, and the cable. Chapter 4 provides a broad overview of the PCI adapter card, including how the major features fit together and how they are used. This chapter also introduces the various memory windows. Chapters 7 - 9 examine the VMEbus adapter card.

The main function of the PCI adapter is to allow PCI processors access to devices and memory installed in the VMEbus chassis. This allows a PCI system user to control a VMEbus chassis. Consequently, the developer or engineer can use the Operating System, compiler, and peripherals available for PCI systems, as well as access VMEbus cards.

The PCI adapter card has four major functional windows: Configuration Registers Window, Control and Status Registers (CSR) Window, Mapping Registers Window, and Remote Memory Window. Accesses to these windows tell the PCI adapter card what action to perform and how to perform it. For example: a write to the Mapping Registers Window tells the adapter card which VMEbus address to use for the byte read, and a byte read from the Remote Memory Window tells the adapter card to read a byte from the VMEbus.

Notes about the PCI adapter card:

- The PCI adapter card is completely self-configuring. All configuration is done automatically at boot time; there are no jumpers on the PCI adapter card.
- PCI bus masters can access up to 32M bytes of VMEbus memory space.
- PCI bus masters can receive and acknowledge any of the seven VMEbus interrupt levels as well as send and receive programmed interrupts.
- DMA transfers can be used to move data between PCI memory and VMEbus memory or Dual Port RAM at sustained data rates up to 26M Bytes/sec. Up to 16M bytes of data can be transferred with a single DMA.

4.1 Configuration Registers

The PCI adapter card conforms to the Configuration Register space as defined in Chapter 6, revision 2.0 of the PCI specification. The Configuration Registers that are necessary for adapter operation are discussed in section 5.7 of this manual. Other Configuration Registers defined in the PCI specification that are not necessary to adapter operation or those that cannot be configured by the user, are not discussed in this manual.

PCI is a self-configuring bus; therefore, there is no setup utility and no way to predetermine how the adapter card will be configured in a particular machine. The Configuration Registers, required by the PCI specification, facilitate the self-configuration. The Configuration Registers contain information that not only self-configures the card but also provides information about the card to the user. For example, the programmer uses the Configuration Registers to determine if a card is installed, the base addresses of the card's memory windows, the interrupt level assigned to the card, and the current status of the card.

Configuration Registers are used to configure the SBS Bit 3 PCI adapter card. These registers can be placed in four categories according to function:

.

- Registers that assist in locating the card -
  - Vendor ID,
  - Device ID,
  - Class Code,
  - Revision ID.
- Registers that tell the location in memory or I/O space of the various windows -
  - I/O Mapped Node I/O Base Address Register,
  - Memory Mapped Node I/O Base Address Register,
  - Mapping Register Base Address Register,
  - Remote Memory Base Address Register.

- Registers that define which interrupt the PCI adapter card is using -
  - Interrupt Line Register.
- Registers that tell the status of the PCI adapter card -
  - Status Register.

Access to the Configuration Registers is dependent on the specific platform. For example, DOS machines allow read and write access to any slot's Configuration Registers through BIOS functions that are defined in the PCI BIOS Specification.

➔ Programmers: Become familiar with your platform's method of accessing Configuration Registers. To map and access various windows, you will need to read their base addresses from the Configuration Registers.

See section 5.7 for detailed information about each Configuration Register. See Appendix C for an explanation of the PCI BIOS functions required to read the Configuration Registers and to find the PCI adapter card.

4.2 PCI CSR

There are 32 CSRs that determine how the PCI adapter card functions and report its current status. The CSRs can be accessed two ways: through the I/O space, and through normal memory accesses.

When accessing CSRs through I/O space, the base I/O address can be found by reading the I/O Mapped Node I/O Base Address Configuration Register (configuration longword 0x10). Then CSR at offset X can be accessed by an I/O read or write at the base address plus X.

When accessing CSRs through normal memory accesses, the memory mapped base address is read from the Memory Mapped Node I/O Base Address Configuration Register (configuration longword 0x14). The CSRs can then be accessed as an offset from a pointer that points to the base physical address.

There are 16 local CSRs and 16 remote CSRs. Local CSRs are physically located on the PCI adapter card and do not use the cable when they are accessed. Remote CSRs are physically located on the VMEbus adapter card and a cable access is generated when any remote CSR is read or written.

The 32 CSRs are organized into four groups of eight registers each: local general CSRs, remote general CSRs, local DMA CSRs, and remote DMA CSRs. The local general CSRs are used for controlling adapter features that are implemented on the PCI adapter card as well as for reporting the current status of the PCI adapter card. The remote general CSRs are used for controlling features of the adapter that are implemented on the VMEbus adapter card and for determining the card's current status. The local DMA registers are used for setting up, starting and checking the status of a DMA operation. The remote DMA registers are used for setting up DMA parameters for the VMEbus adapter card. See Chapter 6 for descriptions of each register.

4.3 Mapping Registers

The Mapping Registers provide the details for translating accesses across the adapter, such as: the address to access, address modifier (if the access is to the VMEbus chassis), and if byte swapping should be performed.

The base physical address of the Mapping Registers is read from the Mapping Register Base Address Configuration Register (0x18). To read a specific register, the programmer must read from the address that is equal to the base address of the Mapping Register Window plus the offset of the selected register.

The Mapping Register Window is 64K bytes long and is comprised of 16,384 4 byte registers (4 bytes per register * 16,384 register = 64K bytes). Each register may be accessed either by two words or one longword. Byte accesses to these registers are not allowed.

The Mapping Register Window is divided into three separate sections.

OFFSET FROM BASE (hex)	DESCRIPTION	NUMBER OF REGISTERS
0000 0000 - 0000 7FFF	PCI Bus to VMEbus Mapping Registers	8K
0000 8000 - 0000 BFFF	VMEbus to PCI Bus Mapping Registers	4K
0000 C000 - 0000 FFFF	DMA to PCI Bus Mapping Registers	4K

The first 8,192 registers control how accesses to the Remote Memory Window are translated to the VMEbus. The next 4,096 register control how accesses from the VMEbus are translated to the PCI bus. The remaining 4,096 registers control how DMA requests access the PCI bus.

See Chapter 5 for detailed information about the Mapping Registers.

4.4 Remote Memory Window

The Remote Memory Window is used by PCI devices to generate accesses on the VMEbus or to optional Dual Port RAM. A PCI device can read or write a byte, word, or longword to any location within this 32M byte window. The access then generates a byte, word, or longword read or write on the VMEbus or Dual Port RAM.

The base PCI address of the Remote Memory Window is found by reading the PCI Remote Memory Base Address Configuration Register (0x1C).

The Remote Memory Window is divided into 8,192 4K byte sections. Each section corresponds to a Mapping Register that provides the details for translating a PCI access within this section to a corresponding VMEbus access. For example, if the base address of the Remote Memory Window is at 0x80000000 and an access is made to address 0x80004024, the fourth Mapping Register is used to calculate the VMEbus address, address modifier, etc. for the access ((0x80004024 - 0x80000000) / 0x1000 = 4). The VMEbus address in the fourth Mapping Register is added to 0x24 (0x80004024 mod 0x1000 = 24) and a byte read at this address is performed on the VMEbus.

4.5 PCI Adapter Card LEDs

On the PCI adapter card:

- The LED labeled RDY is on when the programmable logic arrays on the PCI adapter card are successfully loaded after power-on. *The RDY LED must be on for the card to operate.*

- The LED labeled REM is on when the PCI adapter card is a bus master (PIO or DMA) performing an operation initiated by a VMEbus device.

- The LED labeled LOC is on when the PCI adapter card is a bus slave (PIO) or a bus master (DMA) performing an operation initiated by a PCI device.

Chapter 5: Using PCI Adapter Card Functions

5.0 Introduction

Chapter 5 contains information about how to use PCI adapter card functions, including making VMEbus accesses, allowing VMEbus accesses, handling interrupts, initiating a DMA operation from PCI and Configuration Registers.

5.1 Finding And Mapping the Adapter

Before any of the adapter features can be used, the PCI adapter card must be located and its Configuration Registers read. The method used to find the adapter card and its Configuration Registers is dependent on the Operating System and the system architecture. The basic steps are as follows:

1. Find the PCI adapter card on the PCI bus. Use whatever means are available to your system.

2. Use the information returned in step 1 to read the following PCI adapter card Configuration Registers:

   - Read the Memory Mapped Node I/O Base Address Register (configuration longword 0x14) and obtain a software pointer to this physical address for 32 bytes. The least significant 4 bits of this register should be masked to zero.

     ➔ If you prefer to use I/O space to read and write adapter node I/O registers, I/O Mapped Node I/O Base Address (configuration longword 0x10) can read instead of the memory mapped one in this step. The least significant 4 bits of this register should be masked to zero.

   - Read the Mapping Register Base Address (configuration longword 0x18) and obtain a software pointer to this physical address for 64K bytes. The least significant 4 bits of this register should be masked to zero.

- Read the Remote Memory Base Address (configuration longword 0x1C) and obtain a software pointer to this physical address for 32M bytes. The least significant 4 bits of this register should be masked to zero.

➔ The routines used to find PCI cards and access their Configuration Registers in 80x86 architectures is documented in Appendix C.

5.2 Initialization

Before the PCI adapter card's functions can be used, the following set up sequence must be performed to initialize the adapter:

1. Read the Local Status Register to make sure bit 0 is clear. This indicates that the remote power is on and that the cable is connected. If the remote power is off or the cable is disconnected, most of the adapter functions are useless. Any attempts to access remote resources result in interface errors.

2. Read the Remote Status Register and discard the results. This clears the cable of interface errors caused by the power on transition.

3. Set bit 7 of the Local Command Register to clear status errors that have been recorded since the adapter was powered on.

4. Read the Local Status Register to make sure no error bits are set.

5.3 Accessing Remote Memory

The PCI adapter allows memory cycles that occur on the PCI bus to be translated into memory cycles on the VMEbus or to the optional Dual Port RAM card. This important feature allows any PCI processor to communicate with any VMEbus device. The adapter can translate PCI accesses into VMEbus accesses to any address within any VMEbus address space or to an installed Dual Port RAM. Programmed I/O (PIO) accesses in the following data widths are supported: byte, word, and longword.

The Remote Memory Window is used when a PCI processor wants to access VMEbus memory or Dual Port RAM. A byte, word, or longword read or write to an address within the Remote Memory Window is translated to a byte, word, or longword read or write on the VMEbus or Dual Port RAM. The data width and cycle type are always preserved across the adapter. Consequently, a byte read from the Remote Memory Window corresponds to a byte read from the VMEbus or Dual Port RAM.

The Remote Memory Window is 32M bytes long. It is always contiguous. The 32M byte window can be logically divided into 8,192 4K byte windows. Each 4K byte window is associated with its own Mapping Register. Therefore, a PCI access to an address that is 16K bytes from the start of the Remote Memory Window uses the fourth Mapping Register to tell how this access should be translated to the VMEbus.

5.3.1 VMEbus Memory

VMEbus memory is divided into A16, A24 and A32 address spaces. The names indicate how many address bits are used when a PIO transfer is directed toward that memory space. For example, address bits A15 - A1 are used for a PIO transfer to A16 address space (VMEbus does not have an A0 bit).

The VMEbus address modifier determines which memory space is addressed. It tells the adapter card being addressed how many address lines to look at and the type of memory cycle. See section 3.2.3 for more information on VMEbus memory and address modifiers.

➔ When accessing the VMEbus, you must know which of the four address spaces you want to access: A16, A24, A32 or Dual Port RAM.

5.3.2 Dual Port RAM

Optional Dual Port RAM is additional memory that can be accessed by either the PCI or VMEbus without linking the two buses. Both adapter cards can access Dual Port RAM at the same time; hardware arbitrates the accesses.

➔ Accesses to Dual Port RAM ignore the address modifier settings.

5.3.3 Mapping Register Window

The 8,192 Remote Memory Mapping Registers control how accesses to the Remote Memory Window are translated to the VMEbus. These 8,192 registers are the only ones that affect the Remote Memory Window and control how accesses to the VMEbus are done.

Each Remote Memory Mapping Register is responsible for a corresponding 4K byte window of the Remote Memory Window. For example, the eighth Remote Memory Mapping Register governs how accesses from 0x8000 to 0x8FFF within the Remote Memory Window are translated to the VMEbus.

OFFSET FROM BASE (hex)	DESCRIPTION	NUMBER OF REGISTERS
0000 0000 - 0000 7FFF	**PCI Bus to VMEbus Mapping Registers**	**8K**
0000 8000 - 0000 BFFF	VMEbus to PCI Bus Mapping Registers	4K
0000 C000 - 0000 FFFF	DMA to PCI Bus Mapping Registers	4K

The Remote Memory Mapping Registers control the following items: the value of the upper VMEbus address; if A16, A24, A32, or Dual Port RAM is accessed; the address modifier used; and how byte swapping is performed.

The Remote Memory Mapping Register format is detailed in section 5.3.3.1.

5.3.3.1 Remote Memory Mapping Register Format

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	A24

D23	D22	D21	D20	D19	D18	D17	D16
A23	A22	A21	A20	A19	A18	A17	A16

D15	D14	D13	D12	D11	D10	D9	D8
A15	A14	A13	A12	Address Modifier 5	Address Modifier 4	Address Modifier 3	Address Modifier 2

D7	D6	D5	D4	D3	D2	D1	D0
Address Modifier 1	Address Modifier 0	Function Code 1	Function Code 0	Byte Swap Byte Data	Word Swap	Byte Swap Non-Byte Data	RAM Invalid

Bit 0 (Map Register Invalid): To enable PCI bus to VMEbus access, this bit must be reset to "0". If it is set to "1", bus timeouts occur. This bit is indeterminate following power-up.

Bit 1 (Byte Swap On Non-Byte Data Enable): When this bit is set to "1", byte swapping occurs on word and longword transfer operations.

Bit 2 (Word Swap Enable): When this bit is set to "1", address bit A1 is inverted for word transfer operations.

Bit 3 (Byte Swap On Byte Data Enable): When this bit is set to "1", byte swapping is enabled for byte transfer operations.

➔ For more information about swapping, see section 3.5.

Bits 4 & 5 (Function Code): These two bits define the destination of the transfer. The following table defines the function code values:

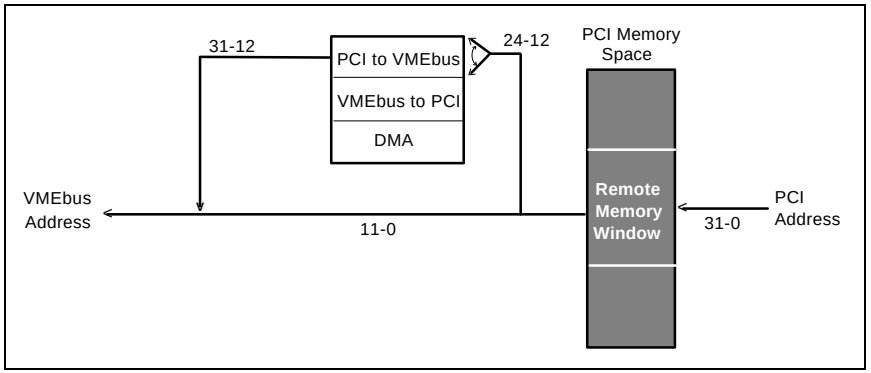
FC1	FC0	FUNCTION
0	0	Reserved
0	1	Access remote bus I/O
1	0	Access remote bus RAM
1	1	Access remote Dual Port RAM

Bits 6 - 11 (Address Modifier 0 - 5): These six bits are used to generate the address modifier for PCI bus to VMEbus access.

Bits 12 - 31 (Remote Address A12 - A31): The VMEbus address or Dual Port RAM address is formed by combining the low 12 bits of the PCI address with the upper 20 bits of this register.

5.3.4 PCI To VMEbus Address

The following diagram shows how the Remote Memory Window and Mapping Registers interact to generate the remote address.



When a PCI memory cycle is generated with a PCI address that falls within the range of the Remote Memory Window, the PCI adapter card uses the PCI address as follows:

1. The base address of the Remote Memory Window is subtracted from the PCI address. The result is divided by 4,096 to determine which PCI to VMEbus Mapping Register to use.

2. The address sent to the VMEbus adapter card is formed by combining bits 31 - 12 of the selected Mapping Register and bits 11 - 0 of the PCI address.

3. The address from step 3, along with the address modifier, function codes, and byte swapping bits from the selected Mapping Register, is sent to the VMEbus adapter card to generate the proper VMEbus memory or Dual Port RAM cycle.

5.3.5 Example of Accessing VMEbus

In this example, a VMEbus disk controller's buffer is accessed and 8K bytes of data are copied to Dual Port RAM. The actual values given below are only examples and are not the values that your PCI system will return.

1. Read the Configuration Register at 0x1C to obtain the physical address of the Remote Memory Window. Returns 0x80000000.

2. Read the Configuration Register at 0x18 to obtain the physical address of the Mapping Register Window. Returns 0x82000000.

3. Obtain a pointer to the base of the Remote Memory and Mapping Register Windows. For this example, virtual and physical addresses are the same. This step is Operating System dependent.

4. Initialize the adapter. See section 5.2.

5. Initialize the Mapping Registers so that the first 8K bytes of the Remote Memory Window point at the disk controller.

   Write 0x12340368 to 0x82000000 and 0x12341368 to 0x82000004. 0x12340000 is the starting VMEbus address. 0x0D is the address modifier. Remote Bus RAM is the function code. Byte swap, byte data is set.

6. Initialize the next two Mapping Registers to point to the first 8K bytes of Dual Port RAM.

   Write 0x00000038 to 0x82000008 and 0x00001038 to 0x8200000C. The starting Dual Port RAM address is 0. The address modifier does not matter. Dual Port RAM is the function code. Byte swap, byte data is set.

7. 0x80000000 now points to the first location of the disk controller's buffer and 0x80002000 points to the first location of the Dual Port RAM.

8. Copy from 0x80000000 to 0x80002000 for 8K bytes by bytes.

➔ The Utilities Diskette includes programming examples that illustrate use of the Remote Memory Window and Mapping Registers. The dumplram example program illustrates accesses to VMEbus memory. The dumpport example program illustrates accesses to Dual Port RAM. See Chapter 12 for information about the Utilities Diskette.

5.4 Allowing VMEbus Accesses

Because PCI allows multiple bus masters, the PCI adapter card allows VMEbus bus masters to access PCI resources. The PCI adapter card translates VMEbus accesses that fall within a particular jumper selectable window to the corresponding PCI accesses. This allows VMEbus processors to manipulate resources on the PCI bus or store data in PCI memory.

Three activities must be completed to allow VMEbus bus masters access to PCI memory:

- Set the Remote RAM jumpers on the VMEbus adapter card;
- Obtain a portion of PCI memory and find its physical address;
- Program the correct VMEbus to PCI Mapping Registers.

5.4.1 Setting Up PCI Memory

For VMEbus bus masters to use PCI resources or memory, the PCI physical address of the resource to be accessed must be determined. Normally, to determine the PCI physical address, a programmer will malloc() a buffer on PCI and obtain a virtual address to this buffer. Next, the physical address of the buffer must be found so that the PCI adapter card's Mapping Registers can be programmed to point to the buffer. After the Mapping Registers are programmed, the PCI and VMEbus can share data in this PCI memory buffer.

Determining a PCI physical address from the buffer's virtual address can be difficult. Each Operating System supplies supporting routines that assist in virtual address to physical address conversion. For windows, the routine CopyPageTable() provides the information required to calculate physical addresses for each virtual address. In UNIX environments, a device driver handles all physical address calculations. For DOS, the Utilities Diskette provides a DOS routine to imitate the windows CopyPageTable() routine and examples of its use.

After the PCI memory physical address is determined, it is used to program the Mapping Registers so that the upper address bits for access from the VMEbus can be modified to access the correct PCI address.

5.4.2 VMEbus Remote RAM Window

The VMEbus Remote RAM Window is similar to the PCI Remote Memory Window. This window allows the VMEbus to access PCI memory by reading or writing to the VMEbus Remote RAM Window. The window is configured by setting jumpers on the VMEbus adapter card. It can be located anywhere in memory and can be up to 16M bytes long.

5.4.3 Mapping Register Window

VMEbus-to-PCI Mapping Registers, the second segment of the Mapping Register Window, are used to control VMEbus accesses into PCI memory space. The VMEbus can access a total of 16M bytes of PCI memory space via 4,096 VMEbus-to-PCI Mapping Registers. Each Mapping Register is responsible for redirecting a 4K byte page of VMEbus memory so that it accesses the appropriate 4K byte page of PCI memory.

OFFSET FROM BASE (hex)	DESCRIPTION	NUMBER OF REGISTERS
0000 0000 - 0000 7FFF	PCI Bus to VMEbus Mapping Registers	8K
0000 8000 - 0000 BFFF	**VMEbus to PCI Bus Mapping Registers**	**4K**
0000 C000 - 0000 FFFF	DMA to PCI Bus Mapping Registers	4K

The Mapping Registers function as follows:

1. A VMEbus processor makes an access to its Remote RAM Window.

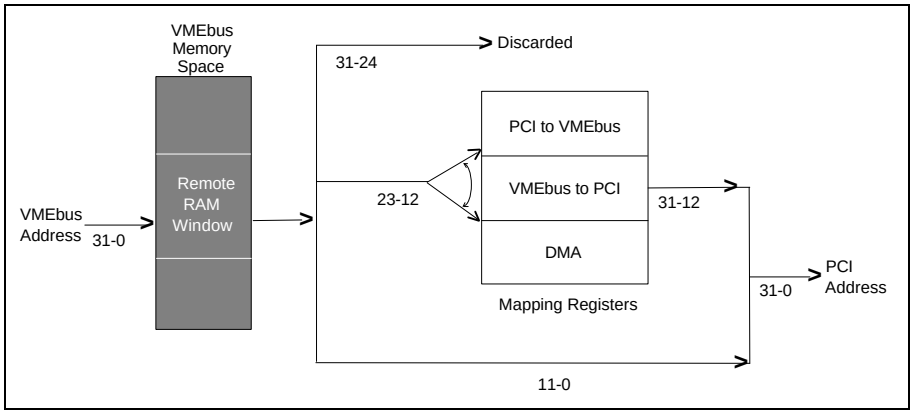
2. The VMEbus address is sent over the cable to the PCI adapter card.

3. The PCI adapter card passes address bits A11 - A0 straight to the PCI bus.

4. Address bits A23 - A12 are an index into the VMEbus-to-PCI Mapping Register table.

5. The selected Mapping Register supplies A31 - A12 of the PCI address lines and any byte swapping that should be performed.

6. The access is made at the calculated address and any data are returned to the VMEbus.



VMEbus-to-PCI Mapping Registers

Be careful when filling Mapping Registers to allow the VMEbus to access the PCI system. A common mistake is to assume that an access to the first address of the VMEbus Remote RAM Window will use the first VMEbus-to-PCI Mapping Register. Sometimes another Mapping Register is used. For example, if the starting address of the Remote RAM Window is at 0x80400000 and a VMEbus processor makes an access to 0x80400000, the address 0x80400000 is sent to the PCI adapter card. Address lines A11 - A0, 0x000, are passed straight to the PCI bus. Address lines A23 - A12, 0x400, are used to index into the VMEbus-to-PCI Mapping Registers. Therefore, this access uses the 1,024th VMEbus-to-PCI Mapping Register at offset 0x9000 from the base address of the Mapping Register Window (0x400 registers * 4 bytes/register = 0x1000 bytes from the start of the VMEbus-to-PCI Mapping Registers; this is 0x8000 bytes from the start of the Mapping Register Window so 0x8000 + 0x1000 = 0x9000).

➔ The simplest way to eliminate the VMEbus address Mapping Register calculation problem is to make sure the VMEbus Remote RAM Window always starts on a 16M byte boundary. Then the first location of VMEbus remote RAM will always use the first VMEbus-to-PCI Mapping Register and the 0x1000 location of the Remote RAM Window will always use the second Mapping Register, and so on.

See section 5.4.3.1 for VMEbus-to-PCI Mapping Register bit definitions.

5.4.3.1 VMEbus-To-PCI Bus Mapping Register Format

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	A24

D23	D22	D21	D20	D19	D18	D17	D16
A23	A22	A21	A20	A19	A18	A17	A16

D15	D14	D13	D12	D11	D10	D9	D8
A15	A14	A13	A12	Reserved	Reserved	Reserved	Reserved

D7	D6	D5	D4	D3	D2	D1	D0
Reserved	Reserved	Reserved	Reserved	Byte Swap Byte Data	Word Swap	Byte Swap Non-Byte Data	RAM Invalid

Bit 0 (Map Register Invalid): To enable VMEbus to PCI bus access, this bit must be reset to "0". If this bit is set to "1", a VMEbus error will occur. This bit is indeterminate following power-up.

Bit 1 (Byte Swap On Non-Byte Data Enable): When this bit is set to "1", byte swapping occurs on word and longword transfer operations.

Bit 2 (Word Swap Enable): When this bit is set to "1", address bit A1 is inverted for word transfers.

Bit 3 (Byte Swap On Byte Data Enable): When this bit is set to "1", byte swapping is enabled for byte transfer operations.

➔ For more information about byte swapping, see section 3.5.

Bits 4 - 11: Reserved.

