GE Fanuc Automation

Programmable Control Products

GE Fanuc's Tundra Universe II™ Based VMEbus Interface

User's Manual

GFK-2122 514-000212-000 April 2002

GFL-002

Warnings, Cautions, and Notes as Used in this Publication

Warning

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

Caution

Caution notices are used where equipment might be damaged if care is not taken.

Note

Notes merely call attention to information that is especially significant to understanding and operating the equipment.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware or software, nor to provide for every possible contingency in connection with installation, operation, or maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc Automation assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc Automation makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

The following are trademarks of GE Fanuc Automation North America, Inc.

Alarm Master
CIMPLICITY
CIMPLICITY 90–ADS
CIMSTAR
Field Control
GEnet
Genius
Helpmate
Logicmaster
Modelmaster
Motion Mate
ProLoop
PROMACRO
PowerMotion
PowerTRAC
Series 90
Series Five
Series One
Series Six
Series Three
VersaMax
VersaPro
VuMaster
Workmaster

Contents

Contents

Chapter 1 General Information

This manual describes the installation and operation of GE Fanuc's PCI-to-VMEbus interface that is built around the Tundra Universe II interface chip (CA91C142).

Reference Material and Other GE Fanuc Manuals

For a detailed explanation of the VMEbus and its characteristics, "The VMEbus Specification" is available from:

VITA
VMEbus International Trade Association
7825 East Gelding Dr., No. 104
Scottsdale, AZ 85260
(480) 951-8866
FAX: (480) 951-0720
Internet: www.vita.com

The following Application and Configuration Guides are available from GE Fanuc to assist in the selection, specification, and implementation of systems based upon GE Fanuc's products:

Analog I/O Products (Built-in-Test) Configuration Guide (catalog number GFK-2084)	Provides assistance in configuring analog I/O subsystems based on GE Fanuc's analog I/O products, including common designs, which offer a wide variety of solutions.
Connector and I/O Cable Application Guide (catalog number GFK-2085)	Describes I/O connections that can be used with GE Fanuc's VMEbus products. Includes connector compatibility information and examples.

General Description

The PCI-to-VMEbus interface is built around the Tundra Universe II interface chip (CA91C142) and provides the following features:

- 33 MHz PCI local bus interface with five separate PCI-to-VMEbus slave images (windows)
- High-performance 64-bit (multiplexed) VMEbus interface with four separate VMEbus-to-PCI slave images (windows)
- Programmable DMA controller with linked list support
- VMEbus system controller functionality
- Automatic slot 1 controller detection
- PCI and VMEbus interrupt generation
- VMEbus interrupt handler
- Communication support via four 32-bit mailbox registers with interrupt capabilities
- Real-time OS support with eight semaphores
- Master/Slave endian conversion hardware (Patent Pending)
- Non slot 1 bus timeout timer
- Auxiliary BERR VME address and Address Modifier capture with interrupt

Since the interface board is built around the Tundra Universe II chip, the majority of this documentation is extracted from the Universe II manual itself. However, there are features of the Universe II chip that are not supported at the board level, such as a 64-bit PCI interface, JTAG testability, and certain powerup options. These unsupported features are appropriately noted within the documentation.

In addition, auxiliary functionality has been added to provide various features such as master/slave endian conversion and a non slot 1 bus timeout timer.

Programming the VMEbus Interface

The VMEbus Interface is configured by programming the auxiliary function global interrupt mask register, the external system registers, and the Universe II Control and Status Registers. In addition, the board has four jumpers which configure the board for generating/receiving VMEbus reset, asserting SYSFAIL on powerup, and mapping the Universe II registers in memory or I/O space. Refer to Chapter 2, *Functional Description*.

The base address of the various registers are located in either configuration space or memory and can be accessed using PCI BIOS calls, or GE Fanuc control function library software. Refer to Appendix A for a complete description of interface registers.

Note

In order to enable access to the VMEbus, both bits (11,10) must be set to high in the System COMM register located at $D800E in memory.

Tundra Corporation Reprinted Information

The information about the Tundra Universe II product is provided by Tundra Corporation and is reprinted here within with permission. This material is used extensively throughout this document. The only changes made to this text were made for section numbering purposes; GE Fanuc-specific information injected directly into the Tundra text is preceded by the GE Fanuc logo:

All of Chapter 2 is provided by Tundra except for the following GE Fanuc-generated sections: Architectural Overview of the VMEbus Interface on page 41, PCI-to-VMEbus Interface Jumpers on page 43, Linked-List Operation on page 131, FIFO Operation and Bus Ownership on page 137.