Bits 12 - 31 (Local Address A12 - A31): The PCI bus address is formed by combining the low 12 bits of the VMEbus address with the upper 20 bits of this register.

5.4.4 Example: Allowing VMEbus Accesses

In this example, a 20K byte PCI memory buffer is set up so that a VMEbus-based disk controller can write data directly to PCI memory. The actual values given below are only examples and are not the values your PCI system will return.

1. Create a 20K byte memory region and find its starting physical address. Make sure the memory cannot be paged out to the disk. In this example, the system's virtual addresses and physical addresses are the same and the PCI memory buffer is located from 0x800000 to 0x805000. *This step is Operating System dependent.*

2. Read Configuration Register 0x18 to obtain the physical address of the Mapping Register Window. Returns 0x82000000.

3. Obtain a pointer to the base of the Mapping Register Window. Because virtual and physical addresses are the same for this example, the pointer's value is 0x82000000.

4. Initialize the PCI adapter card.

5. The VMEbus Remote RAM Window starts at address 0x40A00000 and extends for 20K bytes. Therefore, VMEbus-to-PCI Mapping Registers 2560 - 2564 must be programmed (0x40A00000 ### 0xA00 ### 2560).

6. Program the 2560th VMEbus-to-PCI Mapping Register. Write 0x800000 to location 0x8200A800. 0x800000 is the physical address of the PCI memory buffer with no byte swapping bits set. 0x8200A800 is calculated from 0x82000000 (base address of Mapping Register Window) + 0x8000 (offset of the VMEbus-to-PCI Mapping Registers within the Mapping Register Window) + 0x2800 (offset of the 2560th Mapping Register 0xA00*4=0x2800).

7. Program the 2561st VMEbus-to-PCI Mapping Register. Write 0x801000 to location 0x8200A804.

8. Program the 2562nd VMEbus-to-PCI Mapping Register. Write 0x802000 to location 0x8200A808.

9. Program the 2563rd VMEbus-to-PCI Mapping Register. Write 0x803000 to location 0x8200A80C.

10. Program the 2564th VMEbus-to-PCI Mapping Register. Write 0x804000 to location 0x8200A810.

11. VMEbus processors can now access the 20K byte buffer of PCI memory by writing to the Remote RAM Window

5.5 Handling Interrupts

Interrupts allow hardware to get the attention of an application or Operating System. Interrupts are used when a hardware device needs to be serviced or to signal that a particular event occurred.

When hardware asserts an interrupt, the processor is interrupted on a particular level that was assigned to that hardware device. After the processor is interrupted, it executes a software Interrupt Service Routine (ISR) that is associated with that particular interrupt level. The ISR tells the hardware to stop interrupting and resolves any outstanding hardware issues.

The PCI adapter card is assigned to interrupt on PCI INTA# which will be routed to a specific processor interrupt level by the system at boot time. To determine which level the PCI adapter card will interrupt on, the Interrupt Line Configuration Register at offset 0x3C must be read. This Configuration Register indicates the interrupt level at which the ISR needs to be installed.

When the PCI adapter card issues an interrupt, the ISR is expected to perform various CSR accesses to determine the cause of the interrupt and then make a CSR access to remove the source of the interrupt. The PCI adapter card has four different sources of interrupts:

- Interrupts from the VMEbus chassis backplane interrupts.
- Programmed interrupts.
- An Interface Error Interrupt.
- A DMA Done Interrupt.

When the PCI adapter card is generating an interrupt, bit 7 of the Local Interrupt Control Register is active. This bit can be used by the ISR on systems that share interrupt levels to determine if the PCI adapter card has an active interrupt.

➔ To receive an interrupt from the PCI adapter card, the appropriate enable bit must be set in the Local Interrupt Control Register. For Error Interrupts to be received, bit 5 must be set. For all other PCI adapter card interrupts to occur, bit 6 must be set.

5.5.1 Programmed Interrupts

Model 617 provides two types of programmed interrupts: PT (Programmed interrupt to Transmitter) and PR (Programmed interrupt to Receiver). Both types of programmed interrupts are used to allow a PCI application to communicate with a VMEbus application. See section 3.3.1 and 3.3.2 for descriptions of PT and PR Interrupts.

5.5.1.1 Sending PT Interrupts

A PT Interrupt can be sent to the VMEbus and does not require the PCI processor to make any remote CSR accesses. To send a PT Interrupt to the VMEbus, the PCI processor must choose a cable interrupt line (CINTx) to carry the PT Interrupt. Any cable interrupt line from 1 - 7 can be selected by setting the appropriate code in the Local Interrupt Control Register's bits 2 - 0.

The cable interrupt line determines which VMEbus interrupt level is asserted when a PT Interrupt occurs.

PT CINT SEL BIT			PT APPEARS ON VMEbus
Bit 2	Bit 1	Bit 0	INTERRUPT LEVEL
0	0	0	PT is not sent to VMEbus
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Cable Interrupt Lines Versus VMEbus Interrupt Levels

➔ The seven cable interrupt lines are shared. Make sure the PCI is not configured to send the same level as the VMEbus sends to the PCI.

➔ If CINT1 or CINT2 is selected to carry the PT Interrupt, the R-INT jumper block must be configured. See section 10.3.7 for information on setting the R-INT jumpers.

After a cable line is selected, a PCI processor only needs to set bit 5 of the Local Command Register to send a PT Interrupt to the VMEbus.

For more information about how the VMEbus determines a PT Interrupt is active and how it acknowledges the interrupt, see section 8.5.1.2.

5.5.1.2 Receiving PT Interrupts

A VMEbus processor can send a PT Interrupt to the PCI system without making any remote CSR accesses. A PCI processor can tell if a PT Interrupt is active by reading the Remote Status Register to see if bit 1 is set. A PCI processor can acknowledge a PT Interrupt by setting bit 6 of Remote Command Register 1.

➔ Before a PT Interrupt can be received, normal interrupts must be enabled by setting bit 6 of the Local Interrupt Control Register.

For information about how the VMEbus sends a PT Interrupt, see section 8.5.1.1.

5.5.1.3 Sending PR Interrupts

A PR Interrupt can be sent to the VMEbus and the VMEbus does not need to use the cable to acknowledge the interrupt. The PCI processor must set bit 5 of Remote Command Register 1 to send a PR Interrupt to the VMEbus.

For information about how the VMEbus determines if a PR Interrupt is active and how it acknowledges the interrupt, see section 8.5.1.4.

5.5.1.4 Receiving PR Interrupts

A VMEbus processor can send a PR Interrupt to the PCI system and a PCI processor does not need to make a remote CSR access to acknowledge the interrupt. A PCI processor can tell if a PR Interrupt is active by reading the Local Status Register to see if bit 5 is set.

A PCI processor can acknowledge a PR Interrupt by setting bit 6 of the Local Command Register.

➔ Before a PR Interrupt can be received, Normal interrupts must be enabled by setting bit 6 of the Local Interrupt Control Register.

5.5.2 Error Interrupts

The PCI adapter card can send an interrupt whenever the card detects an operational error. There are four different status errors that can occur:

- A remote bus error.
- An interface timeout.
- A parity error.
- A DMA LRC error.

The Error Interrupt is enabled by setting bit 5 of the Local Interrupt Control Register. To acknowledge an Error Interrupt, the PCI processor must set bit 7 of the Local Command Register.

A remote bus error occurs when a PCI to VMEbus transfer resulted in a VMEbus error. The most likely cause of this error is an incorrect VMEbus address or an incorrect VMEbus address modifier.

An interface timeout occurs when a PCI to VMEbus transfer did not complete before the PCI timeout of 30 µsec ended. This error generally occurs before a remote bus error because the standard VMEbus timeout is 50 µsec. An interface timeout is usually caused by the same conditions that cause a remote bus error.

A parity error occurs when the parity was incorrect during PCI to VMEbus communications. Parity errors should be rare.

A DMA LRC error occurs when a Longitudinal Redundancy Check (LRC) failed during a DMA transfer. LRC errors should be rare.

➔ For more information about status errors, see section 11.2.2.

➔ Before an error interrupt can occur, error interrupts must be enabled in bit 5 of the Local Interrupt Control Register.

5.5.3 VMEbus Backplane Interrupts

Up to seven VMEbus interrupts (IRQ1 - IRQ7) may be passed across the cable interrupt lines to the PCI adapter card. This allows a PCI application to receive and acknowledge VMEbus backplane interrupts. Because the PCI adapter card has only one interrupt level, all seven VMEbus interrupt levels are routed into a single PCI interrupt. The VMEbus interrupt level is recorded in the Local Interrupt Status Register.

By reading bits 1 - 7 of the Local Interrupt Status Register the PCI application can identify active VMEbus backplane interrupts. Bits that are set correspond to the VMEbus backplane interrupts that are active. To acknowledge a VMEbus interrupt level, that level must be written to Remote Command Register 1 IACK Address bits, and a single byte or word read of the Remote IACK Read Low Register must be done. For example, if bits 7 and 4 were set in the register (0x90 was read), this indicates that two VMEbus interrupt levels were active. First, 7 would be written to IACK Address bits and the remote IACK Read Register would be read. Then, 4 would be written to IACK Address bits and the remote IACK Read Register would be read. Some devices that are RORA require additional processing before they release their interrupt line.

The T-INT jumper block on the VMEbus adapter card determine which VMEbus backplane interrupts are passed to PCI. VMEbus interrupt level X is passed to PCI if jumper X of the T-INT jumper block is installed. See section 10.3.6 for more information about the T-INT jumper block.

➔ The ISR should check for a PT Interrupt before looking for VMEbus backplane interrupts because a PT Interrupt uses a cable interrupt line and also sets a bit in the Local Interrupt Status Register.

➔ The Remote IACK Read Register should only be read when a VMEbus backplane interrupt is pending. Do not read this register twice or when a backplane interrupt is not pending. The Remote IACK Read High Register should never be read as a byte.

➔ Before a backplane interrupt can be received, normal interrupts must be enabled by setting bit 6 of the Local Interrupt Control Register.

➔ Do not enable the same level that is used by PCI to send a PT Interrupt. See also section 5.5.1.1.

5.5.4 DMA Interrupts

If the DMA Done Interrupt is enabled, the PCI adapter card will assert an interrupt when the current DMA operation finishes either successfully or unsuccessfully. The DMA Done Interrupt is enabled by setting bit 2 of the Local DMA Command Register. Bit 2 should be set before setting the Start DMA bit.

To acknowledge a DMA Done Interrupt, clear bit 1 of the Local DMA Command Register.

For more information about DMA Interrupts, see section 5.6.5.1.

➔ For the PCI adapter card, a DMA Done Interrupt occurs only if normal interrupts are enabled *and* DMA Done Interrupts are enabled.

5.5.5 Writing An Interrupt Service Routine

The Interrupt Service Routine (ISR) is responsible for determining if the PCI adapter card is generating an interrupt and for acknowledging an interrupt. A general ISR procedure is shown below.

1. Read the Local Interrupt Control Register to determine if the PCI adapter card is generating an interrupt. If the PCI adapter card is generating an interrupt, go to step 2. If all the interrupt sources on the PCI adapter card have been cleared, complete any platform-specific hardware issues and exit the ISR. If the PCI adapter card was not generating an interrupt, pass control to the next ISR registered at this level.

2. If Error Interrupts are enabled, read the Local Status Register. If the Remote Bus Error bit, Timeout bit, LRC Error bit, or Parity Error bit is set, clear the errors by setting bit 7 in the Local Command Register. If there is an interface timeout, read a remote node I/O register (ignore the results) to flush the interface. Go to step 1.

3. Check for a PR Interrupt by reading the Local Status Register. If the PR Interrupt bit is set, clear it by setting bit 6 in the Local Command Register. Go to step 1.

4. Check for a DMA Done Interrupt by reading the Local DMA Command Register. If the DMA Done bit is set and the DMA Enable bit is set, clear the DMA Done bit by clearing bit 1 of the Local DMA Command Register. Go to step 1.

5. Check for a PT Interrupt by reading the Remote Status Register. If the PT Interrupt bit is set, clear it by setting bit 6 of Remote Command Register 1. Go to step 1.

6. Check for a VMEbus backplane interrupt by reading the Local Interrupt Status Register. Each of the bits set indicate a VMEbus interrupt that must be acknowledged. See section 5.5.3 for more information about acknowledging backplane interrupts.

5.6 Initiating A DMA Operation

The Model 617 adapter supports Direct Memory Accesses (DMA) between the PCI bus and VMEbus. DMA is a high-speed method of transferring data that requires little processor attention. The processor initializes a few registers, starts the DMA operation and checks for status errors after the DMA is done. It does not perform the actual data transfers, therefore, is free to do other tasks that do not involve the adapter.

For large size transfers, DMA transfers move data between the PCI bus and VMEbus approximately ten times faster than PIO.

➔ Neither the PCI system nor the VMEbus system can make any type of remote access while a DMA operation is in progress. Also, interrupts cannot be passed between the PCI system and VMEbus systems during a DMA transfer.

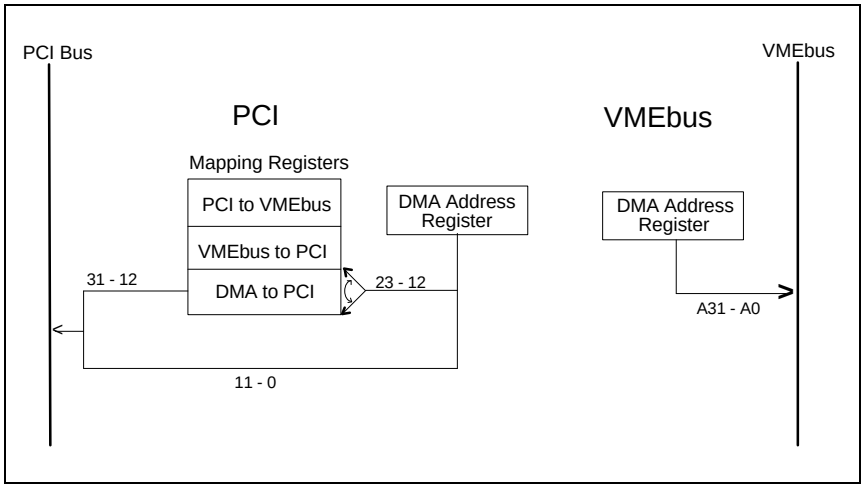
5.6.1 Mapping Register Window

The DMA-to-PCI Bus Mapping Registers, the third segment of the Mapping Register Window, are used to control DMA access to PCI memory space. There are 4,096 DMA-to-PCI Bus Mapping Registers. Each register controls 4K bytes of address space.

OFFSET FROM BASE (hex)	DESCRIPTION	NUMBER OF REGISTERS
0000 0000 - 0000 7FFF	PCI Bus to VMEbus Mapping Registers	8K
0000 8000 - 0000 BFFF	VMEbus to PCI Bus Mapping Registers	4K
0000 C000 - 0000 FFFF	**DMA to PCI Bus Mapping Registers**	**4K**

The PCI and VMEbus physical starting address and the byte count are the main components of each DMA transfer. The VMEbus physical starting address and transfer length are fairly easy to determine. The PCI starting DMA address can be more difficult to determine.

The PCI DMA address has two components: an index to a DMA-to-PCI Mapping Register and a 4K byte offset. The index portion of the PCI DMA address points to a specific DMA-to-PCI Mapping Register. This Mapping Register provides the upper bits of the PCI physical address and information about how to perform byte swapping. The offset section of the PCI DMA address provides the lower bits of the PCI physical address.



Because the DMA transfer uses the DMA-to-PCI Mapping Registers, these registers must be initialized before the DMA is started. The DMA-to-PCI Mapping Registers are programmed to point to a section of PCI memory that will be used in the DMA. The PCI DMA address is then programmed to select the appropriate PCI DMA Mapping Register.

For example, if 2M bytes of PCI memory at physical address 0x1000000 will be used in a DMA operation, assuming no byte swapping bits need to be set, the PCI DMA Mapping Registers should be programmed as follows.

PCI DMA MAPPING REGISTER	OFFSET FROM THE MAPPING BASE ADDRESS	PROGRAM TO
0	0xC000	0x01000000
1	0xC004	0x01001000
2	0xC008	0x01002000
3	0xC00C	0x01003000
• • •	• • •	• • •
511	0xC7FC	0x011FF000

5.6.1.1 DMA-To-PCI Bus Mapping Register Format

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	A24

D23	D22	D21	D20	D19	D18	D17	D16
A23	A22	A21	A20	A19	A18	A17	A16

D15	D14	D13	D12	D11	D10	D9	D8
A15	A14	A13	A12	Reserved	Reserved	Reserved	Reserved

D7	D6	D5	D4	D3	D2	D1	D0
Reserved	Reserved	Reserved	Reserved	Reserved	Word Swap	Byte Swap Non-Byte Data	Invalid

Bit 0 (Map Register Invalid): To enable DMA to PCI bus access, this bit must be reset to "0". If set to "1", bus timeouts occur. This bit is indeterminate following power-up.

Bit 1 (Byte Swap On Non-Byte Data Enable): When set to "1", byte swapping occurs on word and dword (longword) operations.

Bit 2 (Word Swap Enable): When set to "1", address bit A1 is inverted for all transfers.

➔ For more information on byte swapping, see section 3.5.

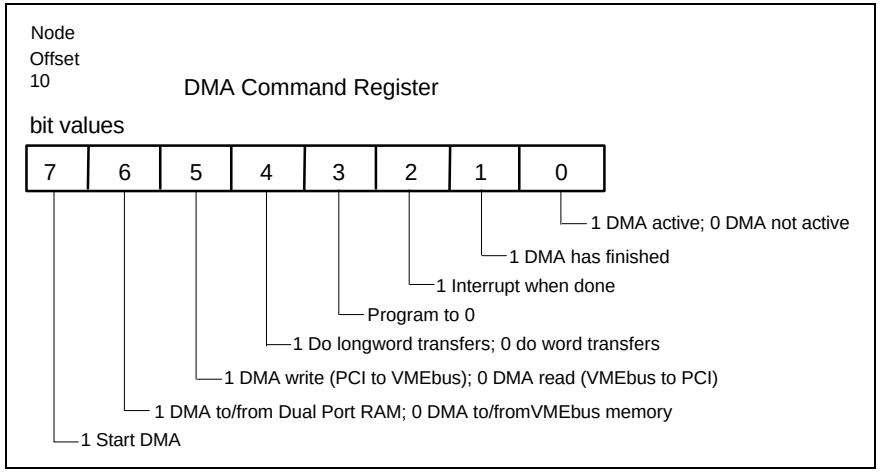
Bits 3 - 11: Reserved.

Bits 12 - 31 (Local Address A12 - A31): The PCI bus address is formed by combining the low 12 bits of the PCI DMA Address Register with the upper 20 bits of this register.

5.6.2 DMA CSRs

For DMA transfers, a number of DMA CSRs need to be programmed:

CSR	SIZE	DESCRIPTION
Local DMA Address	3 bytes	Bits 11 - 0: Address bits A11 - A0 of the PCI starting physical address Bits 23 - 12: Specify which PCI DMA Mapping Register to use
Remote DMA Address	4 bytes	Load with the starting VMEbus address
Local DMA Remainder Count	8 bits	Load with the DMA length in bytes modulo 256 (divide the DMA length by 256 and use the remainder to load this register)
Remote DMA Remainder Count	8 bits	Load with the same value as the Local DMA Remainder Count Register
Local DMA Packet Count	16 bits	The DMA length in bytes divided by 256
Local DMA Command		Controls how the DMA transfer is performed. For example, if it is a read or a write, and/or if it is to Dual Port RAM or Remote RAM, etc. See figure on next page The Local DMA Command Register normally is loaded twice. At the beginning of the DMA register programming, the Local DMA Command Register is loaded with the appropriate bits set, except for the DMA Start bit. Then the other registers are loaded. After all registers are loaded, the DMA Start bit is set



5.6.3 Other CSRs

Several non-DMA registers must also be programmed before a DMA operation can start, including: the Local Interrupt Control Register, Remote Command Register 2, and the Remote Address Modifier Registers.

CSR	DESCRIPTION
Local Interrupt Control	For a DMA Done Interrupt, the Normal Interrupt bit (bit 6) must be set
Remote Command Register 2	The Disable Remote Card Interrupts bit (bit 4) must be set before a DMA operation starts. The Block Mode bit (bit 5) and the Pause Mode bit (bit 7) can be set to control the amount of VMEbus bandwidth used by the DMA
Remote Address Modifier	Must be programmed with address modifier to be presented to the VMEbus during the DMA transfer. The address modifier indicates the address size (A32 or A24) and the type of transfer (Block or Non-Block). The programmed value must correspond to the DMA address and to the Block Mode bit

5.6.4 DMA Transfer Modes

There are three DMA transfer modes: Block Mode, Pause Mode and Non-Block Mode. Remote Command 2 controls which of these modes is used during a DMA.

Block Mode, activated by setting bit 5, has the highest data throughput (about 26M Bytes/sec). In Block Mode, the VMEbus adapter card never transfers more than 256 bytes without rearbitrating for the VMEbus.

In Pause Mode, the DMA Controller rearbitrates for the VMEbus after 64 bytes have been transferred. Pause Mode allows other VMEbus devices to get a bus grant more quickly than when using Block Mode. Pause Mode is activated by setting bits 7 and 5.

Non-Block Mode DMA rearbitrates for the VMEbus after every transfer, allowing minimum latency for other VMEbus masters requesting the bus. This method transfers data less efficiently since it requires a new address cycle for every data cycle. Non-Block Mode DMA occurs when bit 5 is clear.

5.6.5 When Is The DMA Operation Done?

After the DMA Start bit is set, the DMA Done Interrupt and the DMA Done bit can be used to tell if the DMA transfer is done.

5.6.5.1 DMA Done Interrupt

The DMA Done Interrupt can indicate that a DMA transfer is done if the DMA Done Interrupt is enabled and an ISR is installed. The ISR sets a software flag when the PCI adapter card asserts the DMA Done Interrupt. The flag indicates to the application that the DMA operation has ended.

To enable the DMA Done Interrupt, set bit 2 of the Local DMA Command Register and bit 6 of the Local Interrupt Control Register. After the two bits are set, the PCI adapter card will assert its interrupt as soon as the DMA transfer completes. The application must have an ISR installed to service the interrupt when it occurs.

If the following items are true, then the PCI adapter card is generating a DMA Done Interrupt:

- The Interrupt Active bit (bit 7) of the Local Interrupt Control Register is set.
- The Normal Interrupt Enable bit (bit 6) of the Local Interrupt Control Register is set.
- The DMA Interrupt Enable bit (bit 2) of the Local DMA Command Register is set.
- The DMA Done bit (bit 1) of the Local DMA Command Register is set.

The ISR can acknowledge a DMA Done Interrupt by clearing the DMA Done bit (bit 1) of the Local DMA Command Register.

➔ The DMA Done Interrupt occurs whether the DMA completed successfully or not. Therefore, the application must search for status errors by reading the Local Status Register.

5.6.5.2 Polling For DMA Done Bit

The DMA Done bit (bit 1) in the Local DMA Command Register indicates if the DMA operation is complete. When the register is read, if bit 1 is set, the DMA transfer is done.

➔ The DMA Done bit is set whether the DMA completed successfully or unsuccessfully. Therefore, the application must look for status errors by reading the Local Status Register.

➔ Constantly reading the DMA Command Register while a DMA transfer is in progress degrades DMA performance.

5.6.6 Programming Sequence For Initiating A DMA Transfer From PCI

1. Load the Local DMA Command Register. Set all appropriate bits except the Start DMA bit. The Start DMA bit must be clear ("0").

2. Load the Local DMA Address Register's 3 bytes. Bits 23 - 12 are an index to the starting PCI DMA Mapping Register. Bits 11 - 0 are the starting PCI physical address.

3. Load the Remote DMA Address Register's 4 bytes with the starting VMEbus physical address.

4. Load the Local DMA Remainder Count Register. The Remainder Count is the DMA length in bytes modulo 256.

5. Load the Remote DMA Remainder Count Register with the same value as in step 4.

6. Load the Local DMA Packet Count Register. The packet count is the DMA length in bytes divided by 256.

7. For a DMA Done Interrupt, make sure the Local Interrupt Control Register's Normal Interrupt Enable bit is set.

8. Load Remote Command Register 2 to disable VMEbus interrupts. The Pause and Block Mode bits may be set.

9. Load the Remote Address Modifier Register. The address modifier must correspond to the remote DMA address and the Block Mode bit programmed.

10. Read the Local DMA Command Register and set the Start DMA bit. Write this value to the DMA Command Register.

11. The DMA transfer begins.

12. If the DMA Done Interrupt is enabled, wait for the interrupt; or poll the Local DMA Command Register to see if the DMA Done bit is set.

13. The DMA transfer is done.

14. Check for status errors by reading the Local Status Register.

15. Clear the DMA Command Register.

16. Restore Remote Command Register 2, allowing VMEbus interrupts to come through.

5.6.7 Things To Remember

- Make sure the VMEbus address modifier corresponds to the value in Remote Command Register 2's Block Mode bit. Setting the Block Mode bit but giving a Non-Block Mode address modifier causes unpredictable results on the VMEbus.

- Make sure the VMEbus address modifier corresponds to the number of significant address bits programmed into the Remote DMA Address Register. For example, if an A24 address is loaded into the Remote DMA Address Register, the address modifier should be A24.

- Make sure interrupts are disabled via Remote Command Register 2 on the VMEbus adapter card before the DMA transfer is started .

- Never make a cable access while a DMA transfer is in progress. Bit 0 of the DMA Command Register indicates if a DMA operation is in progress.

- If a DMA Done Interrupt is enabled, an interrupt handler must be installed and normal interrupts enabled in the Local Interrupt Control Register.

5.6.8 Example Of Initiating A DMA Operation

In this example, 16K bytes of data from PCI memory are written to the first 16K byte portion of the VMEbus disk controller buffer. Block Mode DMA is used. The values given below are only examples and may not match the values your system will return.

1. Program the DMA-to-PCI Mapping Registers with the physical address of PCI memory.

   - Read the base address of the Mapping Register Window at configuration offset 0x18. Returns 0x82000000.
   - Find the physical address of PCI memory that contains the data to be transferred via DMA. In this example, PCI memory is at 0x800000 - 0x804000.
   - Program the first DMA-to-PCI Mapping Register. Write 0x00800000 to location 0x8200C000. 0x00800000 is the PCI memory physical address and no swapping bits are set. Location 0x8200C000 equals 0x82000000 (base Mapping Register address) + 0xC000 (offset of PCI DMA Mapping Registers).
   - Program the second PCI DMA Mapping Register. Write 0x00801000 to location 0x8200C004.
   - Program the third PCI DMA Mapping Register. Write 0x00802000 to location 0x8200C008.
   - Program the fourth PCI DMA Mapping Register. Write 0x00803000 to location 0x8200C00C.

2. Program the DMA CSRs.

   - Load the Local DMA Command Register with 0x30. Bit 5 is set to write to the VMEbus. Bit 4 is set for longword transfers. Bit 6 is clear to access remote RAM. Bit 2 is clear to disable the DMA Done Interrupt. Bits 7, 3, 1 and 0 should always be clear during this write.
   - Load the Local DMA Address Register with 0. Address bits 23 - 12 are 0 so that DMA-to-PCI Mapping Register 0 is the starting Mapping Register for the DMA transfer. As the Local DMA address increments 16K times, bits 23 - 12 will be equal to 0, 1, 2 and 3. Therefore, the first, second, third and fourth Mapping Registers will be used (these registers were initialized in step 1). Bits 11 - 0 indicate that the lower bits of the starting PCI address are 0.
   - Load the Remote DMA Address Register with 0x12340000. The disk controller's buffer is located at 0x12340000 on the VMEbus.
   - Load the Local DMA Remainder Count Register with 0. 16K bytes modulo 256 equals 0.

- Load the Remote DMA Remainder Count Register with 0. 16K bytes modulo 256 equals 0.
- Load the Local DMA Packet Count Register with 0x40. 16K bytes divided by 256 equals 0x40.

3. Program the other CSRs.
   - Load the Remote Address Modifier Register with 0xF. 0xF is the address modifier for A32 supervisory block transfer.
   - Load Remote Command Register 2 with 0x30. Bit 5 is set for Block Mode. Bit 4 must be set to disable VMEbus interrupt passing. Bit 7 is clear to disable Pause Mode. Bit 6 must always be clear.
4. Start the DMA transfer. Load the Local DMA Command Register with 0xB0. 0xB0 is the previous value with the Start DMA bit (bit 7) set.
5. Wait for the DMA transfer to complete. You can use the processor for other tasks that do not use the adapter.
6. Read the Local DMA Command Register. If the DMA Done bit (bit 1) is clear, go to step 5. If the DMA Done bit is set, the DMA transfer is done.
7. Check for status errors by reading the Local Status Register. If bits 7, 6, 2, 1 or 0 are set, the DMA transfer did not complete successfully. See section 11.2.2 for more information about status errors. If these bits are not set, the DMA completed correctly.
8. Write 0x0 to Remote Command Register 2 to re-enable interrupt passing from the VMEbus. Any pending VMEbus interrupts are now passed to PCI if configured to do so.

5.7 Configuration Registers

The PCI specification requires several Configuration Registers for each PCI card. Only a few of the Configuration Registers are necessary to program the Model 617 adapter. The remaining Configuration Registers, identified as *Reserved* in this manual, are not used or do not provide the adapter user with any useful information. For information about the reserved Configuration Registers, refer to the PCI specification.

The PCI adapter card conforms to the Configuration Register space as defined in Chapter 6, revision 2.0 of the PCI specification.

Device ID (0x02)		Vendor ID (0x00)	
Status (0x06)		Command (0x04)	
Class Code (0x09)			Revision ID (0x08)
Reserved	Reserved	Reserved	Reserved
I/O Mapped Node I/O Base Address Register (0x10)			
Memory Mapped Node I/O Base Address Register (0x014)			
Mapping Register Base Address Register (0x18)			
Remote Memory Base Address Register (0x1C)			
Reserved			
Reserved			
Reserved			
Reserved			
Reserved			
Reserved			
Reserved			
Reserved			
Reserved			
Reserved	Reserved	Reserved	Interrupt Line (0x3C)

5.7.1 Finding And Identifying The PCI Adapter Card

Four Configuration Registers are used by the PCI system to find and identify the installed PCI adapter card: Vendor ID Register, Device ID Register; Revision ID Register, and Class Code Register.

5.7.1.1 Vendor ID Register

The Vendor ID Register is a 16-bit, read-only register at address 0x00. This register identifies the manufacturer of the PCI card. SBS Bit 3's PCI SIG assigned Vendor ID is 0x108A. This register is used to find and identify the adapter within PCI bus space.

5.7.1.2 Device ID Register

The Device ID Register is a 16-bit, read-only register at address 0x02 that further identifies each PCI card. The Model 617 adapter Device ID is 0x01. This register is used to find and identify the adapter within PCI bus space.

5.7.1.3 Revision ID Register

The Revisions ID Register is an 8-bit, read-only register at address 0x08. This register specifies the PCI card assembly revision identifier and should be viewed as an extension to the Device ID. The PCI adapter card generates the ASCII value 0x41 for the manufacturing release revision A. The value increments by one for each subsequent manufacturing release.

5.7.1.4 Class Code Register

The Class Code Register is a 24-bit, read-only register at address 0x09 that is used to identify the generic function of the PCI adapter card. The register is divided into three byte-wide fields. The upper byte (address = 0x0B) is a base class code that broadly classifies the type of function the PCI adapter card performs. The PCI adapter card returns a value of 0x06 that is defined as "bridge device". The middle byte (address = 0x0A) is a sub-class code that more specifically identifies the function performed. The PCI adapter card returns a value of 0x80 that is defined as "other bridge device". The lower byte (address = 0x09) specifies a register level programming interface but is not supported by the PCI adapter card. Reading this register returns a value of 0x00.

This register may be used to find and identify the adapter within PCI bus space.

DESCRIPTION	OFFSET	VALUE
Base Class	0x0B	0x06 Bridge
Sub Class	0x0A	0x80 Other Bridge Device
Interface Specification	0x09	0x00 Not Supported

5.7.2 Where Are The Windows?

Four Configuration Registers are used to locate the adapter address windows in PCI bus space: I/O Mapped Node I/O Base Address Register, Memory Mapped Node I/O Base Address Register, Mapping Register Base Address Register, and Remote Memory Base Address Register.

5.7.2.1 I/O Mapped Node I/O Base Address Register

This 32-bit register is located at address 0x10. It indicates the location in the 64K byte PCI I/O space that the 32 bytes of Node I/O Registers were configured at.

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	A24

D23	D22	D21	D20	D19	D18	D17	D16
A23	A22	A21	A20	A19	A18	A17	A16

D15	D14	D13	D12	D11	D10	D9	D8
A15	A14	A13	A12	A11	A10	A9	A8

D7	D6	D5	D4	D3	D2	D1	D0
A7	A6	A5	0	0	0	0	reserved 1

Bits 5 - 15: Give the starting I/O base address of the Node I/O Registers.

➔ Always mask to zero the lower four bits when reading this register.

➔ Do not write this register.

➔ The Node I/O Register can also be accessed through a memory mapped address.

5.7.2.2 Memory Mapped Node I/O Base Address Register

This 32-bit register is located at address 0x14. It indicates where in the 4G byte PCI memory space the 32 bytes of Node I/O Registers appear. This register is necessary in systems that only have memory mapped I/O.

➔ The PCI adapter card reserves 64K bytes for node I/O even though it only uses the first 32 bytes.

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	A24

D23	D22	D21	D20	D19	D18	D17	D16
A23	A22	A21	A20	A19	A18	A17	A16

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	0	0	0

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	reserved 0	reserved 0	reserved 0	reserved 0

Bits 16 - 31: Gives the starting PCI physical bus address of the Node I/O Registers.

➔ Always mask to zero the lower four bits when reading this register.

➔ Do not write this register.

➔ The Node I/O Register can also be accessed through PCI I/O space.

5.7.2.3 Mapping Register Base Address Register

This 32-bit register is located at address 0x18. It indicates where in the 4G byte PCI memory space the 64K bytes of Mapping Registers are located.

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	A24

D23	D22	D21	D20	D19	D18	D17	D16
A23	A22	A21	A20	A19	A18	A17	A16

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	0	0	0

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	reserved 0	reserved 0	reserved 0	reserved 0

Bits 16 - 31: Provides the starting PCI physical bus address of the Mapping Registers.

➔ Always mask to zero the lower four bits when reading this register.

➔ Do not write this register.

5.7.2.4 Remote Memory Base Address Register

This 32-bit register is located at address 0x1C. It indicates where in the 4G byte PCI memory space the 32M bytes of remote memory is located.

D31	D30	D29	D28	D27	D26	D25	D24
A31	A30	A29	A28	A27	A26	A25	0

D23	D22	D21	D20	D19	D18	D17	D16
0	0	0	0	0	0	0	0

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	0	0	0

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	reserved 0	reserved 0	reserved 0	reserved 0

Bits 25 - 31 (A25 - A31): These bits provide the starting PCI physical address of the Remote Memory Window.

➔ Always mask to zero the lower four bits when reading this register.

➔ Do not write to this register.

5.7.3 Other Registers

Two other Configuration Registers provide useful information about the adapter: the Status Register and the Interrupt Line Register.

5.7.3.1 Command Register

The Command Register is a 16-bit, read/write register at address 0x04. This register provides control of the PCI adapter card's ability to generate and respond to PCI cycles. When 0x0000 is written to this register, the PCI card is logically disconnected from the PCI bus for all accesses except Configuration Register access.

D15	D14	D13	D12	D11	D10	D9	D8
15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Fast Back-to-Back	SERR

D7	D6	D5	D4	D3	D2	D1	D0
7	6	5	4	3	2	1	0
Wait Cycle Control	Parity Error Response	VGA Palette Snoop	Memory Write & Invalidate	Special Cycles	Bus Master	Memory Space	I/O Space

Bit 0 (I/O Space Enable): This bit controls the PCI adapter card's response to I/O mapped Node I/O access. When this bit is set to "1", the PCI adapter card responds to I/O mapped Node I/O accesses. When set to "0", the adapter card's response is disabled.

Bit 1 (Memory Space Enable): This bit controls the PCI adapter card's response to memory mapped Node I/O, remote RAM accesses or remote bus I/O accesses. When this bit is set to "1", the PCI adapter card responds to remote memory accesses and to memory mapped Node I/O accesses. When this bit is set to "0", the PCI adapter card's response is disabled.

Bit 2 (Bus Master Enable): This bit controls the PCI adapter card's ability to act as a master on the PCI bus. When this bit is set to "1", the PCI adapter can function as a bus master. When this bit is reset to "0", the PCI adapter card is prohibited from generating PCI accesses.

Bit 3 (Special Cycle Enable): The SBS Bit 3 adapter does not support this bit.

Bit 4 (Memory Write And Invalidate Enable): The SBS Bit 3 adapter does not support this bit.

Bit 5 (VGA Palette Snoop Enable): The SBS Bit 3 adapter does not support this bit.

Bit 6 (Parity Error Response Enable): This bit controls the PCI adapter card's response to parity errors. When this bit is set to "1", the PCI adapter card will activate the PERR# signal when a parity error is detected. When this bit is reset to "0", the PCI adapter card will not activate the PERR# signal. This bit will be reset to "0" if the PCI interface signal RST# is activated.

Bit 7 (Wait Cycle Control Enable): The SBS Bit 3 adapter does not support this bit.

Bit 8 (SERR# Enable): The SBS Bit 3 adapter does not support this bit.

Bit 9 (Fast Back-To-Back Enable): The SBS Bit 3 adapter does not support this bit.

Bits 10 - 15: Reserved.

5.7.3.2 Status Register

The Status Register is a 16-bit, read/write register, at address 0x06, that is used to record status information for PCI bus related events. Writes to this register can reset bits but not set them. A bit is reset whenever the register is written, and the data in the corresponding bit location are "1".

D15	D14	D13	D12	D11	D10	D9	D8
Detected Parity Error	Signaled System Error	Received Master Abort	Received Target Abort	Signaled Target Abort	DEVSEL Timing 1	DEVSEL Timing 0	Data Parity Detected

D7	D6	D5	D4	D3	D2	D1	D0
Fast Back-to-Back	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Bits 0 - 6: Reserved.

Bit 7 (Fast Back-To-Back Capable): When the PCI adapter card is a bus slave, it is not capable of accepting fast back-to-back transactions, if the transactions are not to the same agent. The PCI adapter card will set this bit to "0".

Bit 8 (Data Parity Detected): This bit is significant when the PCI adapter card is a bus master. It is set to "1" when the following three conditions are met: the PCI adapter card asserted PERR# itself or observed PERR# asserted; the PCI adapter was a bus master when the error occurred; the Parity Error Response bit in the Command Register is set.

Bits 9 & 10 (DEVSEL Timing): Bits 9 and 10 encode the timing of DEVSEL#. The PCI bus defines the encoding of these bits. Both bits are read-only and indicate the slowest time that the PCI adapter card asserts DEVSEL# for any bus command except Configuration Read and Configuration Write. The PCI adapter card returns a value of 10B for slow.

Bit 11 (Signaled Target Abort): This bit will be set to "1" by the PCI adapter card whenever it is a bus slave and terminates a transaction with a Target Abort cycle.

Bit 12 (Received Target Abort): This bit is set to "1" by the PCI adapter card when it is a bus master and its transaction is terminated with a Target Abort cycle.

Bit 13 (Received Master Abort): This bit is set to "1" by the PCI adapter card when it is a bus master and its transaction is terminated with a master abort.

Bit 14 (Signaled System Error): The adapter does not support this bit.

Bit 15 (Detected Parity Error): The PCI adapter card will set this bit to "1" whenever it detects a parity error, independent of the state of Command Register bit 6 (see section 4.4).

5.7.3.3 Interrupt Line Register

This 8-bit, read-only register located at address 0x3C is used to communicate interrupt line routing information. Power-up-self-test (POST) software writes the routing information into the Interrupt Line Register as it initializes and configures the system. The input of the system interrupt controller to which the PCI adapter card is connected is indicated by the value in this register. Device drivers and Operating Systems can use this information to determine priority and vector information. Values in the Interrupt Line Register are system architecture specific.

D7	D6	D5	D4	D3	D2	D1	D0
Int 7	Int 6	Int 5	Int 4	Int 3	Int 2	Int 1	Int 0

➔ Do not write to this register.

Chapter 6: CSR Accessed From The PCI Bus

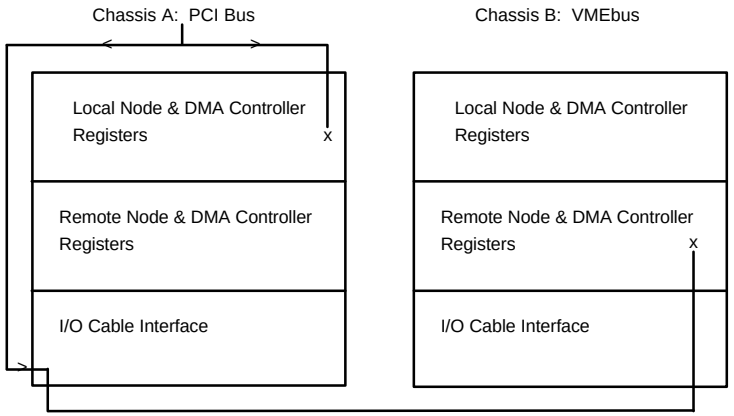
6.0 CSR Accessed From The PCI Bus

Chapter 6 describes PCI adapter card Control and Status Registers (CSR). These registers are accessed through either the I/O Mapped Node I/O Register Window or the Memory Mapped Node I/O Register Window as bytes or words but not longwords.

There are 32 bytes of Node I/O register, 16 of which are located on the PCI adapter card and 16 on the VMEbus adapter card.

The first eight bytes of the PCI adapter card I/O space are for a PCI processor to control and check status of the local adapter card in the PCI chassis -- the adapter Local Node Registers. The following eight bytes of the adapter I/O space are for a PCI processor to talk to the remote (VMEbus) adapter card registers -- the adapter Remote Node Registers.

The final 16 bytes of I/O space are for the PCI processor to talk to the DMA Controller Registers. Eight bytes comprise the Local (PCI) DMA Controller Registers and eight bytes are for the Remote (VMEbus) DMA Controller Registers.



6.1 Local Node Registers

PCI Local Node Registers are located on the PCI adapter card and are addressed by PCI processors.

PCI I/O ADDRESS (hex)	WRITE FUNCTION	READ FUNCTION
I/O Base + 00	Local Command	Local Command
I/O Base + 01	Interrupt Control	Interrupt Control
I/O Base + 02	-- reserved --	Local Status
I/O Base + 03	-- reserved --	Interrupt Status
I/O Base + 04	PCI Control	PCI Control
I/O Base + 05	-- reserved --	-- reserved --
I/O Base + 06	-- reserved --	-- reserved --
I/O Base + 07	-- reserved --	-- reserved --

6.1.1 Local Command Register

The Local Command Register is an 8-bit, read/write register located on the PCI adapter card (address = I/O Base + 0x00).

BIT	FUNCTION
7	Clear Status Register (write only)
6	Clear PR Interrupt (write only)
5	Send PT Interrupt
4	reserved -- program to "0"
3	reserved -- program to "0"
2	reserved -- program to "0"
1	reserved -- program to "0"
0	reserved -- program to "0"

CLEAR STATUS REGISTER (bit 7): Communication between the two systems is monitored for cable parity errors, VMEbus errors, DMA LRC errors, and interface timeouts. These errors are recorded in the Local Status Register. When bit 7 is set to "1", the Status Error bits in the Local Status Register are cleared.

CLEAR PR INTERRUPT (bit 6): If a VMEbus processor sends a PR Interrupt to the PCI system, setting this bit clears it. Writing this bit as a "1" also clears the PR Interrupt bit in the Local Status Register.

SEND PT INTERRUPT (bit 5): When this bit is set to "1", a PT Interrupt is transmitted to the VMEbus system via the cable interrupt lines. When this bit is reset to "0", the PT Interrupt request is removed.

6.1.2 Interrupt Control Register

The Interrupt Control Register is an 8-bit, read/write register located on the PCI adapter card (address = I/O Base + 0x01). Register bits are defined as follows:

BIT	FUNCTION
7	Interrupt Active (read only)
6	Normal Interrupt Enable
5	Error Interrupt Enable
4	reserved
3	reserved
2	PT CINT SEL2
1	PT CINT SEL1
0	PT CINT SEL0

INTERRUPT ACTIVE (bit 7): This bit is set when the PCI adapter card is generating an interrupt. Bit 7 is cleared when the source of the interrupt has been cleared.

NORMAL INTERRUPT ENABLE (bit 6): When this bit is set to "1", interrupts resulting from PR or PT Interrupts, cable interrupts, or DMA Done Interrupts are enabled. When "0", these interrupt sources will not cause a PCI interrupt. Use bit 7 to determine if the PCI adapter card is currently generating an interrupt.

ERROR INTERRUPT ENABLE (bit 5): When this bit is set to "1", interrupts resulting from parity errors, remote bus errors, DMA LRC errors, and interface timeouts are enabled. When "0", Error Interrupts will not cause a PCI interrupt. Use bit 7 to determine if the PCI adapter card is currently generating an interrupt.

PT CINT SEL (bits 2 - 0): These bits are used to map the outgoing PT programmed interrupt to one of seven cable interrupts. PT CINT SEL bits 2 - 0 can be used to select a VMEbus cable interrupt based on the following table:

PT CINT SEL BIT			VMEbus CINT SELECTED
Bit 2	Bit 1	Bit 0	
0	0	0	PT Disabled
0	0	1	CINT1
0	1	0	CINT2
0	1	1	CINT3
1	0	0	CINT4
1	0	1	CINT5
1	1	0	CINT6
1	1	1	CINT7

➔ A cable interrupt should be used to transmit only one interrupt source.

6.1.3 Local Status Register

The Local Status Register is an 8-bit, read-only register located on the PCI adapter card (address = I/O Base + 0x02).

BIT	FUNCTION
7	Interface Parity Error
6	Remote Bus Error
5	Receiving PR Interrupt
4	reserved
3	reserved
2	Interface Timeout
1	LRC Error
0	Remote Bus Power Off or I/O Cable Is Off

INTERFACE PARITY ERROR (bit 7): If a cable interface parity error occurs on a chassis to chassis transfer, bit 7 is set to "1". It is reset to "0" when the Clear Status Register bit is set to "1" in the Local Command Register.

REMOTE BUS ERROR (bit 6): If a VMEbus bus error (BERR signal) occurs on a chassis to chassis transfer, bit 6 is set to "1". It is reset to "0" when the Clear Status Register bit is set to "1" in the Local Command Register.

RECEIVING PR INTERRUPT (bit 5): Set to "1" if a PR Interrupt is received from the VMEbus adapter card. This bit is reset to "0" when the Clear PR Interrupt bit is set to "1" in the Local Command Register.

INTERFACE TIMEOUT (bit 2): This bit is set to "1" if the PCI adapter card has waited 30 µsec for a response to a command issued to the VMEbus adapter card. If the operation does not complete within the specified time interval, this bit is set to "1" and operation terminated. This bit is reset to "0" when the Clear Status Register bit is set to "1" in the Local Command Register. Any time an interface timeout occurs, we recommend that a read of a remote node I/O register be done and the data from the read ignored. This flushes the interface.

LRC ERROR (bit 1): Set to "1" if the Longitudinal Redundancy Check circuitry detects an error on a DMA transfer. The bit is reset to "0" when the Clear Status Register bit is set to "1" in the Local Command Register.

REMOTE BUS POWER OFF or I/O CABLE IS OFF (bit 0): Set to "1" if the VMEbus chassis power is off or if the I/O cable is not connected. It also is "1" when SYSRESET is active on the VMEbus. Attempts to communicate with remote resources will fail and result in interface errors. No Error Interrupt is generated when this bit is set. Bit 0 automatically resets to "0" when the source of the error is resolved.

6.1.4 Interrupt Status Register

The Interrupt Status Register is an 8-bit, read-only register (address = I/O Base + 0x03). This register is located on the PCI adapter card and is addressed by PCI processors.

BIT	FUNCTION
7	Cable Interrupt Pending - CINT7
6	Cable Interrupt Pending - CINT6
5	Cable Interrupt Pending - CINT5
4	Cable Interrupt Pending - CINT4
3	Cable Interrupt Pending - CINT3
2	Cable Interrupt Pending - CINT2
1	Cable Interrupt Pending - CINT1
0	reserved

CABLE INTERRUPT PENDING (bits 7 - 1): If one or more of these bits is set to "1", the cable interrupt corresponding to that bit is pending. Except for the one interrupt that corresponds to the PT Interrupt from the VMEbus adapter card, normally, the seven cable interrupts correspond to the seven VMEbus interrupts .

6.1.5 PCI Command Register

The PCI Command Register is an 8-bit, read/write register (address = I/O Base + 0x04). This register is located on the PCI adapter card and is addressed by PCI processors.

BIT	FUNCTION
7	reserved
6	reserved
5	reserved
4	reserved
3	reserved
2	reserved
1	reserved
0	Target Abort

TARGET ABORT (bits 0): When "1" is written to this bit, the PCI adapter card generates a Target Abort Cycle on the PCI bus when it detects an interface timeout. When "0" is written to this bit, the PCI adapter card generates a Target Disconnect when it detects an interface timeout. Some personal computers hang if this bit is set and an interface timeout occurs.

6.2 Remote Node Registers

Eight adapter Remote Node Registers are controlled by processors on the PCI chassis, but are located on the *remote* (VMEbus) adapter card.

PCI I/O ADDRESS (hex)	WRITE FUNCTION	READ FUNCTION
I/O Base + 08	Remote Command Register 1	Remote Status Register
I/O Base + 09	Remote Command Register 2	Remote Command Register 2
I/O Base + 0A	-- reserved --	-- reserved --
I/O Base + 0B	-- reserved --	-- reserved --
I/O Base + 0C	Adapter ID	Adapter ID
I/O Base + 0D	Remote VMEbus Address Modifier	Remote VMEbus Address Modifier
I/O Base + 0E	-- reserved --	Remote IACK Read LOW
I/O Base + 0F	-- reserved --	Remote IACK Read HIGH

6.2.1 Remote Command Register 1

Remote Command Register 1 is a write-only register located on the VMEbus adapter card (address = I/O Base + 0x08).