Chapters 3, 4, 5, and 6, are exclusively GE Fanuc.

Appendix A contains the following GE Fanuc information:

- The auxiliary function global interrupt mask register,
- The system register section, and
- The mailbox register section

The Universe Control and Status Register material in Appendix B is a direct reprint of the Tundra-provided information.

Appendixes B through G are reprinted information from Tundra.

Benefits of the Universe II

Interfacing the VMEbus with PCI presents a number of opportunities and challenges. The Universe II solves the problems and allows you to benefit from the opportunities.

The opportunities involve merging the best of the VMEbus and PCI bus worlds. The VMEbus is a proven standard specifically designed to support embedded systems. The distributed environment of the VMEbus supports multiprocessing and real-time intensive applications. A large number of off-the-shelf boards, software, and chassis components are available. VMEbus supports 21-slot systems without bridging, and is continually evolving while providing backward compatibility. Hot swap solutions and higher performance protocols have been defined and will be incorporated in future revisions of VMEbus.

Meanwhile, PCI has become a standard local bus. As a result, the leading semiconductor vendors have built PCI support into their newest processor and peripheral families.

The challenges involved in interfacing VMEbus to PCI include: address mapping, byte-lane swapping, and cycle mapping.

To allow VMEbus single-board computer vendors to benefit from PCI components, while overcoming the challenges involved in merging PCI with VMEbus, Tundra has developed a PCI-to-VMEbus interface controller, the Universe II. The Universe II is the industry-proven, high-performance 64-bit VMEbus-to-PCI interface, fully compliant with VME64 and tailored for the new generation of high performance PCI processors and peripherals.

The availability of the Universe II eases the development of multimaster, multiprocessor architectures on VMEbus systems using PCI. The Universe II is ideally suited for CPU boards acting as both master and slave in the VMEbus system and that require access to PCI systems. With the Universe II, you know that as your system increases in complexity, you have silicon that continues to provide everything you need in a bridge. The elegant design of the Tundra Universe II, some of the best applications engineers in the industry, and this manual will make it as easy as possible for you to use the most sophisticated VMEbus interface.

Features

The Universe II (CA91C142) is the *de facto* industry-standard PCI bus-to-VMEbus bridge, providing:

- 64-bit, 33 MHz PCI bus interface
- fully compliant, high-performance 64-bit VMEbus interface
- integral FIFOs buffer multiple transactions in both directions
- programmable DMA controller with linked-list support
- industry leading performance

- wide range of VMEbus address and data transfer modes
  - A32/A24/A16 master and slave, (not A64 or A40)
  - D64/D32/D16/D08 master and slave, (no MD32)
  - MBLT, BLT, ADOH, RMW, LOCK, location monitors
- nine user-programmable slave images on VMEbus and PCI bus ports
- seven interrupt lines on either bus and flexible mapping of software and hardware sources of hardware interrupt
- automatic initialization for slave-only applications
- flexible register set, programmable from both the VMEbus and the PCI bus
- four mailboxes and location monitor for message-oriented system
- support for RMWs, lock cycles, and semaphores guarantee exclusive access
- bus isolation mode for board maintenance, diagnostics, and live fault recovery
- full VMEbus system controller functionality
- several powerup options
- IEEE 1149.1 JTAG testability support
- commercial (0º to 70º C), industrial (-40º to 85º C) and extended temperature (-55º to 125ºC) options
- available in 313-pin Plastic BGA and 324-pin Ceramic BGA.

Past and Future of the Universe

The Universe II (CA91C142) is a pin and software-compatible revision of the Universe (CA91C042). The Universe was developed subsequently to the SCV64, Tundra’s VME interface for non-PCI applications. The Universe II is the next generation of the Universe, and has been designed to exceed new customer expectations and to correct errata in the original Universe. The rich set of feature and performance enhancements are based on extensive consultation with our customers.