BIT	FUNCTION
7	Reset VMEbus Adapter Card (one-shot, allow 1 sec)
6	Clear PT Interrupt
5	Send PR Interrupt
4	Lock VMEbus
3	reserved -- should always be programmed to "0" --
2	IACK Address Bit 2
1	IACK Address Bit 1
0	IACK Address Bit 0

➔ Take care when reading from the Remote Status Register and writing to Remote Command Register 1 because the bits are not in the same positions for reads and writes.

RESET VMEbus ADAPTER CARD (bit 7): The VMEbus adapter card has a power-on reset circuit that resets the card when power is applied to the VMEbus chassis. This reset may also be activated from the PCI chassis by writing a "1" to bit 7 of the Remote Command Register. After triggering the reset, your program should wait one second before starting another remote access or VMEbus cycle.

Writing a "0" to this bit clears the Was Reset flag in the Remote Status Register.

If the SYSRESET jumper in the SYS jumper block is installed, this reset also drives the VMEbus global reset signal. See section 10.3.1 for more information about the SYS jumper block.

CLEAR PT INTERRUPT (bit 6): When the VMEbus adapter card is sending a PT Interrupt to the PCI bus, this bit is used to clear this interrupt. Setting this bit will cause the PT Interrupt to be cleared.

SEND PR INTERRUPT (bit 5): A PCI processor sends a PR Interrupt to the VMEbus chassis by writing a "1" to this bit. Writing a "0" has no effect.

LOCK VMEbus (bit 4): Writing a "1" to this bit sets the Lock Bus bit. If the Lock Bus bit is set, the address strobe signal on the VMEbus remains active after the first VMEbus access preventing any other VMEbus master from using the bus and permitting the PCI bus to convert a read operation followed by a write operation into an atomic read-modify-write on the VMEbus.

To use the Lock Bus function, the PCI system user should set the Lock Bus bit and perform a read followed by a write (to the same address). Then, quickly clear the Lock Bus bit. We recommend that interrupts be disabled during this operation.

The Lock Bus function is useful in multi-processor applications in which processors often signal availability of a resource through an indivisible read-modify-write semaphore operation. The Lock Bus bit may also be used by the PCI bus to make accesses to Dual Port RAM indivisible.

Read-modify-write operations (such as when the LOCK prefix is used) are automatically indivisible.

IACK ADDRESS BITS (bits 2-0): The IACK address bits determine which VMEbus interrupt level is acknowledged when the IACK Read Register is read. These three bits must be set to the desired VMEbus interrupt level before the IACK Read Register is read. See section 6.3.6 for more information on IACK Read.

6.2.2 Remote Status Register

The Remote Status Register is a read-only register located on the VMEbus adapter card (address = I/O Base + 0x08).

BIT	FUNCTION
7	VMEbus Was Reset
6	IACK Address Bit 1
5	PR Was Sent
4	Lock Bus Not Set (*inverted* state of the Lock Bus flip-flop)
3	reserved -- must always be programmed to "0" --
2	IACK Address Bit 2
1	Receiving PT Interrupt
0	IACK Address Bit 0

➔ Take care when reading from the Remote Status Register and writing to Remote Command Register 1 because the bits are not in the same positions for reads and writes.

VMEbus WAS RESET (bit 7): Set to "1" when SYSRESET occurs on the VMEbus or the VMEbus adapter card is reset using bit 7 of the Remote Command Register 1. This bit is cleared when a "0" is written to bit 7 of the Remote Command Register 1.

Whenever the VMEbus adapter card is reset, the adapter should be re-initialized.

IACK ADDRESS BIT 1 (bit 6): Shows the state of the IACK Address Bit 1 written to the Remote Command Register 1. This bit and bits 2 and 0 are non-contiguous to maintain compatibility with previous SBS Bit 3 adapter models.

PR INTERRUPT WAS SENT (bit 5): Bit 5 is a "1" when the PCI adapter card is sending a PR Interrupt to the VMEbus.

LOCK BUS NOT SET (bit 4): Shows the *inverted* state of the Lock Bus flip-flop controlled by bit 4 of the Remote Command Register 1.

IACK ADDRESS BIT 2 (bit 2): Shows the state of the IACK Address Bit 2 written to Remote Command Register 1.

RECEIVING PT INTERRUPT (bit 1): This bit is a "1" when the PCI adapter card is receiving a PT Interrupt from the VMEbus system.

IACK ADDRESS BIT 0 (bit 0): Shows the state of the IACK Address Bit 0 written to Remote Command Register 1.

6.2.3 Remote Command Register 2

Remote Command Register 2 is located on the VMEbus adapter card (address = I/O Base + 0x09).

BIT	FUNCTION
7	DMA Controller Pause on 16 Transfers
6	reserved -- should always be programmed to "0" --
5	VMEbus Block Mode DMA Controller Operation
4	Disable Remote Adapter Card Interrupt Passing
3	Program to "0"
2	Program to "0"
1	Program to "0"
0	Program to "0"

DMA CONTROLLER PAUSE ON 16 TRANSFERS (bit 7): Setting this bit during a DMA Controller Block Mode operation causes the VMEbus adapter card DMA Controller to never transfer more than 64 bytes without rearbitrating for the VMEbus. This pause allows other VMEbus masters to receive a bus grant quickly if a DMA operation is in progress. The Pause Mode bit has effect only if the Block Mode bit (bit 5) is also set.

VMEbus BLOCK MODE DMA CONTROLLER OPERATION (bit 5): Used during DMA Controller transfers to select VMEbus adapter card Block Mode operation. During Block Mode, the VMEbus DMA Controller never transfers more than 256 bytes before it rearbitrates for the VMEbus. Block Mode is the fastest DMA mode, but it may not allow other VMEbus cards enough access to the bus. The Pause Mode bit (bit 7) can be set so that the DMA Controller re-arbitrates for the VMEbus more frequently.

DISABLE REMOTE ADAPTER CARD INTERRUPT PASSING (bit 4): Writing a "1" to this bit prevents VMEbus adapter card interrupts from coming across the cable to the PCI adapter card. This bit must be set before starting a DMA transfer from the PCI system and should be cleared when the DMA finishes.

6.2.4 Adapter ID Register

A byte read of the adapter ID Register (address = I/O Base + 0x0C) returns the hex value 80 that identifies the card on the other end of the cable as a VMEbus card. A write to this register has no effect.

6.2.5 Remote VMEbus Address Modifier Register

The Address Modifier Register is a read/write register used only during adapter DMA Controller operations to present an address modifier to the VMEbus. The Address Modifier Register is located on the VMEbus adapter card (address = I/O LO + 0x0D).

The register is loaded, before starting the DMA Controller operation, with the address modifier appropriate to the VMEbus memory -- signaling the proper address width and Block/Non-Block transfer mode.

Remote Command Register 2 bit 5 (Block Mode DMA) must be clear if the address modifier is a Non-Block Mode transfer.

6.2.6 Remote IACK Read Registers

A PCI processor can instruct the adapter to perform an interrupt acknowledge cycle on the VMEbus by reading from the IACK Read Registers. The adapter converts a read from these registers (in the PCI chassis) into a remote interrupt acknowledge cycle (on the VMEbus), activating the VMEbus IACK line and presenting a 3-bit IACK code corresponding to the interrupt level acknowledged.

The IACK Read LOW Register is a read-only register located on the VMEbus adapter card (address = I/O Base + 0x0E). The IACK Read HIGH Register is a read-only register located on the VMEbus adapter card (address = I/O Base + 0x0F).

The 3-bit IACK code presented by the VMEbus adapter card is set by writing to Remote Command Register 1 bits 2-0. See also section 6.2.1.

The IACK Read LOW Register can be read as a byte or both IACK Registers can be read as a word.

➔ Never read the IACK Read HIGH Register as a byte.

➔ Two IACK Reads cause *two* IACKs to occur; the second read can cause a VMEbus bus error.

➔ For information about using the IACK Read Registers see sections 5.5.3 and 11.1.3.

6.3 DMA Controller Registers

This section covers the DMA Controller Registers accessed from the PCI bus. Refer to sections 3.3.3 - 3.3.4 for a general description of DMA.

➔ For information about programming a DMA, refer to section 5.6.

6.3.1 DMA Controller and Error Status Registers Accessed From The PCI Bus

The registers listed in the following table are located on the local (PCI) adapter card and are addressed by a PCI processor to initialize a DMA Controller operation.

PCI I/O ADDRESS (hex)	WRITE FUNCTION	READ FUNCTION
I/O Base + 10	DMA Command	DMA Command
I/O Base + 11	DMA Remainder Count	DMA Remainder Count
I/O Base + 12	DMA Packet Count 0-7	DMA Packet Count 0-7
I/O Base + 13	DMA Packet Count 8-15	DMA Packet Count 8-15
I/O Base + 14	DMA PCI Address 2-7	DMA PCI Address 2-7
I/O Base + 15	DMA PCI Address 8-15	DMA PCI Address 8-15
I/O Base + 16	DMA PCI Address 16-23	DMA PCI Address 16-23
I/O Base + 17	-- reserved --	-- reserved --

The registers in the table below are located on the remote (VMEbus) adapter card and are addressed by a PCI processor to initiate a DMA Controller operation.

PCI I/O ADDRESS (hex)	WRITE FUNCTION	READ FUNCTION
I/O Base + 18	DMA Remainder Count	DMA Remainder Count
I/O Base + 19	-- reserved --	-- reserved --
I/O Base + 1A	DMA VMEbus Address 16-23	DMA VMEbus Address 16-23
I/O Base + 1B	DMA VMEbus Address 24-31	DMA VMEbus Address 24-31
I/O Base + 1C	DMA VMEbus Address 0-7	DMA VMEbus Address 0-7
I/O Base + 1D	DMA VMEbus Address 8-15	DMA VMEbus Address 8-15
I/O Base + 1E	Slave Status	Slave Status
I/O Base + 1F	-- reserved --	-- reserved --

6.3.2 Local DMA Controller Command Register

The Local DMA Controller Command Register is an 8-bit, read/write register located on the PCI adapter card (address = I/O Base + 0x10).

BIT	FUNCTION
7	Start DMA
6	DMA DP
5	DMA Transfer Direction
4	DMA Word/Longword Select
3	reserved -- program to "0"
2	Enable DMA Done Interrupt
1	DMA Done Flag
0	DMA Active

START DMA (bit 7): When this bit is set to "1", a Controller Mode DMA transfer is initiated. This bit is reset to "0" when the DMA transfer has completed. DMA transfers that exceed 16 msec are aborted and the interface timeout status error is set. Set this bit only after all other command bits and registers have been set up.

DMA DP (bit 6): When this bit is set to "1", the DMA controller routes data to Dual Port RAM. When this bit is reset to "0", the DMA controller routes data to VMEbus RAM.

DMA TRANSFER DIRECTION (bit 5): Writing a "1" to bit 5 causes the DMA Controller to do a write, transferring data from PCI bus to VMEbus. Writing a "0" causes a DMA read, transferring data from VMEbus to PCI bus.

DMA WORD/LONGWORD SELECT (bit 4): Writing a "1" to this bit causes the DMA controller to perform longword (4 byte) transfers. Writing a "0" to this bit causes the DMA controller to perform word (2 byte) transfers.

ENABLE DMA DONE INTERRUPT (bit 2): Writing a "1" to bit 2 activates the DMA Done Interrupt on the PCI adapter card at the completion of a DMA operation. The Normal Interrupt Enable bit in the Local Interrupt Command Register must also be set for the DMA Done Interrupt to occur.

DMA DONE FLAG (bit 1): Bit 1 presents the status of a DMA operation to software in the same manner as the DMA Done Interrupt does for the hardware. It clears itself when a new DMA operation begins. Writing a "0" to bit 1 also clears the DMA Done status and clears the DMA Done Interrupt.

DMA ACTIVE (bit 0): This bit is set to "1" when a DMA transfer is currently active for either local or remote initiated operations.

6.3.3 Local DMA Remainder Count Register

The Local DMA Remainder Count Register is an 8-bit read/write register located on the PCI adapter card (address = I/O Base + 0x11).

Before starting a DMA transfer, the Local DMA Remainder Count Register is loaded with the lowest eight bits of the number of bytes to be transferred by the DMA Controller. The remainder count is the byte count modulo 256. The values in this register must be multiples of 4 bytes for longword DMA transfers and multiples of 2 bytes for word DMA transfers.

The same value must also be loaded in the Remote DMA Remainder Count Register.

6.3.4 Local DMA Packet Count Register

The Local DMA Packet Count Register is a 16-bit read/write register located on the PCI adapter card (address = I/O Base + 0x12). It is loaded with the upper two significant bytes of the number of bytes to be transferred during a DMA operation. The packet count is the DMA byte count divided by 256.

As the DMA controller operation progresses, the contents of the Packet Count Register decrements. A read of the register during a DMA returns the number of 256 byte packets that still need to be transferred.

6.3.5 Local DMA PCI Address Register

This 24-bit, read/write register (address = I/O Base + 0x14) is used to specify the PCI DMA starting address. The DMA starting address specifies both the DMA-to-PCI Mapping Register to use and the lower bits of the PCI physical address. For longword transfers, the PCI address must be a multiple of four. For word transfers, the PCI address must be a multiple of two.

PCI DMA ADDRESS BITS (bits 1 - 23): Bits 1 - 11 directly access physical addresses. Bits 12 - 23 index a DMA-to-PCI Mapping Register that holds physical addresses.

As the DMA progresses, the Local DMA Address Register increments by two (for word DMAs) or four (for longword DMAs).

6.3.6 Remote DMA Controller Remainder Count Register

The Remote DMA Controller Remainder Count Register is located on the VMEbus adapter card (address = I/O Base + 0x18).

Before starting the DMA, the Remote DMA Controller Remainder Count Register is loaded with the lowest eight bits of the number of bytes to be transferred by the adapter DMA. The value in the register must be a multiple of 4 bytes for longword DMA transfers and a multiple of 2 bytes for word DMA transfers.

The same value must also be written to the Local DMA Remainder Count Register on the PCI adapter card.

6.3.7 Remote DMA VMEbus Address Registers

The Remote DMA VMEbus Address Registers are located on the VMEbus adapter card (address = I/O Base + 0x1A).

The 4-byte Remote DMA Address Registers are loaded by the user with the first VMEbus address to be accessed. As the DMA operation progresses, the contents of the registers increment by four for longword operations and by two for word operations.

The registers must be read as four bytes or two words.

The VMEbus address must be a multiple of four for longword transfers and a multiple of two for word DMAs.

If the DMA is to Dual Port RAM, only the lowest-order 15, 17, 20, 21, 22, or 23 bits in the address counter are used to access the 32K byte, 128K byte, 1M byte, 2M byte, 4M byte, or 8M byte Dual Port RAM. The upper bits in the Remote DMA Address Register are not used.

➔ To DMA to Dual Port RAM, set the Dual Port bit in the Local DMA Command Register. Do not DMA to the Dual Port RAM Window on the VMEbus.

➔ The PCI processor or a VMEbus master should never attempt to read remote I/O registers, remote bus I/O, or remote bus RAM during a DMA Controller operation.

6.3.8 Slave Status Register

A read of the Slave Status Register (address = I/O Base + 0x1E) provides the information defined in section 9.1.2.

A write to this register with data bit 7 set clears the errors reported by this register.

This register is usually only needed for Slave Mode DMA.

Chapter 7: The VMEbus Adapter Card

7.0 Introduction

The Model 617 adapter has three distinct parts: the PCI adapter card, the VMEbus adapter card, and the cable. Chapter 7 provides an overview of the VMEbus adapter card, including how the major features fit together and how they are used. Chapter 4 examines the PCI adapter card.

The primary use of the VMEbus adapter card is allowing PCI processors access to VMEbus devices. Also, it is used to access PCI devices and memory, to send interrupts to the PCI bus, and to begin DMA transfers between the VMEbus and PCI bus. A PCI processor must initialize the PCI adapter card before a VMEbus processor can access any PCI memory or start a DMA transfer.

The VMEbus adapter card has four major components: jumper blocks, CSR, Remote Memory Window, and Dual Port RAM Window. These components allow access to the VMEbus adapter card functions and control how the adapter performs the functions.

➔ Before the VMEbus adapter card is installed in the VMEbus card cage, it must be configured by setting the jumpers on the card.

➔ If the VMEbus adapter card is to be the system controller, the SYS, BGO-BGI and BR jumper blocks must changed. See section 10.4.

Notes about the VMEbus adapter card:

- All configuration is done via jumpers on the VMEbus adapter card.
- VMEbus masters can access up to 16M bytes of PCI memory.
- VMEbus processors can send and receive programmed interrupts to and from PCI processors.
- DMA transfers can be used to transfer data between VMEbus memory and PCI memory at rates up to 26M Bytes/sec and up to 16M bytes per transfer.
- The VMEbus adapter card can function as the system controller.

7.1 VMEbus Adapter Card Jumper Blocks

➔ VMEbus adapter card jumper blocks are diagrammed and described in Chapter 10.

The nine jumper blocks on the VMEbus adapter card control how the adapter operates. The jumper blocks are grouped and explained in the following table:

JUMPER BLOCK	LABEL	DESCRIPTION
Bus Request Level	BR	Determines the level at which the VMEbus adapter card requests the VMEbus, if it is the system arbiter, and selects the arbitration mode
Bus Grant Out & In	BGO-BGI	
Priority/Round-Robin	P/R	
Arbiter	ARB	
System	SYS	Controls several functions including if the adapter card is a transmitter and enables or disables the system controller features
I/O Window	I/O	Determines the location of the 32 bytes of CSR in VMEbus A16 address space
Dual Port RAM Window	Dual-Port	Determines the location of the Dual Port RAM Window in A24 and/or A32 address space. The starting and ending address are specified by the HI and LO jumpers. A32 and A24 can be enabled and disabled independently
Remote Memory Window	REM-RAM	Determines the location of the Remote Memory Window in A24 and/or A32 address space. The starting address and ending address are specified by the HI and LO jumpers. A32 and A24 can be enabled and disabled independently
Received Interrupt	R-INT	Determines how the VMEbus adapter card asserts its interrupts on the VMEbus backplane
Transmitted Interrupt	T-INT	Determines which VMEbus interrupts are passed to PCI
Address Bias	BIAS	Not used for the Model 617 adapter

7.2 VMEbus CSR

There are 32 CSRs that determine how the VMEbus adapter card functions and report its current status.

The I/O node registers are located in VMEbus A16 space. The starting address is defined by the I/O LO jumpers and continues for 32 bytes.

There are 16 local CSRs and 16 remote CSRs. Local CSRs are physically located on the VMEbus adapter card and do not use the cable when they are accessed. Remote CSRs are physically located on the PCI adapter card and a cable access is generated when any remote CSR is read or written.

The 32 CSRs are organized into four groups of eight registers each: local general CSRs, remote general CSRs, local DMA CSRs, and remote DMA CSRs. The local general CSRs are used for controlling adapter features that are implemented on the VMEbus adapter card as well as for reporting the current status of the VMEbus adapter card. The remote general CSRs are used for controlling features of the adapter that are implemented on the PCI adapter card and for determining the card's current status. The local DMA registers are used for checking the status of a DMA operation and for setting up the DMA registers that reside on the VMEbus adapter card, including the local address and the starting DMA address. The remote DMA registers are used for setting up DMA parameters for the PCI adapter card; for example, the starting DMA PCI address. See Chapter 9 for descriptions of each register.

7.3 Remote RAM Window

The Remote RAM Window allows VMEbus masters to become masters on the PCI bus. Memory accesses that fall within the Remote RAM Window are converted to accesses on the PCI bus. Consequently, VMEbus masters can use and control PCI devices and transfer data to PCI memory.

The location of the Remote RAM Window is determined by the REM-RAM LO and HI jumpers. VMEbus accesses that have addresses equal to or greater than the REM-RAM LO jumper setting but less than the REM-RAM HI jumper setting are sent across the cable to the PCI adapter card.

The Remote RAM Window can appear in A32 space, A24 space, or both spaces. The A24 window address is the A32 address truncated to the least significant 24 bits.

➔ Make sure the starting address of the Remote RAM Window is a multiple of the window size. We recommend that the Remote RAM Window always be placed on a 16M byte boundary. As a result, the first VMEbus-to-PCI Mapping Register will correspond to the start of the Remote Memory Window. See section 5.4.3.

➔ For more information on using the Remote RAM Window see section 8.2.

➔ For more information about setting the Remote RAM (REM-RAM) jumpers see section 10.3.4.

7.4 Dual Port RAM Window

The Dual Port RAM Window allows VMEbus masters to read or write the Dual Port RAM. Dual Port RAM is an optional expansion card that resides on the VMEbus adapter card and provides additional memory that can be accessed by both the PCI and VMEbus adapter cards. Because the Dual Port RAM card resides on the VMEbus adapter card, it does not require an additional VMEbus card slot or any remote resources for access.

The location of the Dual Port RAM Window is determined by the Dual-Port LO and HI jumpers. VMEbus accesses that have addresses equal to or greater than the Dual-Port LO jumper setting but less than the Dual-Port HI jumper setting are sent to the Dual Port RAM.

The Dual Port RAM Window can appear in A32 space, A24 space, or both spaces. The A24 window address is the A32 address truncated to the least significant 24 bits.

➔ The Dual Port RAM Window starting address should be a multiple of the window's size. For example, for a 512K byte Dual Port RAM Window, the starting address should start on a 512K byte boundary.

➔ For more information about using the Dual Port RAM Window see section 8.3.

➔ For more information about setting the Dual Port RAM (Dual-Port) jumpers see section 10.3.5.

7.5 VMEbus System Controller Mode

Every VMEbus chassis must have one and only one system controller. The system controller provides essential signals to all installed cards, determines which card has control of the VMEbus, and monitors bus activity. The system controller can either be a separate card or just a small part of another card in the VMEbus. In either case, the card with the active system controller must be installed in slot 1.

The VMEbus adapter card can be configured via jumpers to work in a VMEbus chassis that already has a system controller or it can provide the system controller functionality.

When configured for System Controller Mode, the Model 617 VMEbus card provides bus arbitration as a Single-Level (SGL) arbiter on level three or a four-level Priority (PRI) or Round-Robin (RRS) arbiter. The adapter operates in release-on-request mode except when the Bus Lock flip-flop is set.

In SGL arbitration mode, the adapter is the highest priority bus master. It responds to bus requests on level three from other bus masters, and activates the level three bus grant line when the adapter does not need the VMEbus.

A priority arbiter provides requesters preferential control of the data transfer bus over the other levels. By definition, BR3 is the highest priority, and BR0 is the lowest. When two or more requests are pending, the arbiter assigns control of the bus in the appropriate order by granting the bus in this sequence.

The priority arbiter must assert BCLR when a bus master of higher priority than the one in control of the bus initiates a request. When BBSY is asserted and a request is pending, the arbiter will drive BCLR if the pending request is of higher priority than the bus grant of the previous arbitration. Although the current bus master is not required to relinquish control of the bus in any prescribed time limit, it can continue transferring data until it reaches an appropriate stopping point.

A round-robin arbiter gives equal priority to all bus request levels. It grants control of the bus on a rotating basis. Upon release of the bus, the arbiter steps one level and tests for an active request and asserts a bus grant. If no request is active, it continues stepping through the levels until a request is found.

The round-robin arbiter can optionally drive the BCLR signal. In RRS mode BCLR is asserted whenever a master requests the bus on a level other than the last one granted. It does not assert BCLR if a master on the same level requests the bus.

When the VMEbus adapter card is the system controller for a VMEbus chassis it must be installed in slot 1. As the system controller, it provides the following functions:

- Bus arbitration as a single-level (SGL) bus arbiter or a four-level bus arbiter in Priority (PRI) or Round-Robin (RRS) mode.
- The SYSCLK and SYSRESET* signals.
- A 48 µsec bus timeout timer.

The system controller features are selected via three jumper blocks on the VMEbus adapter card: BGO-BGI, BR, and SYS. Bus arbitration mode is selected via the P/R and ARB jumper blocks. SGL arbitration is selected by default if all bus masters are requesting on level three.

If the VMEbus adapter card is installed in slot 1, all three jumper blocks must be configured for System Controller Mode. If the VMEbus card is installed in any other slot, the system controller features must be disabled. See section 10.4 for information about setting jumpers on the VMEbus adapter card for System Controller Mode.

➔ The VMEbus chassis will not operate correctly without a system controller, or if two or more cards are attempting to provide system controller functions.

➔ Make sure the VMEbus adapter card is jumpered correctly for the slot in which it is installed.

7.6 VMEbus Adapter Card LEDs

There are three LEDs on the VMEbus adapter card:

- The LED labeled READY is on when the programmable logic arrays on the VMEbus adapter card are successfully loaded after power-on. *This LED must be on for the card to operate.*

- The LED labeled REMOTE is on when the VMEbus adapter card is processing a command from the PCI adapter card.

- The LED labeled LOCAL is on when the VMEbus adapter card is addressed by the VMEbus chassis. This LED is lit if the adapter card recognizes a VMEbus address even if no active cycle (address strobe) is in progress.

Chapter 8: Using VMEbus Adapter Card Functions

8.0 Introduction

Chapter 8 explains how to use the various VMEbus adapter card functions including: accessing PCI memory, allowing PCI processors to access VMEbus memory, using interrupts, and starting a DMA transfer.

8.1 Initialization

Before VMEbus devices can use the features of the Model 617 adapter, the VMEbus adapter card should be initialized. The register accesses for initialization are outlined below.