The Universe II offers a low-risk, feature-rich, high-performance solution for VMEbus-based PCI applications. Some of the performance enhancements offered by the Universe II include:

- improved PCI bus bandwidth utilization
- improved register access performance
- improved VMEbus slave and VME master performance
- increased FIFO depth
- DMA improved to optimize transfer rate on each bus
- improved linked-list DMA performance
- significantly improved coupled transfer performance
- reduced power consumption

Additional features include:

- four mailbox registers
- eight semaphores
- four location monitors
- new software interrupts
- more slave images

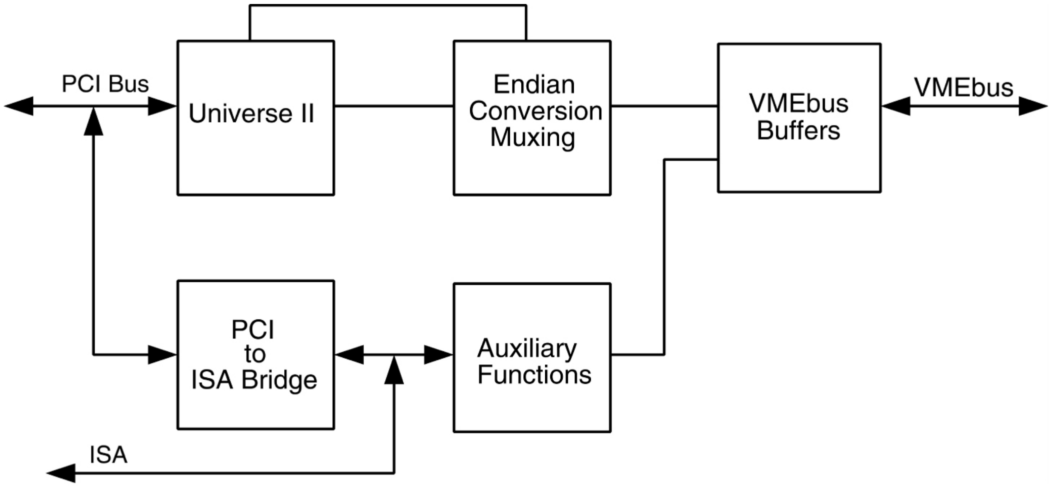
The Universe II revision is another example of Tundra’s commitment to supporting the VMEbus community. Tundra is actively participating in VMEbus, PCI bus, and Compact PCI bus standards and vendor associations, as well as related SIGs. Tundra will continue to propose and support enhancements to these specifications, while increasing both the range of options available to our customers and the compatibility between VME and PCI. Please visit our web site at www.tundra.com to keep abreast of these developments.

Chapter 2 Functional Description

Architectural Overview

The VMEbus Interface consists of a Universe II chip, endian conversion muxing hardware, and miscellaneous auxiliary functions hardware. These components are graphically illustrated in Figure 2-1. The majority of the interface functionality including the basic PCI-to-VMEbus interface, DMA controller functionality, and VMEbus system controller functionality, is provided by the Universe II chip. The auxiliary functions hardware provides the system registers, the endian conversion control logic, the non-slot 1 bus timeout timer, the VME BERR address capture and interrupt logic, as well as various other miscellaneous logic.

Figure 2-1: Universe II-Based PCI-to-VMEbus Interface



PCI-to-VMEbus Interface Jumpers

The VMEbus Interface is tested for system operation and shipped with factory-installed header jumpers. The interface is shipped with the VME board as part of the VMEbus Interface. Figure 2-2 illustrates the physical location of the user-configurable jumpers. Table 2-1 lists each jumper designator, its function, and the factory-installed default configuration.

Figure 2-2: Edge View of the Jumper Locations

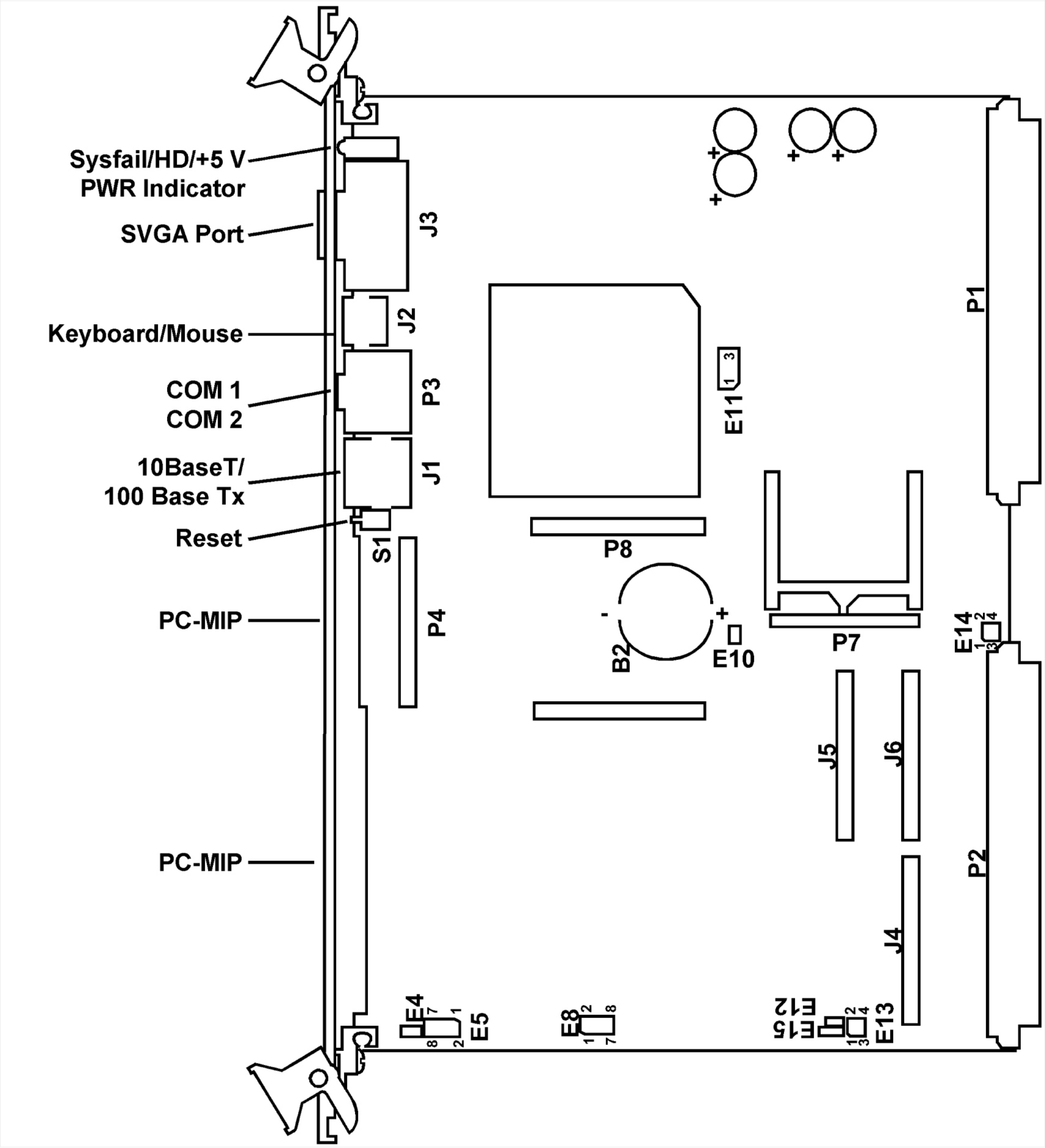


Table 2-1: VMEbus Interface Jumper Functions and Factory Settings

Jumper	Function	Factory Setting
E100	Installed - SYSFAIL not asserted upon reset Removed - SYSFAIL asserted upon reset	Installed
E101	Installed - Universe memory mapped Removed - Universe I/O mapped	Installed (Should not be removed)
E102	Installed - Drives VMEbus SYSRESET Removed - Does not drive	Installed
E103	Installed - Receives VMEbus SYSRESET Removed - Does not receive	Installed

Please refer to the VMEbus Interface Volume I manual for other jumper settings.

Caution

The Universe II chip maps the 4K register set in both I/O and memory space each with 4 Kbyte resolution. The registers may thus be accessed using either mode. However, in order to maintain compatibility with existing software, memory space access should be used. The address of the registers are contained in the Universe II's configuration space base address registers 0 and 1 (for example, config space offset 0x10 and 0x14).

Universe II Architectural Overview

This section introduces the general architecture of the Universe II. This description makes frequent reference to the functional block diagram provided in Figure 2-3. Notice that for each of the interfaces, VMEbus and PCI bus, there are three functionally distinct modules: master module, slave module, and interrupt module. These modules are connected to the different functional channels operating in the Universe II. These channels are:

- VMEbus Slave Channel
- PCI bus Target Channel
- DMA Channel
- Interrupt Channel
- Register Channel

The Architectural Overview is organized into the following sections:

- VMEbus Interface
- PCI Bus Interface
- Interrupter and Interrupt Handler
- DMA Controller

These sections describe the operation of the Universe II in terms of the different modules and channels illustrated in Figure 2-3.