1. Read from the Local Status Register to test if the remote chassis has power and that the cable is connected. If the remote chassis is not powered up or the cable is disconnected most of the adapter functions are useless (the optional Dual Port RAM will still work). Any attempts to access remote resources will result in interface timeouts.
2. Read from the Remote Status Register to flush interface errors caused by the power-on transition.
3. Write with data 80 (hex) to the Local Command Register to clear status register power-on errors.
4. Read from the Local Status Register to ensure no interface errors occurred and that the preceding steps were successful.

8.2 Accessing PCI Memory

The VMEbus processor can use the Model 617 adapter to access PCI memory and Dual Port RAM. In this chapter, PCI memory is often referred to as remote memory because the adapter creates a memory window on the VMEbus that allows access to the remote PCI memory. Accesses the VMEbus processor makes to the Remote RAM Window are translated by the adapter into memory accesses on the PCI bus. The Remote RAM Window can appear anywhere in either VMEbus A24 and/or A32 memory spaces (the location is set via the REM-RAM jumper block).

8.2.1 Remote RAM Jumpers

The REM-RAM jumper block allows the Remote RAM Window to be positioned anywhere in VMEbus memory. It can appear in either A24 or A32 address space or in both, and allows access to 64K bytes - 16M bytes of PCI memory from the VMEbus.

If VMEbus processors require access to PCI memory, the REM-RAM jumper block should be configured to enable the appropriate size Remote RAM Window at a convenient place in VMEbus memory space.

➔ Refer to section 10.3.4 for details on configuring the REM-RAM jumper block.

➔ Make sure the starting address of the Remote RAM Window is a multiple of the window size. We recommend that the Remote RAM Window always be placed on a 16M byte boundary. As a result, the first VMEbus-to-PCI Mapping Register will correspond to the start of the Remote Memory Window. See section 5.4.3.

8.2.2 Interaction with Mapping Registers

After the REM-RAM jumpers are set, the location and size of the Remote RAM Window are defined. Before VMEbus processors can begin using the Remote RAM Window, the PCI adapter card must be initialized by a PCI processor. The PCI processor must allocate PCI memory and program the PCI adapter card's VMEbus-to-PCI Mapping Registers to point to this memory. This allows the PCI adapter card to modify the address of the VMEbus remote memory access so that it will access the allocated PCI memory.

Once the VMEbus-to-PCI Mapping Registers have been initialized, VMEbus processors can access PCI memory as if it were local VMEbus memory appearing at the Remote Memory Window.

For more information on programming the PCI Mapping Registers, see section 5.4.

8.3 Accessing Dual Port RAM

In addition to allowing VMEbus masters access to PCI memory, the PCI adapter can also provide access to Dual Port RAM. Dual Port RAM, an optional memory card that installs on the VMEbus adapter card, can be accessed by both PCI bus and VMEbus masters at the same time. VMEbus masters use the Dual Port RAM Window to access Dual Port RAM.

The Dual Port RAM Window can be positioned anywhere in A32 and/or A24 VMEbus memory space via the Dual-Port jumper block.

To access Dual Port RAM, configure the Dual-Port jumper block so that the Dual Port RAM Window appears at a convenient starting address and matches the size of the Dual Port RAM. This window can then be used by any VMEbus device to access Dual Port RAM.

➔ Refer to section 10.3.5 for details on configuring the Dual-Port jumper block.

➔ The Dual Port RAM Window starting address should be a multiple of the window's size. For example, for a 512K byte Dual Port RAM Window, the starting address should start on a 512K byte boundary.

➔ The Dual Port RAM Window size should match the Dual Port RAM size. For a 32K byte Dual Port RAM, set the Dual Port Window to its minimum size of 64K bytes.

8.4 Allowing PCI Accesses

The VMEbus adapter card requires no setup to allow PCI bus masters to become masters on the VMEbus. Any setup necessary concerns registers on the PCI adapter card and is handled by a PCI processor.

➔ The Address Bias function is not needed on the Model 617 adapter. The jumpers must always be set to Pass Through Mode (the factory setting).

8.5 Handling Interrupts

Interrupts are used by hardware devices to signal that the device needs the processor's attention or that a specific event has occurred. When a hardware device asserts an interrupt, an Interrupt Service Routine (ISR) is expected to service the device and acknowledge the interrupt. A single hardware device may have several different reasons to interrupt the processor; therefore, the ISR must be able to determine why the device is interrupting and service that interrupt request.

When a VMEbus interrupts, it asserts one of the seven interrupt levels. The VMEbus processor that is watching that interrupt level starts an interrupt acknowledge (IACK) cycle and reads a value from the interrupting device. This value, the IACK vector, is used by the processor to call the correct ISR. The ISR reads the interrupting device's registers to determine the reason for the interrupt and writes the registers to service the interrupt.

The VMEbus adapter card can assert any of the seven VMEbus interrupt levels and can provide a single 8-bit IACK interrupt vector. The IACK vector can be programmed for any value between 0 and 255 by loading the Local Interrupt Vector Register. This value is used by the processor to select the correct ISR to service the VMEbus adapter card interrupts.

The VMEbus adapter card can generate interrupts on the VMEbus backplane from one of the following sources:

- Programmed interrupts from the PCI adapter card (see section 8.5.1);
- A Status Error Interrupt (see section 8.5.2);
- A DMA Done Interrupt (see section 8.5.3).

The VMEbus adapter card can also send VMEbus backplane interrupts 1 - 7 and programmed interrupts to the PCI adapter card.

➔ To receive an interrupt from the VMEbus adapter card, make sure the Disable All Interrupts From Adapter to VMEbus bit (bit 4) in the Local Command Register is clear. Otherwise, interrupt sources on the VMEbus adapter card will be blocked from reaching the VMEbus.

➔ To send a PT or VMEbus backplane interrupt to the PCI adapter card, the Disable All Interrupts From VMEbus To Adapter bit (bit 2) in the Local Command Register must be clear.

8.5.1 Programmed Interrupts

Programmed interrupts allow a PCI processor and a VMEbus processor to synchronize their communications. There are two types of programmed interrupts: Programmed interrupt to Transmitter (PT) and Programmed interrupt to Receiver (PR). Since the adapter has a hardware cable conflict mechanism, you can use either type of programmed interrupt.

8.5.1.1 Sending PT Interrupts

The PT Interrupt allows a local processor to generate an interrupt on the remote bus without using the cable. To send a PT Interrupt to the PCI adapter card, the VMEbus processor sets the Send PT Interrupt bit (bit 5) of the Local Command Register.

The PT Interrupt pin must be jumpered to one of the CINTx pins in the T-INT jumper block before the PT Interrupt will be transmitted to the PCI adapter card.

➔ Make sure the Disable All Interrupts From VMEbus To Adapter bit (bit 2) in the Local Command Register is cleared.

➔ The CINT line used to sent the PT Interrupt to PCI must not be used to transmit any other type of interrupt including the PT Interrupt from PCI to VMEbus.

➔ See section 5.5.1.2 for information on how to receive a PT Interrupt on the PCI adapter card.

8.5.1.2 Receiving PT Interrupts

If a PCI processor sends a PT Interrupt to the VMEbus, the Receiving PT Interrupt bit (bit 1) of the Remote Status Register will be set. A VMEbus ISR can read this bit to determine if a PT Interrupt is the cause of the interrupt. The PT Interrupt can then be cleared by setting the Clear PT Interrupt bit (bit 6) of the Remote Command Register.

➔ Make sure the Disable All Interrupts From Adapter To VMEbus bit (bit 4) in the Local Command Register is cleared.

➔ If the PCI adapter card has been configured to send its PT Interrupt on cable interrupt line 1 or 2, the CINT1 or CINT2 pin must be jumpered to either the VMEbus IRQ1 or VMEbus IRQ2 pin of the R-INT jumper block.

➔ The R-INT jumper block is diagrammed and discussed in section 10.3.7.

➔ For information on how the PCI adapter card sends a PT Interrupt to the VMEbus adapter card, see section 5.5.1.1.

8.5.1.3 Sending PR Interrupts

The PR Interrupt allows the local processor to generate an interrupt on the remote bus and the remote processor will not need to use the cable to acknowledge the interrupt. To send a PR Interrupt to the PCI adapter card, a VMEbus processor sets the Send PR Interrupt bit (bit 5) of the Remote Command Register.

➔ For information about how to receive a PR Interrupt on the PCI adapter card, see section 5.5.1.4.

8.5.1.4 Receiving PR Interrupts

When a PCI processor sends a PR Interrupt to the VMEbus, the VMEbus processor must be able to determine that the PR Interrupt is the source of the interrupt and be able to acknowledge the PR Interrupt. The Local Status Register can be read and the Receiving PR Interrupt bit (bit 5) will be set when the PR Interrupt is the source of the interrupt. To acknowledge a PR Interrupt, the VMEbus processor sets the Clear PR Interrupt bit (bit 6) of the Local Command Register.

➔ Make sure the Disable All Interrupts From Adapter To VMEbus bit (bit 4) in the Local Command Register is cleared.

➔ To receive a PR Interrupt from the PCI adapter card, the PR Interrupt pin must be jumpered to either the VMEbus IRQ1 or VMEbus IRQ2 pin in the R-INT jumper block.

➔ The R-INT jumper block is diagrammed and discussed in section 10.3.7.

➔ For information on how to send a PR Interrupt from the PCI adapter card to the VMEbus adapter card, see section 5.5.1.3.

8.5.2 Error Interrupts

The VMEbus adapter card can generate an interrupt when it detects that an error occurred. The following four errors are always monitored and recorded in the Local Status Register:

- A remote bus error -- An access to PCI bus space resulted in a bus error (target abort) on the PCI bus.
- An interface timeout -- An access to a remote register did not complete in 40 μsec or a DMA transfer did not complete in 16 msec.
- LRC error -- A Longitudinal Redundancy Check error was detected during a DMA transfer.
- Parity error -- A parity error was detected on information that was transferred over the cable.

The four errors can also cause a VMEbus interrupt. The Error Interrupt is enabled by connecting the Error Interrupt pin to either the VMEbus IRQ1 or VMEbus IRQ2 pins in the R-INT jumper block. If this connection is made and an error occurs, the VMEbus adapter card will assert the appropriate VMEbus interrupt level.

When the Error Interrupt is enabled an ISR should check the Local Status Register to see if an error is the source of the interrupt. Then, if any error bit is set (bits 7, 6, 3, and 2), an error is the source of the interrupt. The Error Interrupt can be acknowledged by setting the Clear Status Errors bit (bit 7) of the Local Command Register.

➔ Make sure the Disable All Interrupts From Adapter To VMEbus bit (bit 4) in the Local Command Register is cleared

➔ The R-INT jumper block is diagrammed and discussed in section 10.3.7.

8.5.3 DMA Interrupts

The VMEbus adapter card can be configured to assert an interrupt when the current DMA finishes. Before the DMA Done Interrupt can be received, the DMA Done pin must be connected to either VMEbus IRQ1 or VMEbus IRQ2 in the R-INT jumper block and enabled by setting the DMA Done Interrupt bit (bit 2) of the Local DMA Command Register. See section 8.6.4.1 for additional information about the DMA Done Interrupt.

An ISR can tell if the DMA Done Interrupt is active by reading the DMA Done bit (bit 1) of the Local DMA Command Register and can acknowledge it by clearing the bit.

➔ Make sure the Disable All Interrupts From Adapter To VMEbus bit (bit 4) in the Local Command Register is cleared

➔ See section 8.6 for information on how to program a DMA for the VMEbus.

➔ The R-INT jumper block is diagrammed and discussed in section 10.3.7.

8.5.4 Sending Backplane Interrupts To PCI

The VMEbus adapter card can send any of the seven VMEbus interrupt levels to the PCI bus. Therefore, the PCI processors can service interrupting VMEbus devices.

To pass a VMEbus interrupt level to PCI, the corresponding jumper in the T-INT jumper block must be installed. For example, if VMEbus interrupt levels 5 and 6 are to be handled by a PCI processor, then jumpers 5 and 6 of the T-INT jumper block should be installed to connect VMEbus interrupt levels 5 and 6 to cable interrupts 5 and 6. The T-INT jumper block is diagrammed in section 10.3.6.

➔ Remember that only one device may be assigned to respond to any VMEbus interrupt level. If a VMEbus interrupt level is going to be sent to the PCI bus, no other VMEbus device can respond to that level.

➔ If a CINT line has been assigned to carry a PT Interrupt, it may not be used to pass a backplane interrupt.

➔ Refer to section 5.5.3 for detailed information about receiving VMEbus backplane interrupts.

8.5.5 Writing An ISR

An Interrupt Service Routine (ISR) is a software routine that handles the interrupts generated by a specific hardware device. A general ISR procedure for the VMEbus adapter card is provided below.

1. If the Error Interrupt is jumpered, check for status errors. Read the Local Status Register to check if the Parity Error bit, the LRC Error bit, the Interface Timeout bit, and/or the Remote Bus Error bit are set. If a status error occurred, set the Clear Error bit in the Local Command Register and exit the ISR.

2. If the PR Interrupt is jumpered, check for a PR Interrupt. Read the Local Status Register to see if the Receiving PR Interrupt bit is set. If a PR Interrupt is active, set the Clear PR Interrupt bit in the Local Command Register and exit the ISR.

3. If the DMA Done Interrupt is jumpered, check for a DMA Done Interrupt. Read the Local DMA Command Register to see if both the DMA Done bit and DMA Interrupt Enable bit are set. If the DMA Done Interrupt is active, clear the DMA Done bit in the Local DMA Command Register and exit the ISR.

4. Check for the PT Interrupt. Read the Remote Status Register and see if the Receiving PT Interrupt bit is set. If the PT Interrupt is active, set the Clear PT Interrupt bit in the Remote Command Register and exit the ISR.

➔ Remember to initialize the Local Interrupt Vector Register and install an ISR to handle the interrupt.

8.6 Initiating A DMA Operation

The Model 617 adapter supports Direct Memory Access (DMA) transfers. DMA is a method of transferring data between the PCI bus and VMEbus that has two distinct advantages over random access (PIO) transfers:

- DMA transfer rates are approximately ten times faster than PIO transfers.
- DMA transfers require very little attention from the processor. The processor is only required to set up a few registers, start the DMA, and check for errors after the DMA completes. While the DMA transfer occurs, the processor can do other processing that doesn't require use of the adapter.

A DMA can be started from either the PCI bus or the VMEbus. However, only one DMA can occur at one time. Consequently, neither the PCI nor VMEbus processor can start another DMA until the current one is finished. To avoid a DMA conflict in which both sides try to start a DMA at the same time, it often is best to have only one side program and start DMA transfers. Normally, the PCI processor does all the DMA programming because the DMA-to-PCI Mapping Registers must be setup before the DMA starts and they are not visible from the VMEbus.

➔ Neither system is allowed to make remote accesses while a DMA transfer is in progress.

➔ During DMA transfers, interrupt passing between the adapter cards must be disabled.

➔ For information on starting a DMA transfer from the PCI side of the adapter, see section 5.6.

8.6.1 PCI Initialization

The PCI adapter card has several registers that control its operation. Most of the PCI registers are not accessible from the VMEbus; therefore, they must be initialized by a PCI processor before a VMEbus processor can effectively use the adapter.

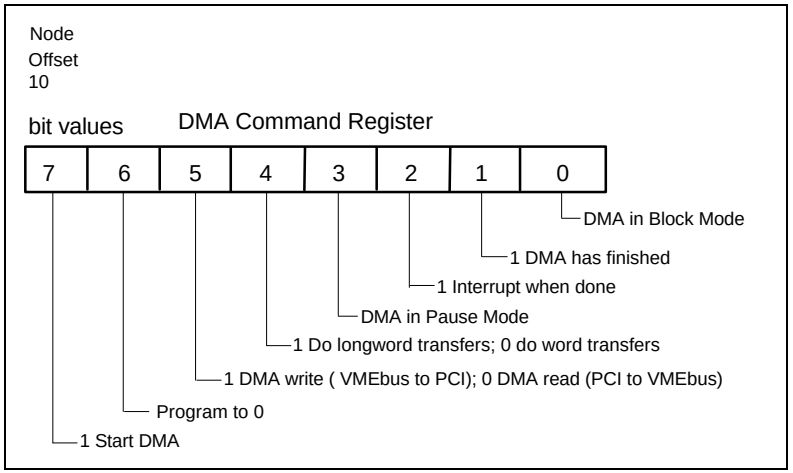
The DMA-to-PCI Mapping Registers translate the PCI DMA address into a physical PCI bus address. These registers must be initialized by a PCI processor to point to PCI memory. The PCI processor must also communicate to the VMEbus processor which DMA-to-PCI Mapping Registers were initialized. The VMEbus processor needs this information to form the starting remote DMA address value and to determine how long the DMA can be before it can program a DMA transfer.

➔ For information on how to program the DMA-to-PCI Mapping Registers, see section 5.6.1.

8.6.2 DMA CSRs

To start a DMA transfer, the following DMA CSRs need to be programmed by a VMEbus master:

DMA REGISTER	PROGRAMMING
Local DMA Address	Load with the VMEbus address of the first location to be transferred
Remote DMA Address	Load bits 11-0 with A11 through A0 of the first PCI physical address to access. Load bits 23-12 with the starting DMA-to-PCI Mapping Register to use
Local DMA Remainder Count	Load with the least significant 8 bits of the DMA length in bytes. This is the same as the DMA length in bytes modulo 256
Remote DMA Remainder Count	Load with the same value as the Local DMA Remainder Count Register
Local DMA Packet Count	Load with the DMA transfer size in bytes divided by 256
Local DMA Command	Load with a bit mask indicating how the DMA is to be performed (write/read, block/non-block). Must be loaded twice: first, at the beginning of DMA programming, with all appropriate bit settings except the DMA Start bit (bit 7); then, again at the end of DMA programming with the same bits set as before plus the DMA Start bit. The second write starts the DMA. See figure on next page



Three DMA operating modes may be used when accessing the VMEbus:

- Block Mode -- in Block Mode, the VMEbus DMA Controller never transfers more than 256 bytes before it rearbitrates for the VMEbus. Block Mode has the highest data throughput, but may starve other VMEbus devices.

  Block Mode is activated by setting bit 0 of the Local DMA Command Register.

- Pause Mode -- in Pause Mode, the VMEbus DMA Controller never transfers more than 64 bytes before it rearbitrates for the bus. Pause Mode allows other VMEbus devices to use the VMEbus more frequently than Block Mode during DMA.

  Pause Mode is activated by setting bits 3 and 0 of the Local DMA Command Register.

- Non-Block Mode -- the VMEbus adapter card rearbitrates for the VMEbus after each transfer. Non-Block Mode operation is identical to random access.

  Non-Block Mode is activate when bit 0 of the Local DMA Command Register is clear.

8.6.3 Other CSRs

In addition to DMA CSRs, three other registers must be programmed before the DMA Start bit is set:

REGISTER	PROGRAMMING
Local Command	Interrupt passing to the PCI adapter card must be disabled by setting the Disable All Interrupts From VMEbus To Adapter bit (bit 2)
Local Address Modifier	The VMEbus address modifier to be used during the DMA must be loaded into this register
Local Interrupt Vector	If the DMA Done Interrupt is enabled and jumpered, this register must be loaded with the interrupt vector of the ISR that will handle the DMA Done Interrupt

8.6.4 When Is The DMA Operation Done?

There are two ways to tell if the DMA has completed: wait for the DMA Done Interrupt, and polling for the DMA Done bit.

8.6.4.1 DMA Done Interrupt

The VMEbus adapter card can assert VMEbus interrupt level 1 or level 2 when the DMA completes.

To enable the DMA Done Interrupt so that the VMEbus adapter card will interrupt the VMEbus when the DMA completes, the DMA Done Interrupt bit (bit 2) in the Local DMA Command Register must be set and the DMA Done Interrupt pin must be connected to VMEbus IRQ1 or VMEbus IRQ2 pins in the R-INT jumper block. The application needs to have an ISR installed to handle the DMA Done Interrupt at the interrupt vector that was loaded into the Local Interrupt Vector Register.

The ISR can read the DMA Done bit (bit 1) of the Local Command Register to tell if the DMA Done is the cause of the interrupt. To acknowledge a DMA Done Interrupt the DMA Done bit (bit 1) of the Local DMA Command Register should be cleared.

➔ The Disable All Interrupts From Adapter To VMEbus bit (bit 4) in the Local Command Register must be clear. Otherwise, the DMA Done Interrupt will be blocked from the VMEbus.

➔ The DMA Done Interrupt occurs whether the DMA completed successfully or not. Consequently, the application must search for status errors by reading the Local Status Register.

8.6.4.2 Polling For DMA Done Bit

The DMA Done bit (bit 1) in the Local DMA Command Register indicates if the DMA operation is complete. When the register is read, if bit 1 is set, the DMA transfer is done.

➔ The DMA Done bit is set whether the DMA completed successfully or unsuccessfully. Therefore, the application must look for status errors by reading the Local Status Register.

➔ Constantly reading the DMA Command Register while a DMA transfer is in progress degrades DMA performance.

8.6.5 Programming Sequence For Initiating A DMA From VMEbus

1. Load the Local DMA Command Register. Set all appropriate bits except the DMA Start bit. The DMA Start bit must be clear.

2. Load the Local DMA Address Register. The starting VMEbus address for the DMA should be placed in this register.

3. Load the Remote DMA Address Register. Bits 23-12 should be the starting DMA-to-PCI Mapping Register numbered from 0. Bits 11-0 should be the starting PCI bus address bits A11-A0.

4. Load the Local DMA Remainder Count Register. The remainder count is the DMA length in bytes modulo 256.

5. Load the Remote DMA Remainder Count Register. Load with the same value as the Local DMA Remainder Count Register.

6. Load the Local DMA Packet Count Register. The packet count is the DMA length in bytes divided by 256.

7. For a DMA Done Interrupt, make sure the Local Interrupt Vector Register is loaded with the correct vector for your installed ISR.

8. Load Local Command Register. Disable the VMEbus adapter card from sending interrupts to the PCI adapter card by setting bit 2. Make sure bit 4 is clear if the DMA Done Interrupt is enabled and jumpered.

9. Load the Local Address Modifier Register. The address modifier must match the Block Mode bit and the Local DMA address programmed above.

10. Read the Local DMA Command Register and set the DMA Start bit. Write this value to the Local DMA Command Register. The DMA transfer begins.

11. If the DMA Done Interrupt is enabled, wait for the interrupt; or poll the Local DMA Command Register to see if the DMA Done bit is set. The DMA transfer is done.

12. Check for status errors by reading the Local Status Register.

13. Clear the Disable VMEbus to Adapter Interrupt bit (bit 2) in the Local Command Register. Any interrupts that were pending while the DMA was in progress will be passed to the PCI bus.

8.6.6 Things To Remember

- Never make a cable access while a DMA transfer is in progress from either side of the adapter.

- Make sure the local address modifier matches the local DMA address. For example, if a 24-bit address was loaded, make sure the address modifier is appropriate for A24.

- Make sure the local address modifier matches the Block Mode bit in the DMA Command Register. For example, if the Block Mode bit is set, the address modifier must be appropriate for Block Mode.

- Make sure to disable interrupts to the PCI bus. Bit 2 of the Local Command Register must be set.

- Make sure bit 4 of the Local Command Register is not set if the DMA Interrupt is enabled. Setting bit 4 would prevent the DMA Done Interrupt from reaching the VMEbus.

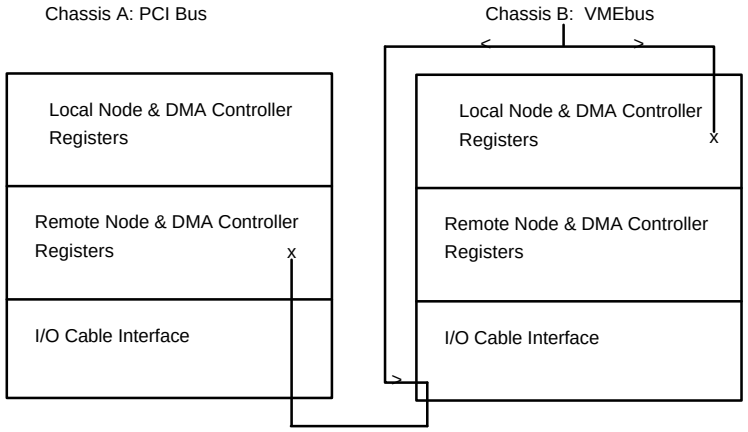
Chapter 9: CSR Accessed From The VMEbus

9.0 VMEbus CSR

Chapter 9 describes the VMEbus adapter card Control and Status Registers (CSR). These registers are accessed by a VMEbus master through I/O space window that is defined with the I/O HI and I/O LO jumpers on the VMEbus adapter card.

The first eight bytes of the VMEbus adapter card's I/O space are for a VMEbus processor to talk to *its* adapter Local Node Registers. The second eight bytes are for the VMEbus processor to talk to its Remote Node (PCI) adapter Registers. The next eight bytes of the I/O space are for the VMEbus to talk to the Local (VMEbus) DMA Controller Registers. The final eight bytes comprise the Remote (PCI) DMA Controller Registers.

➔ All node I/O registers can be accessed as either bytes or words, but not as longwords.



9.1 Local Node Registers

The following table shows the location and definition of the first eight bytes of I/O on the VMEbus adapter card. These registers are located on the VMEbus adapter card and addressed by VMEbus masters.

VMEbus I/O ADDR (hex)	WRITE FUNCTION	READ FUNCTION
I/O LO + 00	-- reserved --	-- reserved --
I/O LO + 01	Local Command	Local Command
I/O LO + 02	-- reserved --	-- reserved --
I/O LO + 03	-- reserved --	Local Status
I/O LO + 04	Address Modifier	Address Modifier
I/O LO + 05	-- reserved --	-- reserved --
I/O LO + 06	-- reserved --	-- reserved --
I/O LO + 07	Interrupt Vector	Interrupt Vector

9.1.1 Local Command Register

The VMEbus Local Command Register is a read/write register located on the VMEbus adapter card (address = I/O LO + 0x01).

BIT	FUNCTION
7	Clear Status Register Errors (write only)
6	Clear PR Interrupt (write only)
5	Send PT Interrupt
4	Disable All Interrupts From Adapter To VMEbus
3	reserved -- program to "0"
2	Disable All Interrupts From VMEbus To Adapter
1	reserved -- program to "0"
0	reserved -- program to "0"

CLEAR STATUS REGISTER ERRORS (bit 7): Communication between the two systems is monitored for I/O cable parity errors, PCI bus errors, DMA LRC errors, and interface timeouts. Any of these types of errors sets its corresponding bit in the Local Status Register. These error bits stay set until cleared by writing a "1" to bit 7 of the Local Command Register.

CLEAR PR INTERRUPT (bit 6): Writing a "1" to this bit clears a PR Interrupt that was sent from a PCI processor. The Receiving PR Interrupt bit in the Local Status Register is also cleared.

SEND PT INTERRUPT (bit 5): When this bit is set to "1", a PT Interrupt is transmitted to the PCI system via the cable interrupt lines. When this bit is reset to "0", the PT Interrupt request is removed.

DISABLE ALL INTERRUPTS FROM ADAPTER TO VMEbus (bit 4): If this bit is set to "1", the adapter card cannot pass interrupts to the VMEbus backplane.

DISABLE ALL INTERRUPTS FROM VMEbus TO ADAPTER (bit 2): If this bit is set to "1", the VMEbus adapter card will not send any interrupts to the PCI adapter card. Set this bit before a DMA transfer is started and clear it after the DMA is finished.

9.1.2 Local Status Register

The Local Status Register is a read-only register located on the VMEbus adapter card (address = I/O LO + 0x03).

BIT	FUNCTION
7	Interface Parity Error
6	Remote Bus Error
5	Receiving PR Interrupt
4	reserved
3	LRC Error
2	Interface Timeout
1	Sending PT Interrupt
0	Remote Bus Power Off or I/O Cable Is Off

INTERFACE PARITY ERROR (bit 7): Bit 7 is set to "1" if an interface parity error occurs on a chassis-to-chassis transfer. It is reset to "0" when the Clear Status Register bit is set in the Local Command Register.

REMOTE BUS ERROR (bit 6): This bit is set to a "1" if an access to PCI: tries to use an invalid Mapping Register; results in a PCI bus parity error; is not responded to by a PCI slave; or is aborted by the PCI slave. It is reset to "0" when the Clear Status Register bit is set in the Local Command Register.

RECEIVING PR INTERRUPT (bit 5): Set to "1" when a PR Interrupt is received from the PCI adapter card. This bit is reset to "0" when the Clear PR Interrupt bit is set to "1" in the Local Command Register.

LRC ERROR (bit 3): Set to "1" if the Longitudinal Redundancy Check circuitry detects an error on a DMA transfer. The bit is reset to "0" when the Clear Status Register bit is set to "1" in the Local Command Register.

INTERFACE TIMEOUT (bit 2): Set to "1" if a DMA transfer does not complete within 16 msec or a Remote Register access does not complete within 40 ###sec. It is reset when the Clear Status Register bit is set to "1" Local Command Register.

SENDING PT INTERRUPT (bit 1): Set to "1" if the VMEbus adapter card is currently sending a PT Interrupt to the PCI. It is reset to "0" when a PCI processor acknowledges the interrupt.

REMOTE BUS POWER OFF or I/O CABLE IS OFF (bit 0): Bit 0 is a "1" if the PCI system power is off or if the I/O cable is not connected. If this bit is a "1", attempts to communicate with the PCI bus result in interface errors. It is automatically reset to "0" when the source of the error is corrected.

9.1.3 Address Modifier Register

The Address Modifier Register is a read/write register used only during adapter DMA Controller operations to present an address modifier to the VMEbus. The Address Modifier Register is located on the VMEbus adapter card (address = I/O LO + 0x04).

The Address Modifier Register is loaded, before starting the DMA Controller operation, with the address modifier that signals the proper address width and Block/Non-Block transfer mode.

9.1.4 Interrupt Vector Register

This 8-bit read/write register, located on the VMEbus adapter card (address = I/O LO + 0x07), holds the interrupt vector presented to a VMEbus processor when that processor acknowledges an interrupt from the VMEbus adapter card.

If the VMEbus adapter card asserts an interrupt, it passes the contents of the Interrupt Vector Register to the local VMEbus processor during the interrupt acknowledge cycle.

The Interrupt Vector Register is preset to FF at power-on time. The register is read from and written to by a VMEbus processor -- *not* by a PCI processor.

9.2 Remote Node Registers

The eight Remote Node Registers are accessed by processors on the VMEbus, but are located on the *remote* (PCI) adapter card.

VMEbus I/O ADDR (hex)	WRITE FUNCTION	READ FUNCTION
I/O LO + 08	-- reserved --	-- reserved --
I/O LO + 09	Remote Command	Remote Status
I/O LO + 0A	-- reserved --	-- reserved --
I/O LO + 0B	-- reserved --	-- reserved --
I/O LO + 0C	-- reserved --	-- reserved --
I/O LO + 0D	PCI Adapter ID	PCI Adapter ID
I/O LO + 0E	-- reserved --	-- reserved --
I/O LO + 0F	-- reserved --	-- reserved --

9.2.1 Remote Command Register

The Remote Command Register is an 8-bit write-only register located on the PCI adapter card (address = I/O LO + 0x09). This register is addressed by VMEbus processors.

BIT	FUNCTION
7	Reset Remote Register
6	Clear PT Interrupt
5	Send PR Interrupt
4	reserved
3	reserved
2	reserved
1	reserved
0	reserved

RESET REMOTE REGISTER (bit 7): When this bit is set to "1", the Remote Parity Error Status bit in the Remote Status Register is reset.

CLEAR PT INTERRUPT (bit 6): When the PCI adapter card is sending a PT Interrupt to the VMEbus, this bit is used to clear this interrupt. Setting this bit will cause the PT Interrupt to be cleared.

SEND PR INTERRUPT (bit 5): A VMEbus processor can send a PR Interrupt to the PCI bus by setting this bit. The VMEbus processor can remove its PR Interrupt request by writing a "0" to this bit.

9.2.2 Remote Status Register

The Remote Status Register is an 8-bit, read-only register located on the PCI adapter card (address = I/O LO + 0x09).

BIT	FUNCTION
7	reserved
6	reserved
5	Sending PR Interrupt
4	reserved
3	Remote Parity Error
2	reserved
1	Receiving PT Interrupt
0	reserved

SENDING PR INTERRUPT (bit 5): When this bit is set to "1", the VMEbus adapter card is sending a PR Interrupt to the PCI bus.

REMOTE PARITY ERROR (bit 3): When this bit is set to "1", a PCI bus parity error was detected during a PCI bus master operation.

RECEIVING PT INTERRUPT (bit 1): When this bit is set to "1", the PCI adapter card is sending a PT Interrupt to the VMEbus.

9.2.3 PCI Adapter ID Register

A byte read of this 8-bit, read-only register (address = I/O LO + 0x0D) returns the hex value 0xAB that identifies the card on the other end of the cable as a PCI card. A write to this register has no effect.

9.3 DMA Controller Registers

Sections 9.3.1 - 9.3.7 describe the DMA Controller Registers accessed from the VMEbus. See section 3.3 for an overview of DMA transfers.

➔ See section 8.6 for specific information about programming a DMA transfer.

9.3.1 DMA Controller Registers Accessed From The VMEbus

The registers listed in the table below are located on the VMEbus adapter card and are addressed by processors on the VMEbus system.

VMEbus I/O ADDR (hex)	WRITE FUNCTION	READ FUNCTION
I/O LO + 10	DMA Command	DMA Command
I/O LO + 11	DMA Remainder Count	DMA Remainder Count
I/O LO + 12	DMA VMEbus Address 24-31	DMA Address 24-31
I/O LO + 13	DMA VMEbus Address 16-23	DMA Address 16-23
I/O LO + 14	DMA VMEbus Address 8-15	DMA Address 8-15
I/O LO + 15	DMA VMEbus Address 0-7	DMA Address 0-7
I/O LO + 16	DMA Packet Count 8-15	DMA Packet Count 8-15
I/O LO + 17	DMA Packet Count 0-7	DMA Packet Count 0-7

The registers listed in the following table are located on the PCI adapter card and are addressed by VMEbus processors.

VMEbus I/O ADDR (hex)	WRITE FUNCTION	READ FUNCTION
I/O LO + 18	DMA PCI Remainder Count	DMA PCI Remainder Count
I/O LO + 19	-- reserved --	-- reserved --
I/O LO + 1A	-- reserved --	-- reserved --
I/O LO + 1B	-- reserved --	-- reserved --
I/O LO + 1C	DMA PCI Address 8-15	DMA PCI Address 8-15
I/O LO + 1D	DMA PCI Address 2-7	DMA PCI Address 2-7
I/O LO + 1E	-- reserved --	-- reserved --
I/O LO + 1F	DMA PCI Address 16-23	DMA PCI Address 16-23

9.3.2 Local DMA Controller Command Register

The local DMA Controller Command Register is located on the VMEbus adapter card (address = I/O LO + 0x10).

BIT	FUNCTION
7	Start DMA
6	reserved - program to "0"
5	DMA Transfer Direction
4	DMA Word/Longword Select
3	DMA Local Pause
2	Enable DMA Done Interrupt
1	DMA Done Flag
0	DMA Local Block Mode

START DMA (bit 7): Writing a "1" to bit 7 starts a DMA Controller operation. This bit should be set only after all other DMA registers are ready. DMA transfers that exceed 16 msec are aborted and the interface timeout status error is set.

➔ DMA may only occur between VMEbus and PCI memory; not between VMEbus memory and Dual Port RAM.

DMA TRANSFER DIRECTION (bit 5): Writing a "1" to bit 5 causes the DMA Controller to do a write and transfer data from VMEbus to PCI bus. Writing a "0" causes a DMA read and transfers data from PCI bus to VMEbus.

DMA WORD/LONGWORD SELECT (bit 4): Writing a "1" to bit 4 causes the DMA Controller to perform longword (32-bit) data transfers. Writing a "0" to this bit causes word (16-bit) data transfers.

DMA LOCAL PAUSE (bit 3): Writing a "1" to this bit causes the VMEbus adapter card DMA Controller to pause at least once every 64 bytes. This pause allows other VMEbus masters to receive a bus grant quickly during a DMA operation. For Pause Mode to operate, the Block Mode bit must also be set.

ENABLE DMA DONE INTERRUPT (bit 2): Writing a "1" to bit 2 enables the DMA Done Interrupt on the VMEbus adapter card at the completion of a DMA operation. The DMA Done Interrupt pin must also be jumpered to the VMEbus IRQ1 or VMEbus IRQ2 pin in the R-INT jumper block.

DMA DONE FLAG (bit 1): Presents the status of a DMA operation to software in the same manner as the DMA Done Interrupt does for the hardware. It clears itself when a new DMA operation begins. Writing a "0" to bit 1 also clears the DMA Done Interrupt.

DMA LOCAL BLOCK MODE (bit 0): Writing a "1" to this bit causes the VMEbus adapter card DMA Controller to pause at least once every 256 bytes. Block Mode is the fastest DMA mode; however, it may keep other VMEbus devices from using the VMEbus. If this is a problem, set the Pause Mode bit.

9.3.3 Local DMA Remainder Count Register

The Local DMA Remainder Count Register is an 8-bit read/write register located on the VMEbus adapter card (address = I/O LO + 0x11).

Before starting DMA, the Local DMA Remainder Count Register is loaded with the lowest eight bits of the number of bytes to be transferred by the adapter DMA Controller. The remainder count is the byte count modulo 256. The values in the register must be multiples of 4 bytes for longword transfers and multiples of 2 bytes for word transfers. The same value must also be written to the Remote DMA Remainder Count Register.

9.3.4 Local DMA VMEbus Address Register

The Local DMA VMEbus Address Register is composed of four 8-bit read/write registers located at address I/O LO + 0x12 through I/O LO + 0x15. The register's four bytes are loaded (before starting the DMA) with the first address to be accessed on the VMEbus. As the DMA operation progresses, the contents of the register increment by four for longword operations and by two for word operations. An I/O read of these registers during DMA produces the address count at that time in the DMA (the *next* address to be accessed).

DMA VMEbus address bit 31 (written to bit 7 of I/O LO + 0x12) is the most significant bit of the address. DMA VMEbus address bit 0 (written to bit 0 of I/O LO + 0x15) is the least significant address bit.

For longword transfers, the address must be a multiple of four. For word transfers, the address must be a multiple of two.

The Local DMA VMEbus Address Register can be accessed as either 4-byte registers or 2-word registers, not as a longword.

9.3.5 Local DMA Packet Count Registers

The Local DMA Packet Count Registers, located on the VMEbus adapter card, are found at address I/O LO + 0x16 through I/O LO + 0x17.

Before starting the DMA, the registers are loaded with the upper two significant bytes of the number of bytes to be transferred during the DMA. The packet count is the number of bytes to be transferred divided by 256.

As the DMA Controller operation progresses, the contents of the registers decrement by one for every data packet transferred. An I/O read of the registers during DMA produces the number of packets remaining to be transferred at that time (within 256 bytes).

9.3.6 Remote DMA Remainder Count Register

The Remote DMA Remainder Count Register is an 8-bit, read/write register located on the PCI adapter card (address = I/O LO + 0x18) that is addressed by VMEbus processors.

This register is loaded with the least significant byte of the number of bytes to be transferred during a DMA operation. The remainder count is the byte count modulo 256. The number of bytes transferred must be a multiple of four bytes for longword transfers and two bytes for word transfers.

The same value must be loaded into the Local and Remote DMA Remainder Count Registers.

9.3.7 Remote DMA PCI Address Registers

The three, 24-bit Remote DMA PCI Address Registers are located on the PCI adapter card (address = I/O LO + 0x1C, I/O LO + 0x1D, and I/O LO + 0x1F). See also section 6.3.7.

The Remote DMA PCI Address Registers are loaded from the VMEbus (before starting the DMA) with the first DMA-to-PCI Mapping Register to be used and the low order bit of the PCI physical starting address.

The address must be a multiple of four for longword DMAs and a multiple of two for word DMAs.

➔ A VMEbus processor should never attempt to read or write PCI I/O registers or PCI RAM during a DMA Controller operation.

PCI DMA ADDRESS REGISTER (bits 12 - 23): Bits 12 - 23 indicate which DMA-to-PCI Mapping Register is used during the start of the DMA.

PCI DMA ADDRESS REGISTER (bits 0 - 11): Hold the starting PCI physical address bits A11 - A0.

Chapter 10: Setting The VMEbus Adapter Card Jumpers

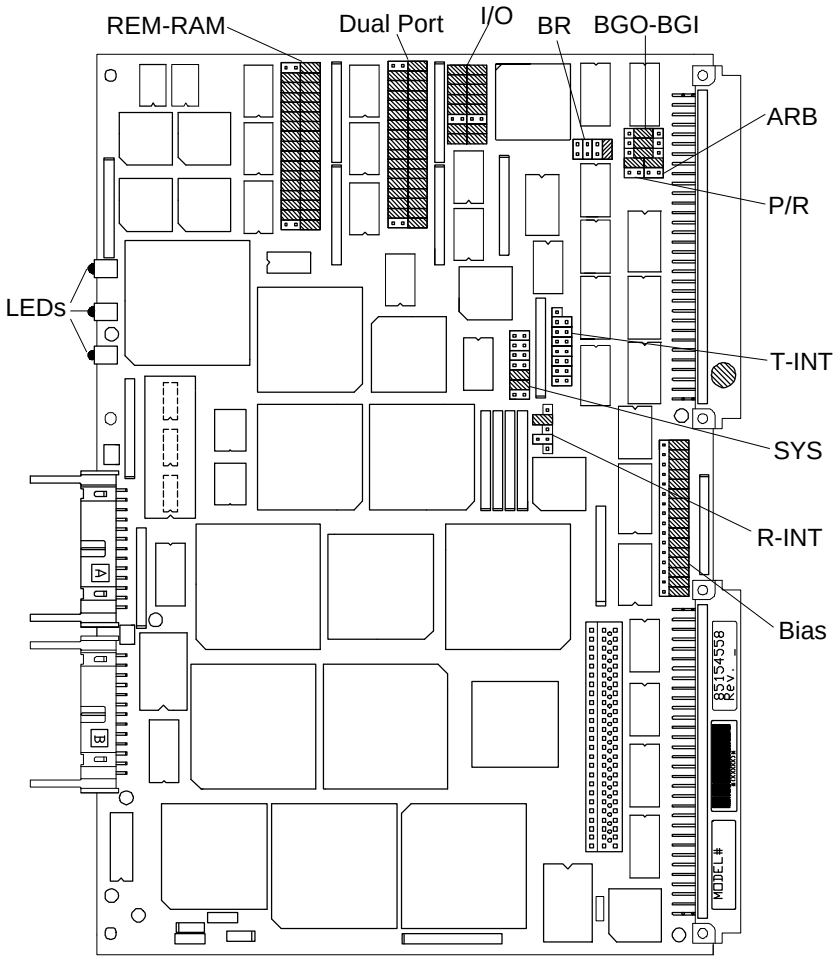
10.0 Introduction

Chapter 10 provides detailed descriptions of jumper blocks on the VMEbus adapter card and instructions for their configuration. It is important that you understand how each feature is configured in order to use the Model 617 adapter correctly.

10.1 Configuration Notes

- The factory settings are listed in section 10.2 and diagrammed in each section that describes a jumper block.
- When the VMEbus adapter card is installed in slot 1 as system controller, the SYS and BGO-BGI and BR jumper blocks must be changed. See sections 10.3.1, 10.3.2 and 10.4.

10.1.1 VMEbus Adapter Card Diagram



Location of Jumper Blocks and LEDs

10.2 VMEbus Adapter Card Factory Settings

The VMEbus adapter card is configured as follows when shipped:

VMEbus Dual Port RAM Range	Disabled
VMEbus Remote RAM	Disabled
VMEbus Address Bias	Pass Through
VMEbus Adapter I/O Range	2000 - 201F (hex)
VMEbus R-INT Jumpers	CINT to VMEbus IRQ1
VMEbus T-INT Jumpers	None
VMEbus Grant/Request Level	Bus Requester on Level 3

VMEbus card NOT driving VMEbus SYSCLK
VMEbus card NOT driving VMEbus SYSRESET
VMEbus card NOT driving VMEbus timeout (BERR)

10.3 VMEbus Adapter Card Jumper Blocks

This section describes the jumper blocks on the VMEbus adapter card.

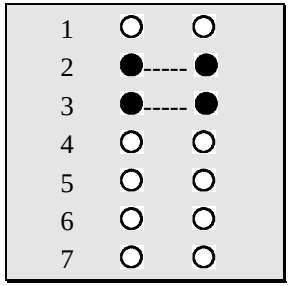
➔ Refer to the diagram in section 10.1.1 for jumper block locations.

➔ If the jumper is IN, the bit is "0". If the jumper is OUT, the bit is "1".

10.3.1 System Jumpers

➔ If the VMEbus adapter card is to be system controller, it must be installed in slot 1 of the chassis and have jumpers 4, 5, and 6 in the SYS block installed.

➔ Jumper IN: ●-------● Jumper OUT: ○ ○



SYS
(factory setting)

Jumper is always OUT.
This card is a "transmitter" when jumper is IN.
Jumper is always IN.
Jumper if this card will drive VMEbus SYSCLK.
Jumper if this card will drive VMEbus SYSRESET.
Jumper if this card will detect bus timeouts.
Jumper is always OUT.

TRANSMITTER CARD SELECTION (jumper 2): This jumper is IN when the VMEbus adapter card requires the option to send commands to the PCI adapter card (be a cable transmitter). It does not exclude the PCI adapter card from also being a cable transmitter. See section 3.3 for information on transmitters and receivers.

If jumper 2 is not installed, the VMEbus adapter card will generate bus errors on any access to the PCI adapter card (remote registers, remote RAM, etc.).

VMEbus SYSCLK DRIVE (jumper 4): Allows the adapter card to supply the VMEbus SYSCLK signal to its VMEbus backplane. This jumper should only be installed when the VMEbus adapter card is the system controller.

VMEbus SYSRESET DRIVE (jumper 5): Allows the adapter card to supply VMEbus SYSRESET to its VMEbus backplane, which may be either a VMEbus adapter card power-on reset or a programmed reset from the PCI bus. When the VMEbus adapter card is the system controller, it is usually jumpered to drive SYSRESET.

VMEbus TIMEOUT (jumper 6): Allows the adapter card to drive the VMEbus BERR (bus error) signal to its VMEbus backplane if any transfer on the bus exceeds 48 μsec (bus timeout). When the VMEbus adapter card is the system controller, it is usually jumpered to detect bus timeout (jumper IN).

10.3.2 Bus Grant And Bus Request Jumpers

The Bus Grant (BGO-BGI) jumper block and the Bus Request (BR) jumper block establish the adapter card Bus Grant and Request levels.

If the VMEbus system has a system controller, the VMEbus adapter card can be configured to use any of the four Bus Request levels. If there is no system controller in the VMEbus system, the adapter can be set to become a single-level arbiter on level three or a four-level arbiter in priority or round-robin mode.

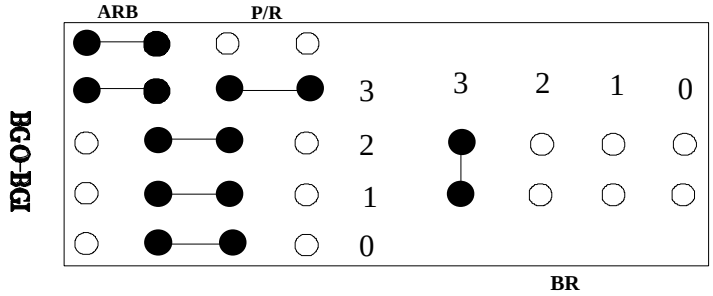
- The ARB jumper enables the arbitration feature of the adapter card to drive the four bus grant lines (BGxIN) and bus clear (BCLR).

- The P/R jumper selects the arbitration mode. When the jumper is OUT, priority arbitration is selected. When the jumper is IN, round-robin arbitration is selected.

- If the adapter card is to be system controller, it must be installed in slot 1 of the VMEbus chassis. Jumpers 4 (SYSCLK: Drive) and 5 (SYSRESET: Drive) in the SYS block should be installed. Jumper 6 (TIMEOUT) should also be installed unless another module provides TIMEOUT. See also SYS block diagram and description of jumpers in section 10.3.1.

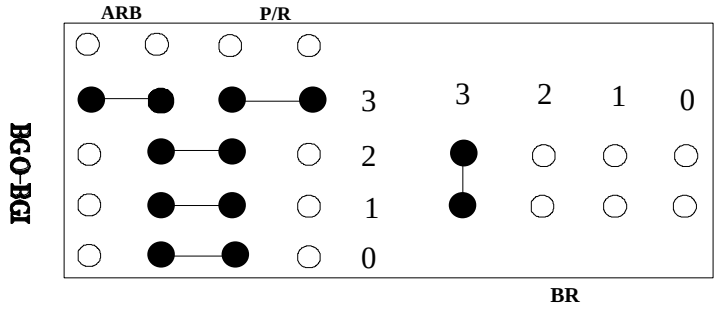
If the VMEbus adapter card *is to be the system controller*, configure the jumpers as follows with priority arbitration selected:

➔ Jumper IN: ●-------● Jumper OUT: ○ ○

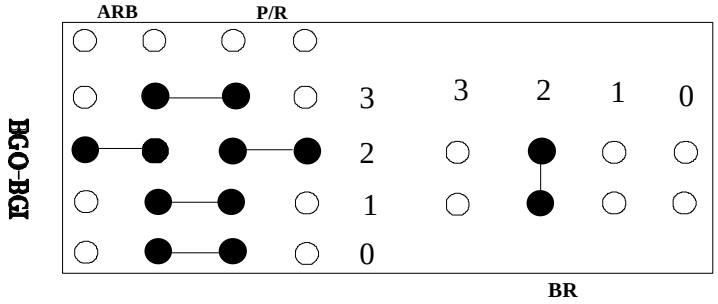


The following diagrams summarize the four configuration options for the VMEbus adapter card when the VMEbus chassis *is not the system controller* (the card may be installed in any slot except slot 1).