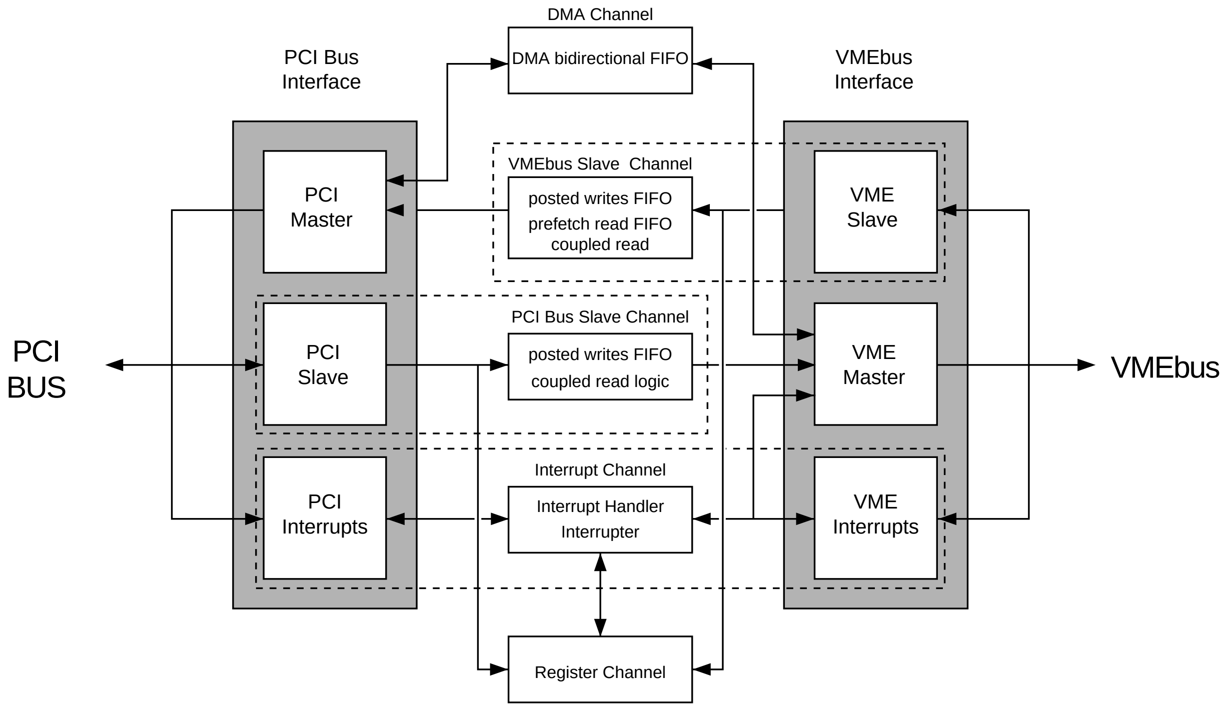
VMEbus Interface

Universe II as VMEbus Slave

The Universe II VMEbus Slave Channel accepts all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to augment 3U applications, for example, A40 and MD32). Incoming write transactions from the VMEbus may be treated as either coupled or posted, depending upon the programming of the VMEbus slave image (see “VME Slave Images” on page 2-49). With posted write transactions, data is written to a Posted Write Receive FIFO (RXFIFO), and the VMEbus master receives data acknowledgment from the Universe II. Write data is transferred to the PCI resource from the RXFIFO without the involvement of the initiating VMEbus master (see “Posted Writes” on page 2-18 for a full explanation of this operation). With a coupled cycle, the VMEbus master only receives data acknowledgment when the transaction is complete on the PCI bus. This means that the VMEbus is unavailable to other masters while the PCI bus transaction is executed.

Read transactions may be either prefetched or coupled. If enabled by the user, a prefetched read is initiated when a VMEbus master requests a block read transaction (BLT or MBLT) and this mode is enabled. When the Universe II receives the block read request, it begins to fill its Read Data FIFO (RDFIFO) using burst transactions from the PCI resource. The initiating VMEbus master then acquires its block read data from the RDFIFO rather than from the PCI resources directly.

Figure 2-3: Architectural Diagram for the Universe II



Universe II as VMEbus Master

The Universe II becomes VMEbus master when the VMEbus Master Interface is internally requested by the PCI Bus Target Channel, the DMA Channel, or the Interrupt Channel. The Interrupt Channel always has priority over the other two channels. Several mechanisms are available to configure the relative priority that the PCI Bus Target Channel and DMA Channel have over ownership of the VMEbus Master Interface.

The Universe II's VMEbus Master Interface generates all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to augment 3U applications, for example, A40 and MD32). The Universe II is also compatible with all VMEbus modules conforming to pre-VME64 specifications. As VMEbus master, the Universe II supports Read-Modify-Write (RMW), and Address-Only-with-Handshake (ADOH) but does not accept RETRY* as a termination from the VMEbus slave. The ADOH cycle is used to implement the VMEbus Lock command allowing a PCI master to lock VMEbus resources.

PCI Bus Interface

Universe II as PCI Target

Read transactions from the PCI bus are always processed as coupled. Write transactions may be either coupled or posted, depending upon the setting of the PCI bus target image (see “PCI Bus Target Images” on page 2-52). With a posted write transaction, write data is written to a Posted Write Transmit FIFO (TXFIFO) and the PCI bus master receives data acknowledgment from the Universe II with zero wait-states. Meanwhile, the Universe II obtains the VMEbus and writes the data to the VMEbus resource independent of the initiating PCI master (see “Posted Writes” on page 2-18 for a full description of this operation).

To allow PCI masters to perform RMW and ADOH cycles, the Universe II provides a Special Cycle Generator. The Special Cycle Generator can be used in combination with a VMEbus ownership function to guarantee PCI masters exclusive access to VMEbus resources over several VMEbus transactions (see “The Special Cycle Generator” on page 2-44 and “Using the VOWN bit” on page 2-47 for a full description of this functionality).

Universe II as PCI Master

The Universe II becomes PCI master when the PCI Master Interface is internally requested by the VMEbus Slave Channel or the DMA Channel. There are mechanisms provided which allow the user to configure the relative priority of the VMEbus Slave Channel and the DMA Channel.

Interrupter and Interrupt Handler

Interrupter

The Universe II Interrupt Channel provides a flexible scheme to map interrupts to the PCI bus or VMEbus Interface. Interrupts are generated from hardware or software sources (see “Interrupt Generation” on page 2-60 and “Interrupt Handling” on page 2-67 for a full description of hardware and software sources). Interrupt sources can be mapped to any of the PCI bus or VMEbus interrupt output pins. Interrupt sources mapped to VMEbus interrupts are generated on the VMEbus interrupt output pins VIRQ# [7:1]. When a software and hardware source are assigned to the same VIRQn# pin, the software source always has higher priority.

Interrupt sources mapped to PCI bus interrupts are generated on one of the INT# [7:0] pins. To be fully PCI compliant, all interrupt sources must be routed to a single INT# pin.

For VMEbus interrupt outputs, the Universe II interrupter supplies an 8-bit STATUS/ID to a VMEbus interrupt handler during the IACK cycle, and optionally generates an internal interrupt to signal that the interrupt vector has been provided (see “VMEbus Interrupt Generation” on page 2-64).

Interrupts mapped to PCI bus outputs are serviced by the PCI interrupt controller. The CPU determines which interrupt sources are active by reading an interrupt status register in the Universe II. The source negates its interrupt when it has been serviced by the CPU (see “PCI Interrupt Generation” on page 2-61).

VMEbus Interrupt Handling

A VMEbus interrupt triggers the Universe II to generate a normal VMEbus IACK cycle and generate the specified interrupt output. When the IACK cycle is complete, the Universe II releases the VMEbus and the interrupt vector is read by the PCI resource servicing the interrupt output. Software interrupts are ROAK, while hardware and internal interrupts are RORA.

DMA Controller

The Universe II provides an internal DMA controller for high performance data transfer between the PCI and VMEbus. DMA operations between the source and destination bus are decoupled through the use of a single bidirectional FIFO (DMAFIFO). Parameters for the DMA transfer are software configurable in the Universe II registers (see “DMA Controller” on page 2-75).

The principal mechanism for DMA transfers is the same for operations in either direction (PCI-to-VMEbus, or VMEbus-to-PCI), only the relative identity of the source and destination bus changes. In a DMA transfer, the Universe II gains control of the source bus and reads data into its DMAFIFO. Following specific rules of DMAFIFO operation (see “FIFO Operation and Bus Ownership” on page 2-89), it then acquires the destination bus and writes data from its DMAFIFO.