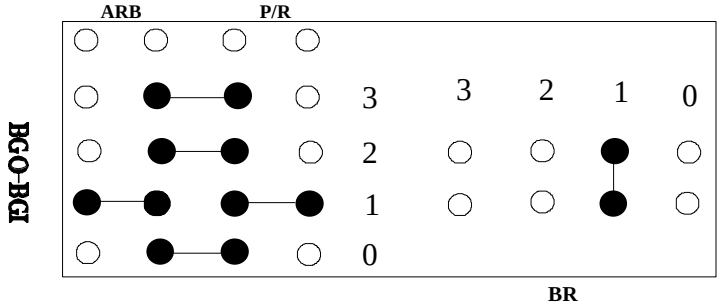
➔ In all four cases, the Bus Request level always matches the Bus Grant Daisy Chain level.



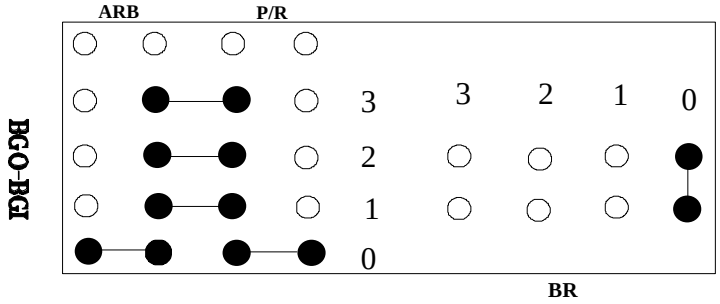
Bus Request 3
(shipping configuration)



Bus Request 2



Bus Request 1



Bus Request 0

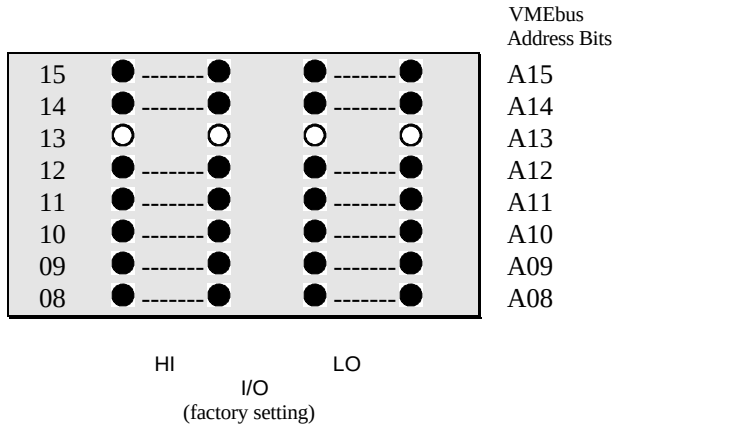
10.3.3 I/O Range Jumpers

The I/O jumper block sets the range of the VMEbus adapter card registers in VMEbus I/O space.

The VMEbus adapter uses 32 bytes of I/O space. The first 16 bytes are for miscellaneous control registers -- eight for the local and eight for the remote adapter card. The next 16 bytes are used for DMA Control and Status Registers -- eight for local and eight for remote DMA registers.

➔ Always set the I/O LO and I/O HI jumpers to the same setting.

➔ Jumper IN: ●-------● Jumper OUT: ○ ○



The table below translates the jumper settings into hex digits.

A15 A11	A14 A10	A13 A09	A12: A08:	VME I/O ADDR FIRST HEX DIGIT VME I/O ADDR SECOND HEX DIGIT
IN	IN	IN	IN	0
IN	IN	IN	OUT	1
IN	IN	OUT	IN	2
IN	IN	OUT	OUT	3
IN	OUT	IN	IN	4
IN	OUT	IN	OUT	5
IN	OUT	OUT	IN	6
IN	OUT	OUT	OUT	7
OUT	IN	IN	IN	8
OUT	IN	IN	OUT	9
OUT	IN	OUT	IN	A
OUT	IN	OUT	OUT	B
OUT	OUT	IN	IN	C
OUT	OUT	IN	OUT	D
OUT	OUT	OUT	IN	E
OUT	OUT	OUT	OUT	F

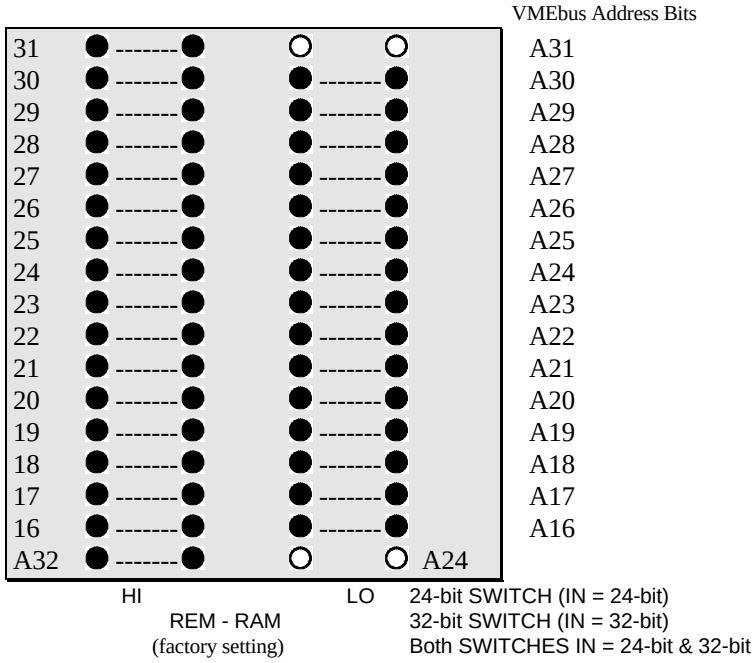
adapter I/O range logic responds to short I/O address modifiers 29 and 2D.

10.3.4 Remote RAM Jumpers

The REM-RAM HI and LO jumpers select the starting address and the address range that a VMEbus master references in VMEbus address space to read from or write to memory in the remote (PCI) chassis. The REM-RAM LO jumpers set the starting VMEbus address and the REM-RAM HI jumpers set the ending address of the memory window. The Remote RAM Window can be described as YYYY0000 - ZZZZ0000 where YYYY is specified by the REM-RAM LO jumpers and ZZZZ is specified by the REM-RAM HI jumpers. Two additional jumpers (labeled A32 and A24) select whether the card uses 32-bit address space or 24-bit address space.

If the A32 jumper is installed, the window is decoded in VMEbus A32 address space. If the A24 jumper is installed, the window is decoded in A24 address space (bits 24 - 31 are ignored). If both jumpers are installed, the window is present in both A32 and A24 address space.

- There cannot be any other VMEbus device or Dual Port RAM at the addresses assigned for REM-RAM, because there would be two memories addressed at the same time; neither would work properly.
- The Remote RAM Window must start at an address that is an integer multiple of the selected remote RAM size. Thus, for a 128K byte remote RAM, the window must start on a 128K byte address.
- To disable the A32 or A24 Remote RAM Window, remove the appropriate A32 or A24 jumper.
- The smallest Remote RAM Window is 64K bytes. Do not exceed 16M bytes of remote RAM.
- We recommend that the Remote RAM Window always be placed on a 16M byte boundary. As a result, the first VMEbus-to-PCI Mapping Register will correspond to the start of the Remote Memory Window.
- Jumper IN: ●-------● Jumper OUT: ○ ○



In 32-bit mode, adapter remote RAM responds to address modifiers 09, 0A, 0B, 0D, 0E, and 0F. In 24-bit mode, adapter remote RAM responds to address modifiers 39, 3A, 3B, 3D, 3E, and 3F.

The VMEbus can perform 8-bit, 16-bit, or 32-bit random access or 16- or 32-bit Block Mode transfers to remote RAM.

The table below translates the jumper settings into hex digits.

A31 A27 A23 A19	**A30 A26 A22 A18**	**A29 A25 A21 A17**	**A28: A24: A20: A16:**	**FIRST HEX DIGIT SECOND HEX DIGIT THIRD HEX DIGIT FOURTH HEX DIGIT**
IN	IN	IN	IN	0
IN	IN	IN	OUT	1
IN	IN	OUT	IN	2
IN	IN	OUT	OUT	3
IN	OUT	IN	IN	4
IN	OUT	IN	OUT	5
IN	OUT	OUT	IN	6
IN	OUT	OUT	OUT	7
OUT	IN	IN	IN	8
OUT	IN	IN	OUT	9
OUT	IN	OUT	IN	A
OUT	IN	OUT	OUT	B
OUT	OUT	IN	IN	C
OUT	OUT	IN	OUT	D
OUT	OUT	OUT	IN	E
OUT	OUT	OUT	OUT	F

VMEbus REM-RAM ADDRESS

10.3.5 Dual Port RAM Jumpers

The Dual-Port HI and LO jumpers select the address range a bus master on the VMEbus references to read or write to Dual Port RAM. The Dual-Port LO jumper block sets the starting VMEbus address and the Dual-Port HI selects the ending address. The Dual Port RAM Window can be described as YYYY0000 - ZZZZ0000 where YYYY is specified by the Dual-Port LO jumpers and ZZZZ is specified by the Dual-Port HI jumpers. Two additional jumpers (labeled A32 and A24) select whether the card responds to 32-bit or 24-bit address space.

If the A32 jumper is installed, the window is decoded in VMEbus A32 address space. If the A24 jumper is installed, the window is decoded in A24 address space (bits 24 - 31 are ignored). If both jumpers are installed, the window is present in both A32 and A24 address space.

The minimum address range is 64K bytes and the maximum is 4G bytes. Set the range to match the size of your Dual Port RAM. If the Dual Port RAM is less than 64K bytes, set the range to 64K bytes.

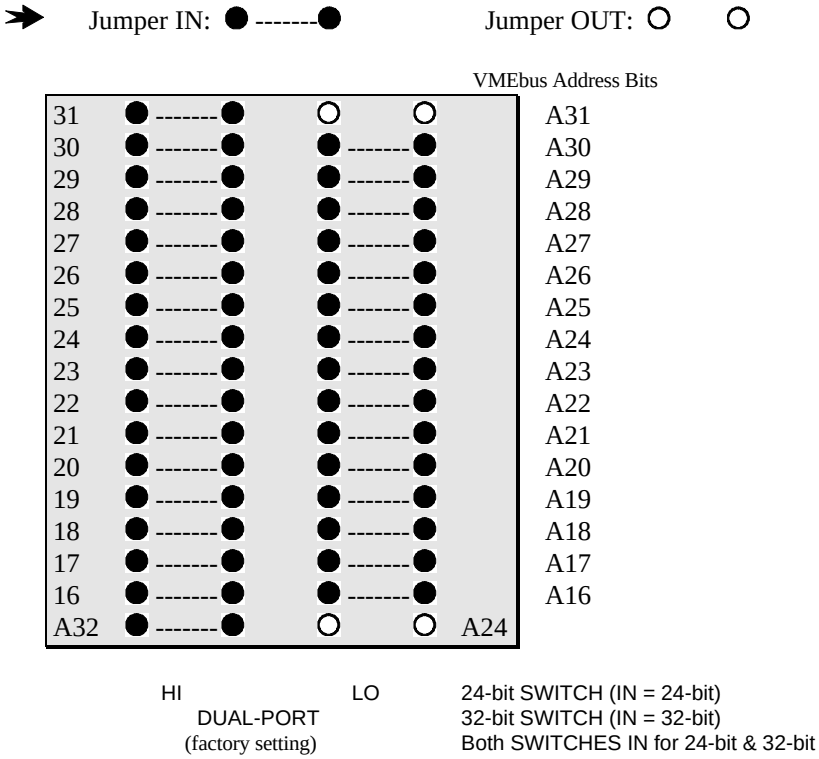
➔ Make sure no other VMEbus device or remote RAM are at the addresses assigned for Dual Port RAM; if two memories are addressed at the same time, neither can work properly.

➔ If Dual Port RAM is not installed or to disable Dual Port RAM, remove the A32 and A24 jumpers.

➔ For any size Dual Port RAM, the Dual Port RAM Window must start at an address that is an integer multiple of the selected Dual Port RAM size. Thus, for a 128K byte Dual Port RAM, the window must start on a 128K byte address in the address space.

In 32-bit mode, Dual Port RAM responds to address modifiers 09, 0A, 0B, 0D, 0E, and 0F. In 24-bit mode, Dual Port RAM responds to address modifiers 39, 3A, 3B, 3D, 3E, and 3F.

The Dual Port RAM module can respond to 8-bit, 16-bit, and 32-bit data transfers from the VMEbus. Dual Port RAM responds to 16-bit and 32-bit block transfers.



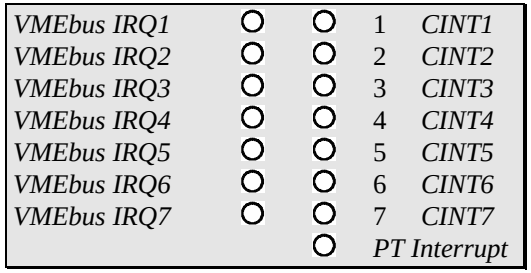
The table below translates the jumper settings into hex digits.

A31 A27 A23 A19	A30 A26 A22 A18	A29 A25 A21 A17	A28: A24: A20: A16:	FIRST HEX DIGIT SECOND HEX DIGIT THIRD HEX DIGIT FOURTH HEX DIGIT
IN	IN	IN	IN	0
IN	IN	IN	OUT	1
IN	IN	OUT	IN	2
IN	IN	OUT	OUT	3
IN	OUT	IN	IN	4
IN	OUT	IN	OUT	5
IN	OUT	OUT	IN	6
IN	OUT	OUT	OUT	7
OUT	IN	IN	IN	8
OUT	IN	IN	OUT	9
OUT	IN	OUT	IN	A
OUT	IN	OUT	OUT	B
OUT	OUT	IN	IN	C
OUT	OUT	IN	OUT	D
OUT	OUT	OUT	IN	E
OUT	OUT	OUT	OUT	F

VMEbus DUAL PORT ADDRESS

10.3.6 Transmitted Interrupt Jumpers

The T-INT jumper block is shown here with no jumpers enabled:



T-INT
(factory setting)

Seven Cable Lines (*CINTx*) can carry interrupts to the PCI bus.

Seven lines (*IRQx*) connect to the VMEbus backplane interrupts.

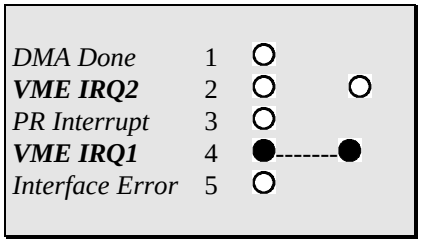
T-INT jumpers select which VMEbus interrupts are transmitted to the PCI adapter card. Any of the seven VMEbus interrupt levels can be sent to the PCI bus by connecting the VMEbus IRQ pin to the corresponding CINTx pin.

The PT Interrupt pin must be connected to a CINTx pin before a VMEbus processor can send a PT Interrupt to the PCI bus. However, a CINTx pin cannot be connected to the PT Interrupt pin and a VMEbus IRQ pin. For example, the PT Interrupt and VMEbus IRQ7 cannot both be connected to the CINT7 pin. Also, if the PCI adapter card is using one CINT line to pass its PT Interrupt to the VMEbus, that CINT line may not be jumpered on the T-INT jumper block.

10.3.7 Received Interrupt Jumpers

The R-INT jumper block is shown here as the card is shipped:

➔ Jumper IN: ●-------● Jumper OUT: ○ ○



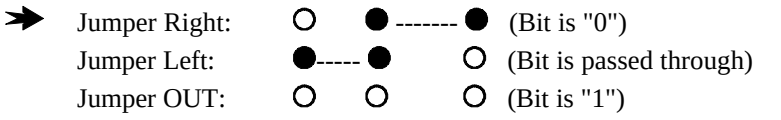
R-INT
(factory setting)

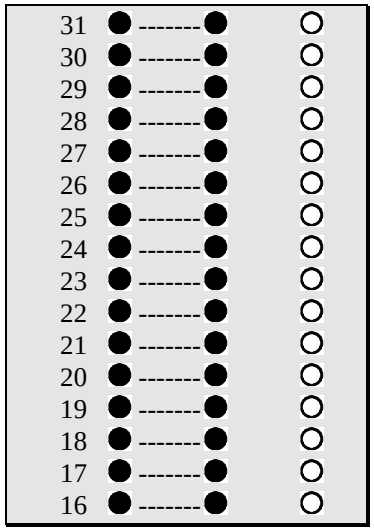
The R-INT jumper block allows the VMEbus adapter card to generate interrupts on the VMEbus backplane. There are five possible interrupt sources on the VMEbus adapter card: DMA Done, cable interrupt 1 from PCI, cable interrupt 2 from PCI, status error, and PT Interrupt. These five sources can be connected to the VMEbus IRQ1 and VMEbus IRQ2 pins in any possible combination. For example, the status error and DMA Done Interrupt pins can be connected to VMEbus IRQ1, while the PR and CINT2 interrupt pins could be connected to VMEbus IRQ2.

PCI has only one interrupt source to send to the VMEbus adapter card, the PT Interrupt. The PT Interrupt from the PCI can be placed on any of the seven CINTx lines using the PCI's Local Interrupt Control Register. If the PCI sends the PT Interrupt over on CINT1 and CINT2, that pin must be jumpered to VMEbus IRQ1 or VMEbus IRQ2.

10.3.8 Address Bias Jumpers

➔ The Address Bias function is not needed on the Model 617 adapter. The jumpers must always be set to Pass Through Mode (the factory configuration).





Passes A31 to the VMEbus
Passes A30 to the VMEbus
Passes A29 to the VMEbus
Passes A28 to the VMEbus
Passes A27 to the VMEbus
Passes A26 to the VMEbus
Passes A25 to the VMEbus
Passes A24 to the VMEbus
Passes A23 to the VMEbus
Passes A22 to the VMEbus
Passes A21 to the VMEbus
Passes A20 to the VMEbus
Passes A19 to the VMEbus
Passes A18 to the VMEbus
Passes A17 to the VMEbus
Passes A16 to the VMEbus

Bias
(factory setting)

10.4 Setting Jumpers For System Controller Mode

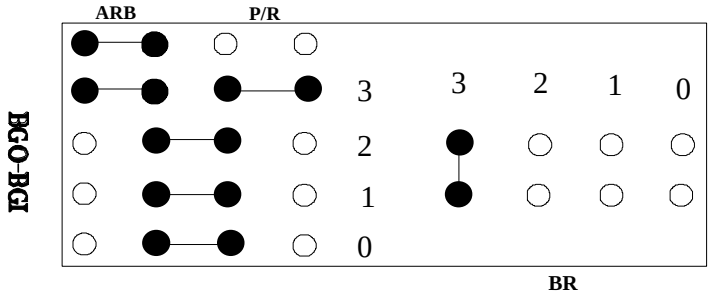
The VMEbus adapter card can be installed in slot 1 and can function as the VMEbus system controller. If the VMEbus adapter card is installed in any other slot, it functions as a normal bus master on the VMEbus. Several jumpers must be installed or removed before the VMEbus adapter card can be moved in or out of slot 1.

Before installing the VMEbus adapter card in slot 1, the following jumpers must be installed:

JUMPER BLOCK	INSTALL JUMPER	FUNCTION
SYS	4	Causes the VMEbus adapter card to drive SYSCLK
SYS	5	Causes the VMEbus adapter card to drive SYSRESET on power up and when the adapter card is reset by the PCI system
SYS	6	Causes the VMEbus adapter card to detect bus timeouts and assert the bus error signal
BGO-BGI	Configure as shown in diagram below	Set request level for the system controller card. Jumpers set to match request level and pass bus grants on unused levels
P/R ARB	Configure as shown in diagram below	Cause the VMEbus adapter card to perform bus arbitration functions and select the arbitration mode

BGO-BGI, P/R and ARB jumper block configuration for System Controller Mode:

➔ Jumper IN: ●-------● Jumper OUT: ○ ○



After removing the VMEbus adapter card from slot 1 and before installing it in any other slot, remove the following jumpers:

JUMPER BLOCK	REMOVE JUMPER	FUNCTION
SYS	4	The VMEbus adapter card will *not* drive SYSCLK
SYS	5	The VMEbus adapter card will *not* drive SYSRESET
SYS	6	The VMEbus adapter card will *not* detect bus timeouts
BGO-BGI	Refer to diagrams in section 10.3.2 for Bus Request 3 - Bus Request 0	Set the request level for the new system controller card
ARB	ARB	The VMEbus adapter card will *not* drive the four bus grant lines (BGxIN) and bus clear (BCLR)

Chapter 11: Common Problems

11.0 Introduction

Many problems encountered by users are related to how the adapter is configured and/or programmed. Chapter 11 includes solutions for the most common problems SBS Bit 3 customers experience when setting up and using the Model 617 adapter. If your problem is not discussed in this chapter or the solution does not work in your environment, call SBS Bit 3's technical support at 612-905-4700.

11.1 Software Problems

When using the adapter, many of the problems encountered are related to adapter configuration and programming. This section deals with the common problems that can occur when either adapter card is set up or programmed improperly.

11.1.1 Data Order Is Incorrect

If the byte order is jumbled when data are transferred between the PCI bus and VMEbus, change the byte swapping bits in the appropriate Mapping Register.

➔ See section 3.5 for a detailed discussion on byte swapping.

➔ PCI accesses to VMEbus memory are controlled by the PCI to VMEbus Mapping Registers.

➔ VMEbus accesses to PCI memory are controlled by the VMEbus to PCI Mapping Registers.

➔ DMA transfers between the PCI bus and VMEbus are controlled by the DMA-to-PCI Mapping Registers.

11.1.2 Dual Port RAM Alignment

If applications on each bus can correctly read and write data to Dual Port RAM, but cannot see data written by the other bus, there is a problem with memory alignment.

Often users assume that if a VMEbus processor accesses byte X from the start of its Dual Port RAM Window and a PCI Processor accesses byte X from the start of its Dual Port RAM Window, then both processors will access the same byte in Dual Port RAM. This assumption is not always true because the Dual Port RAM only looks at the bus address of the access and does not consider the Dual Port RAM Window's starting address. For example, if a 1M byte Dual Port RAM is located at 0x180000 to 0x280000 in VMEbus A24 memory space, when a VMEbus processor writes to location 0x180020, the Dual Port RAM uses the first 20 bits of the address to calculate which location of the memory is being address.

In the example above, Dual Port RAM location 0x80020 not 0x20 would be addressed. Problems with memory alignment can occur if the Dual Port RAM Window on the VMEbus starts at an address that is not a multiple of the Dual Port RAM's size. There are two ways to avoid this problem:

- Make sure the Dual-Port jumpers are set so that the starting address of the Dual Port RAM Window is a multiple of the Dual Port RAM's size. The PCI-to-VMEbus Mapping Registers that point at Dual Port RAM should be programmed to start at Dual Port address 0.

- If the Dual Port RAM Window on the VMEbus must reside at a non-multiple of the Dual Port RAM size, the PCI-to-VMEbus Mapping Registers that point at Dual Port RAM should be programmed to start at a Dual Port RAM address equal to the VMEbus address of the start of the Dual Port RAM Window. For example, if a 2M byte Dual Port RAM must be at 0x100000 to 0x300000 in VMEbus space, the PCI-to-VMEbus Mapping Register that points at Dual Port RAM should be programmed with a starting address of 0x100000.

11.1.3 Bus Error Or Unexpected Status ID (Interrupt Vector) Returned When Reading IACK Read Register

The most common cause of this problem is a program running on the PCI that attempts to generate a VMEbus IACK cycle by doing a byte read of I/O LO + 0x0F instead of I/O LO + 0x0E of the IACK Read Register. Reading I/O LO + 0x0F as a byte causes an illegal condition on the VMEbus resulting in a VMEbus error and interface timeout.

Two solutions are available to the programmer:

- Do a byte read of I/O LO + 0x0E to perform the IACK cycle and receive an eight-bit vector.
- Do a word read of I/O LO + 0x0E to perform the IACK cycle and receive a 16-bit vector.

Other causes of bus errors during IACK cycles:

- An interrupt is not asserted for the interrupt level being acknowledged, or no backplane interrupts are being asserted. Check the IACK Address bits in the Remote Status Register accessed by the PCI.
- The IACK daisy-chain lost continuity on the VMEbus backplane. Check the backplane to make sure the IACK daisy-chain jumper is installed for each slot in which no card is installed.
- There are contending drivers on the IACK daisy-chain. Check the VMEbus backplane to make sure the IACK daisy-chain jumper is removed from each slot in which a card is installed.
- More than one bus master is trying to respond to an interrupt level. Make sure that only one VMEbus bus master is able to acknowledge any given VMEbus interrupt level.

11.1.4 Programming Issues

Three programming issues affect adapter use. Two of these issues involve DOS and its available compilers; the other is a general programming issue.

11.1.4.1 Volatile

Most of the adapter functions are accessed through memory mapped hardware. Memory mapped hardware allows access to its functions through the memory space with any instructions that can refer to memory.

Memory mapped hardware is not normal memory and cannot be treated the same way as normal memory. For example, most compilers optimize code to remove all unnecessary memory accesses. This would be incorrect for a memory mapped hardware register in which each access performs some function on the device.

Care must be taken to avoid these optimizations when compiling or assembling. Most computer languages have some method that tells the compiler to turn off all optimization when dealing with this variable. For the C language, this is the *volatile* type modifier and should be used for all variables that refer to the adapter's memory mapped features.

11.1.4.2 Accessing Addresses Above One Megabyte

All adapter windows are configured above the 1M byte DOS limit. Consequently, for DOS programs to use the adapter, they must have a way to access memory above 1M byte. Several methods may be used, including use of obscure DOS BIOS calls and commercially available DOS extensions such as DOS Protected Mode Interface (DPMI).

Refer references listed in section 2.4 for more information on extending DOS.

➔ Additional Reference: Ray Duncan, et. al., *Extending DOS*, Addison-Wesley, Route 128, Reading, MA 01867.

11.1.4.3 Making D32 Accesses

Many compilers available for 80x86 platforms cannot generate 32-bit code; therefore, they cannot generate instructions that use 32-bit memory locations as arguments. These compilers break any 32-bit memory reference into two 16-bit memory references.

The adapter can be used with these compilers unless a VMEbus device is used that can only be accessed by longwords. If producing longword accesses on the VMEbus is important to an application, make sure the compiler to be used generates 32-bit code.

11.2 Hardware Problems

11.2.1 Using The VMEbus Adapter Card LEDs As Diagnostic Tools

Use the REMOTE and LOCAL LEDs in conjunction with the Remote and Local Status Registers to track down the cause of a problem.

- READY: Immediately after power-up, the READY LED should be illuminated on both adapter cards indicating that the on-board programmable gate arrays have been successfully loaded. If either LED is not lit, the adapter will not function. If the VMEbus adapter card is installed in slot 1 and its LED is not lit, the VMEbus chassis will not function.

  If either READY LED is not lit, either the PCI adapter is broken or the PCI or VMEbus power supply may be slow ramping, or does not have a uniform voltage ramp-up.

- REMOTE: When the VMEbus adapter card is processing a command from the PCI adapter card, the REMOTE LED will be illuminated. After the command is processed, the LED will switch off. Therefore, for a single read or write, the length of time the LED will be lit is too short to perceive with the naked eye. If the PCI is placed in a software loop to repeatedly access the VMEbus adapter card, the LED will appear to be lit continuously. Varying degrees of brightness indicate a corresponding rate of activity. If an interface timeout occurs when accessing the VMEbus adapter card, note the state of the REMOTE LED. If it is lit but there is no longer activity on the cable, the command that received the interface timeout did not complete on the VMEbus adapter card.

- LOCAL: When a CPU in the VMEbus chassis accesses the VMEbus adapter card, the LOCAL LED is illuminated. This LED remains lit as long as the address being generated is recognized by the SBS Bit 3 VMEbus adapter card.

11.2.2 Error In The Local Status Register

Any time a local processor accesses resources on the remote adapter card or remote bus status errors can occur. The Local Status Register can be read to determine where an error occurred and the error type. There are five types of status errors:

- Interface parity errors,
- VMEbus bus errors,
- Interface timeout errors,
- DMA LRC errors,
- Remote power off or cable disconnected.

11.2.2.1 Local Status Register Bit 7: Interface Parity Error

Data are sent across the adapter cable with parity and are checked upon arrival at either Model 617 adapter card. If a mismatch occurs, bit 7 is set to "1" indicating that either the actual data or the parity bit may be corrupt. The Interface Parity Error bit will not clear until reset via the Local Command Register reset mechanism.

Interface parity errors occur rarely. If parity errors occur frequently, either the PCI adapter, the PCI motherboard, or the VMEbus chassis has a problem. Make sure you have the grounding strap on the cable attached to the VMEbus chassis.

Common causes of interface parity errors:

- The cable grounding strap is not connected.
- Power is off in the remote chassis, or the cable is disconnected.
- The cable is loose or not fully attached.
- Bent pins in one or both of the cable connectors.
- Multiple SYSCLK generators in the VMEbus chassis.
- A component failed on one or both Model 617 adapter cards.

11.2.2.2 Local Status Register Bit 6: Remote Bus Error

Any remote access that terminates improperly will cause a remote bus error. PCI bus errors occur when the slave device detects an error with the transfer and asserts the target abort signal or no slave responds within six PCI bus cycles. VMEbus errors occur when the slave device detects an error and asserts the BERR signal or no slave responds within 48 to 54 µsec (the de facto standard).

If a remote cycle failed during an access, bit 6 in the Local Status Register is set. It will not clear until reset via the Local Command Register reset mechanism.

Common cause of remote bus errors:

- An incorrect bus address.

Common causes of VMEbus errors:

- An incorrect VMEbus address modifier.
- A remote access performed to a slow slave device that takes longer than 48 - 54 µsec to respond.
- A VMEbus interrupt acknowledge cycle for which no interrupting card responds.
- Bus grant daisy-chain jumpers are not configured properly on the VMEbus chassis.
- The VMEbus adapter card's SYS, BGI-BGO, or BR jumper blocks are not configured correctly for the slot in which the adapter card is installed.
- An access is made to the Dual Port RAM Window when no Dual Port RAM is installed.
- An incorrect data size. For example, an attempt to perform a D16 transfer to a D08 device.
- A read performed from a write-only location, or a write performed to a read-only location.
- A parity error was detected internal to a VMEbus memory card. This typically occurs when the memory location is read before being written (initialized) at least once.

➔ If a PCI bus master starts a remote VMEbus cycle that does not complete within 30 µsec, the PCI adapter card will terminate its local cycle and set its Interface Timeout bit. The VMEbus cycle will continue until either VMEbus slave responds or a bus timeout occurs. If a VMEbus error occurs after 30 µsec, the next PCI remote access may cause the PCI remote bus error bit to be set with the last cycle's bus error.

11.2.2.3 Local Status Register Bit 2: Interface Timeouts

Any access to a remote resource that lasts longer than a predetermined length of time will timeout and set bit 2 of the Local Status Register. For PCI-to-VMEbus accesses, the timeout is 30 µsec. VMEbus-to-PCI accesses timeout in 40 µsec. DMA transfers can never exceed 16 msec. The Timeout bit does not clear until it is reset via the Local Command Register reset mechanism.

Common causes of interface timeouts:

- An access is attempted while the remote chassis is off or the cable is disconnected.
- The same conditions that cause VMEbus errors.
- The Bus Grant daisy chain is broken.
- A VMEbus slave device response was longer than 28 µsec (PCI bus timeout is 30 µsec; SBS Bit 3 remote access cycle time is 2 µsec).
- Multiple system clocks (*SYSCLK*) or no SYSCLK. There must be one and only one SYSCLK in a VMEbus system.
- The VMEbus adapter card is suffering from bus grant starvation. When many bus masters share the same bus grant level, the requesting device closest to the arbiter in the bus grant daisy-chain receives the grant. Devices farther away from the arbiter are not granted the bus as often and may starve.

  Bus grant starvation usually is an intermittent condition, it will not occur on every transfer.

  Solutions for bus grant starvation are:

  - ❒ Make VMEbus devices that do block transfers (BLT) re-arbitrate for the bus more often during DMAs.
  - ❒ Switch to a multi-level grant arbitration scheme or move the Model 617 VMEbus adapter card closer to the bus arbiter in slot 1.
  - ❒ Make sure all VMEbus devices use the VMEbus as fairly as possible.

11.2.2.4 Local Status Register Bit 0: Cable Disconnected

When the remote chassis is powered off or the adapter cable is not connected to one or both of the adapter cards, Local Status Register bit 0 is set. Attempts to make a remote access to Remote Node Registers or to remote memory, will fail with various status errors. The Power Off or Cable Disconnected bit automatically clears when the cable is connected or the remote power is turned on.

11.2.3 PCI Motherboards

The PCI motherboard incorporates several features that allow CPUs to operate faster by shortening the time required to access main memory. Because many of the PCI adapter card's features are accessed through memory mapped windows, the fast access features of the motherboard may interfere with proper adapter operation and may need to be disabled.

11.2.3.1 Cache

A cache is a very high speed memory that resides between the CPU and main memory. It is the most common hardware device used to improve CPU speed.

The cache stores information that the CPU frequently references so that it can be retrieved faster. The memory mapped windows of the adapter must not be cached. The cache should be disabled either globally or just the section of memory in which the windows reside.

11.2.3.2 PCI Slots

The PCI specification indicates that all slots should be able to support bus mastership. However, some PCI motherboards have slots that only support PCI slave devices. The PCI adapter must be placed in a slot that supports a PCI bus master device. Refer to your specific PCI motherboard manual to find out which PCI slots are capable of bus mastership.

Also, some PCI BIOS require that PCI master devices be placed in the first open PCI master slot; otherwise, the device will not be master enabled.

11.2.3.3 Concurrent Accesses

Because PCI devices can access VMEbus memory and VMEbus masters can access PCI memory, at times both devices may try to access remote memory simultaneously (concurrent accesses).

The adapter can handle concurrent accesses by giving a retry signal to the PCI device requesting use of the adapter. The adapter arbitrates for the PCI bus and completes the action from the VMEbus. The PCI device that was retried then requests use of the adapter again and completes its transaction.

Most PCI motherboards do not support concurrent accesses and will not allow the adapter to complete the VMEbus transaction. A state of livelock may occur where the PCI motherboard and the PCI adapter card are in a constant loop of giving each other a retry response. These problems may appear on any concurrent access or may only appear on a specific concurrent access combination. For example, a VMEbus device is doing a read from PCI memory at the same time the PCI processor is writing a Remote Node Register.

Problems with concurrent accesses normally cause the PCI system to hang or a VMEbus error to occur.

Chapter 12: Utilities Diskette

12.0 Introduction

Chapter 12 describes how to install the example programs that are on the Utilities Diskette and the function of each executable.

Example programs on the Utilities Diskette are written in the C programming language, but are useful even if you do not have a C compiler. The set up routine and utility programs are contained in the C sub-directory.

12.1 Quick Install Procedure

1. Insert the Utilities Diskette in the appropriate diskette drive.

2. Change to the selected drive in DOS.

3. Enter INSTALL X: where X is the drive that the software should be installed on. For example, to install the utilities software from drive B: onto drive C:, switch to drive B and type INSTALL C:.

   The installation program will place the utilities software in a directory under the root called BT809\VX.Y where X.Y is the version of the Model 809 Utilities that you received.

4. Read the file readme.txt for additional information about the utilities and a description of changes made since the last software release.

5. Read the file proginfo.txt for detailed information about the utilities.

12.2 Executable Files

EXECUTABLE FILE	FUNCTION
SETUP.EXE	Activates the PCI adapter card and checks for status errors
DUMPPORT.EXE	Dumps the first 256 addresses of the optional Dual Port RAM
DUMPLRAM.EXE	Dumps 256 bytes of VMEbus RAM
DUMPMAP.EXE	Displays the first 64 (256 bytes) of Mapping Registers
DEMOPDMA.EXE	Reads or writes to/from the VMEbus memory or Dual Port RAM using DMA
BT_ACCESS.EXE	Allows the user to read or write from VMEbus memory or Dual Port RAM. BT_ACCESS is more flexible than either dumpport or dumplram
BT_QCHK.EXE	Performs a quick check to make sure the hardware is functioning correctly
INTHNDLR.EXE	Displays a message when a programmed interrupt was received from the VMEbus

Appendix A: Glossary

The following terms are used throughout this manual:

"0": Zero.

"1": One.

Adapter Node Input/Output: Any access to the I/O registers contained on either the local or remote adapter card. These are referred to as local node I/O and remote node I/O, respectively.

Big Endian: A byte ordering scheme in which the most significant bytes are in the numerically lower addresses. The VMEbus is big endian. See also Little Endian.

Bit: A single digit in a binary number (0 or 1).

Byte: 8 bits.

Clear: "0". Examples: Clear bit 6 of register A; write bit 6 of register A to "0".

CSR: Adapter Control and Status Registers.

Direct Memory Access Transfers (DMA): The adapter may be programmed to transfer large blocks of data across the cable to or from the remote chassis, rather than requiring a processor to move data.

Dual Port RAM: An optional dual-port memory card attached to the VMEbus adapter card.

Exchanging Interrupts: Sending interrupts to and receiving interrupts from the remote chassis; includes any processing an application should do to acknowledge the receipt of an interrupt.

FIFO: First In First Out. A memory device in which data are retrieved in the same sequence as stored.

G byte: Gigabyte. Two to the thirtieth power (exactly 1,073,741,824 bytes).

Hex: Hexadecimal notation. A numbering system that uses 16 digits (0123456789ABCDEF) to denote a number.

I/O Space: A special address space used to access and control hardware devices. Often special processor instructions are used to generate read and write cycles in the I/O space.

ISR: Interrupt Service Routine. A software routine that services a device that has asserted an interrupt.

K byte: Kilobyte. Two to the tenth power (exactly 1024) bytes.

Linear Address: A processor level address. Used by the processor to reference memory locations on its external bus. Often translated by an external memory management unit into a physical or bus address. See also Virtual Address and Physical Address.

Little Endian: A byte ordering scheme in which the least significant bytes are in the numerically low addresses. The PCI bus is little endian. See also Big Endian.

Local: Indicates that the resource is on this bus and does not require use of the adapter interface cable to access it.

Longword: 32 bits; in the PCI specification, 32-bit data are called words.

M byte: Megabyte. Two to the twentieth power (exactly 1,048,576) bytes.

M Bytes/sec: Megabytes per second. Exactly 1,000,000 bytes per second.

Mapping RAM: Steers accesses in 4K byte segments from PCI address space to VMEbus address space, and from VMEbus onto the PCI bus. There are 8,192 32-bit Mapping Registers available for mapping from PCI address space to VMEbus address space; 4,096 32-bit Mapping Registers are available to map accesses from VMEbus master onto the PCI bus; and there are 4,096 32-bit DMA to PCI bus Mapping RAM Registers.

Memory Mapped Device: A hardware device that allows access to its functionality through memory space. Normal memory instructions can be used to control the device and access its features.

msec: Millisecond. 1/1,000 of a second.

nsec: Nanosecond. 1/1,000,000,000 of a second.

PCI: Peripheral Component Interconnect.

Physical Address: The address that is presented to the bus to reference memory locations.

PIO: Programmed I/O; also referred to as random access.

Programmed Interrupts: Interrupts that can be used by applications to synchronize communications between the two buses. The two types of programmed interrupts are the PT (Programmed to Transmitter) interrupt and the PR (Programmed to Receiver) interrupt.

Receiver: An adapter card that is not allowed to transmit messages across the interface cable. This prevents the card from accessing the remote node I/O registers, remote bus I/O, and remote bus memory, or a remotely-installed Dual Port RAM card.

Remote: Indicates that the resource is on the other bus and requires use of the adapter interface cable to access it.

Remote Bus I/O: Any access to the I/O address space which is located in the remote system chassis.

Remote Bus Memory: Any access to the memory space in the remote chassis. This may be a shared memory section, a device buffer, or any device that responds to a memory access. It does not include Dual Port RAM located on the remote adapter card.

Set: "1". Example: Set bit 4 to a "1".

Transmitter: An adapter card that is allowed to transmit messages across the interface cable. Every pair of adapter cards must have at least one transmitter.

µsec: Microsecond. 1/1,000,000 of a second.

Virtual Address: A software level address. The virtual address is used by applications to reference memory locations and is often translated into a linear address by the processor's internal memory management unit.

Virtual Memory: A facility whereby the effective range of addressable memory locations provided to a process is independent of the size of main memory.

Window: A range of addresses that the adapter responds to for a specific function; a reserved area of memory.

Word: 16 bits; in the PCI specification, 16-bit data are called halfwords.

Appendix B: VMEbus References

B.1 VMEbus Pin Assignments

P1/JI			
Pin #	**Row A**	**Row B**	**Row C**
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK	A17
22	IACKOUT*	SERDAT*	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12 VDC	+5VSTDBY	+12 VDC
32	+5 VDC	+5 VDC	+5 VDC

* = active low

P2/J2			
Pin #	**Row A**	**Row B**	**Row C**
1	User Defined	+5 VDC	User Defined
2	User Defined	GND	User Defined
3	User Defined	RESERVED	User Defined
4	User Defined	A24	User Defined
5	User Defined	A25	User Defined
6	User Defined	A26	User Defined
7	User Defined	A27	User Defined
8	User Defined	A28	User Defined
9	User Defined	A29	User Defined
10	User Defined	A30	User Defined
11	User Defined	A31	User Defined
12	User Defined	GND	User Defined
13	User Defined	+5 VDC	User Defined
14	User Defined	D16	User Defined
15	User Defined	D17	User Defined
16	User Defined	D18	User Defined
17	User Defined	D19	User Defined
18	User Defined	D20	User Defined
19	User Defined	D21	User Defined
20	User Defined	D22	User Defined
21	User Defined	D23	User Defined
22	User Defined	GND	User Defined
23	User Defined	D24	User Defined
24	User Defined	D25	User Defined
25	User Defined	D26	User Defined
26	User Defined	D27	User Defined
27	User Defined	D28	User Defined
28	User Defined	D29	User Defined
29	User Defined	D30	User Defined
30	User Defined	D31	User Defined
31	User Defined	GND	User Defined
32	User Defined	+5 VDC	User Defined

B.2 VMEbus Address Modifier Codes

ADDRESS MODIFIER (HEX)	# OF ADDRESS BITS	TRANSFER TYPE
3F	24	Standard supervisory block transfer
3E	24	Standard supervisory program access
3D	24	Standard supervisory data access
3B	24	Standard non-privileged block transfer
3A	24	Standard non-privileged program access
39	24	Standard non-privileged data access
2D	16	Short supervisory access
29	16	Short non-privileged access
10 - 1F	undefined	User defined
0F	32	Extended supervisory block transfer
0E	32	Extended supervisory program access
0D	32	Extended supervisory data access
0B	32	Extended non-privileged block transfer
0A	32	Extended non-privileged program access
09	32	Extended non-privileged data access
don't care state	3	IACK cycle (uses A01-A03)

Appendix C: PCI BIOS Functions

The following functions are used on MS-DOS PCI systems to determine if specific devices are installed and to access their Configuration Registers.

C.1 Identifying PCI Resources

The following group of functions allow the caller to determine if the PCI BIOS support is installed, and if specific PCI devices are present in the system.

C.1.1 PCI BIOS Present

This function allows the caller to determine if the PCI BIOS interface function set is present, and the current interface version level.

ENTRY:

[AH] 0xB1
[AL] 0x01

EXIT:

[EDX] "PCI", "P" in [DL], "C" in [DH], etc. There is a 'space' character in the upper byte
[AH] Present Status, 0x00 = BIOS present if, and only if, EDX and CF are set properly
[BH] Interface Level Major Version
[BL] Interface Level Minor Version
[CL] Number of last PCI bus in the system
[CF] Present Status, set = No BIOS present,
reset = BIOS present if, and only if, EDX and AH are set properly

➔ If the CARRY FLAG [CF] is cleared and AH is set to 0x00, it is still necessary to examine the contents of [EDX] for the presence of the string "PCI" + (trailing space) to fully validate the presence of the PCI function set.

C.1.2 Find PCI Device

This function returns the location of PCI devices that have a specific device ID and Vendor ID. Given a Vendor ID, Device ID and an Index (N), the function returns the Bus Number, Device Number, and Function Number of the Nth Device/Function whose Vendor ID and Device ID match the input parameters.

ENTRY:

[AH]	0xB1
[AL]	0x02
[CX]	Device ID (0. . . 65535)
[DX]	Vendor ID (0. . . 65534)
[SI]	Index (0. . . N)

EXIT:

[BH]	Bus Number (0. . . 255)
[BL]	Device Number in upper five bits, Function Number in bottom three bits
[AH]	Return Code: 0x0 - successful 0x86 - device not found 0x83 - bad vendor id
[CF]	Completion Status, set = error, cleared = success

Calling software can find all devices having the same Vendor ID and Device ID by making successive calls to this function starting with Index set to zero, and incrementing it until the return code is 'device not found'.

C.2 Accessing Configuration Space

C.2.1 Read Configuration Byte

This function allows reading individual bytes from the configuration space of a specific device. The specific device is indicated together with the offset of the byte to be read.

ENTRY:

[AH]	0xB1
[AL]	0x08
[BH]	Bus Number (0. . . 255)
[BL]	Device Number in upper five bits, Function Number in lower three bits
[DI]	Register Number (0. . . 255)

EXIT:

[CL]	Byte read
[AH]	Return Code: 0x0 - successful 0x87 - bad register number
[CF]	Completion Status, set = error, cleared = success

➔ The Bus Number, Device Number, and Function Number are found by calling the Find PCI Device BIOS function with the Device and Vendor ID of the Model 617 adapter.

C.2.2 Read Configuration Word

This function allows reading individual words from the configuration space of a specific device. The Register Number parameter must be a multiple of two. The specific PCI device is indicated together with the offset of the word to be read.

ENTRY:

[AH]	0xB1
[AL]	0x09
[BH]	Bus Number (0. . . 255)
[BL]	Device Number in upper five bits, Function Number in lower three bits
[DI]	Register Number (0, 2, 4,. . . 254)

EXIT:

[CX]	Word read
[AH]	Return Code: 0x0 - successful 0x87 - bad register number
[CF]	Completion Status, set = error, cleared = success

➔ The Bus Number, Device Number, and Function Number are found by calling the Find PCI Device BIOS function with the Device and Vendor ID of the Model 617 adapter.

C.2.3 Read Configuration Longword

This function allows reading individual longwords from the configuration space of a specific device. The Register Number parameter must be a multiple of four. The specific PCI device is indicated together with the offset of the longword to be read.

ENTRY:

[AH]	0xB1
[AL]	0x0A
[BH]	Bus Number (0. . . 255)
[BL]	Device Number in upper five bits, Function Number in lower three bits
[DI]	Register Number (0, 4, 8,. . . 255)

EXIT:

[ECX]	Longword read
[AH]	Return Code: 0x0 - successful 0x87 - bad register number
[CF]	Completion Status, set = error, cleared = success

➔ The Bus Number, Device Number, and Function Number are found by calling the Find PCI Device BIOS function with the Device and Vendor ID of the Model 617 adapter.

C.2.4 Write Configuration Word

This function allows writing individual words to the configuration space of a specific device. The offset must be a multiple of two. The specific PCI device is indicated together with the offset and byte data to be written.

ENTRY:

[AH]	0xB1
[AL]	0x0C
[BH]	Bus Number (0. . . 255)
[BL]	Device Number in upper five bits, Function Number in lower three bits
[DI]	Register Number (0, 2, 4, . . . 254)
[CX]	Byte Value to Write

EXIT:

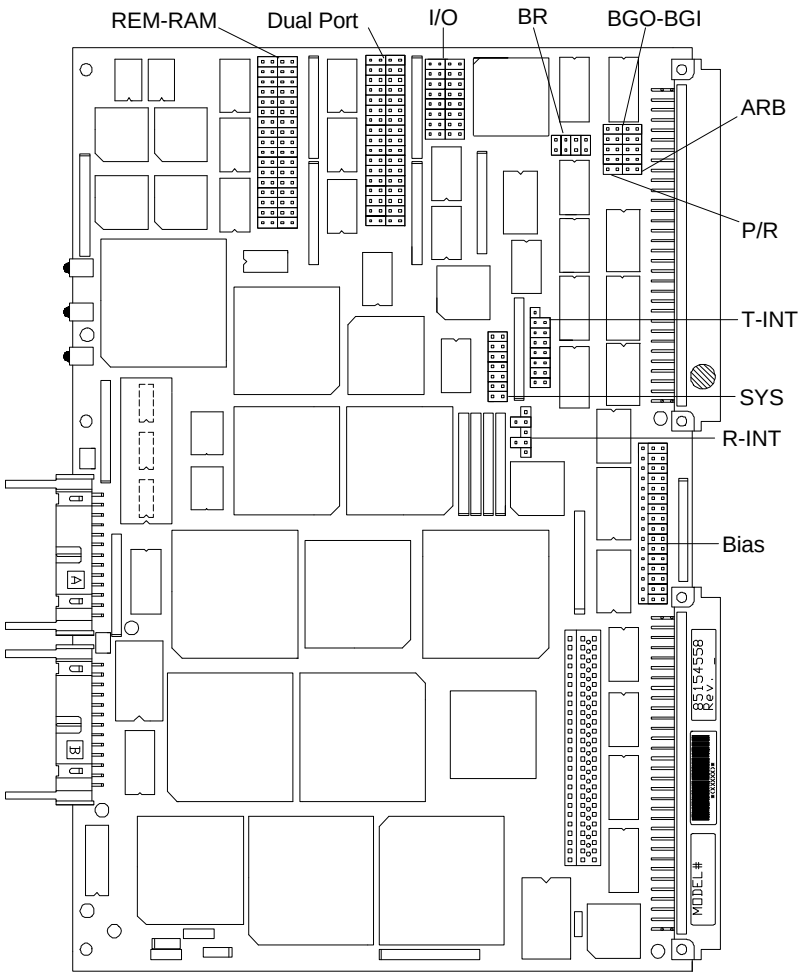
[AH]	Return Code: 0x0 - successful 0x87 - bad register number
[CF]	Completion Status, set = error, cleared = success

➔ The Bus Number, Device Number, and Function Number are found by calling the Find PCI Device BIOS function with the Device and Vendor ID of the Model 617 adapter.

Appendix D: Jumper Configuration Worksheet

Use the following worksheet to document how you have set the jumpers on your Model 617 VMEbus adapter card.

Please have the completed worksheet available when calling SBS Bit 3 for technical support.



Index

C

E

M

S