The DMA controller can be programmed to perform multiple blocks of transfers using entries in a linked-list. The DMA will work through the transfers in the linked-list following pointers at the end of each linked-list entry. Linked-list operation is initiated through a pointer in an internal Universe II register, but the linked-list itself resides in PCI bus memory.

VMEbus Interface

The VMEbus Interface incorporates all operations associated with the VMEbus. This includes master and slave functions, VMEbus configuration and system controller functions. These operations are covered as follows:

- VMEbus Requester, page 2-10
- Universe II as VMEbus Master, page 2-12
- Universe as VMEbus Slave, page 2-17
- VMEbus Configuration, page 2-25
- Automatic Slot Identification, page 2-26
- System Controller Functions, page 2-28
- BI-Mode, page 2-30.
  For information concerning the Universe II VMEbus slave images, see “VME Slave Images”.

VMEbus Requester

Internal Arbitration for VMEbus Requests

Three different internal channels within the Universe II require use of the VMEbus: the Interrupt Channel, the PCI Target Channel, and the DMA Channel. These three channels do not directly request the VMEbus, instead they compete internally for ownership of the VMEbus Master Interface.

The Interrupt Channel (refer to Figure 2-3) always has the highest priority for access to the VMEbus Master Interface. The DMA and PCI Target Channel requests are handled in a fair manner. The channel awarded VMEbus mastership maintains ownership of the VMEbus until it is ‘done’. The definition of ‘done’ for each channel is given on page 2-11 in “VMEbus Release”.

The Interrupt Channel requests the VMEbus master when it detects an enabled VMEbus interrupt line asserted and needs to run an interrupt acknowledge cycle to acquire the STATUS/ID.

The PCI Target Channel requests the VMEbus Master Interface to service the following conditions:

- the TXFIFO contains a complete transaction, or
- if there is a coupled cycle request.

The DMA Channel requests the VMEbus Master Interface if:

- the DMAFIFO has 64 bytes available (if it is reading from the VMEbus) or 64 bytes in its FIFO (if it is writing to the VMEbus), or
- the DMA block is complete (see “DMA Controller” on page 2-75).

In the case of the DMA Channel, the user can optionally use the DMA Channel VMEbus-off-timer to further qualify requests from this channel. The VMEbus-off-timer controls how long the DMA remains off the VMEbus before making another request (see “PCI-to-VMEbus Transfers” on page 2-89).

The Universe II provides a software mechanism for VMEbus acquisition through the VMEbus ownership bit (VOWN in the MAST_CTL register, Table B-160). When the VMEbus ownership bit is set, the Universe II acquires the VMEbus and sets an acknowledgment bit (VOWN_ACK in the MAST_CTL register, Table B-160) and optionally generates an interrupt to the PCI bus (see “VME Lock Cycles—Exclusive Access to VMEbus Resources” on page 2-46). The Universe II maintains VMEbus ownership until the ownership bit is cleared. During the VMEbus tenure initiated by setting the ownership bit, only the PCI Target Channel and Interrupt Channel can access the VMEbus Master Interface.

Request Modes

Request Levels

The Universe II is software configurable to request on all VMEbus request levels: BR3*, BR2*, BR1*, and BR0*. The default setting is for level 3 VMEbus request. The request level is a global programming option set through the VRL field in the MAST_CTL register (Table B-160). The programmed request level is used by the VMEbus Master Interface regardless of the channel (Interrupt Channel, DMA Channel, or PCI Target Channel) currently accessing the VMEbus Master Interface.

Fair and Demand

The Universe II requester may be programmed for either Fair or Demand mode. The request mode is a global programming option set through the VRM bits in the MAST_CTL register (Table B-160).

In Fair mode, the Universe II does not request the VMEbus until there are no other VMEbus requests pending at its programmed level. This mode ensures that every requester on an equal level has access to the bus.

In Demand mode (the default setting), the requester asserts its bus request regardless of the state of the BRn* line. By requesting the bus frequently, requesters far down the daisy chain may be prevented from ever obtaining bus ownership. This is referred to as “starving” those requesters. Note that in order to achieve fairness, all bus requesters in a VMEbus system must be set to Fair mode.

VMEbus Release

The Universe II VMEbus requester can be configured as either RWD (release when done) or ROR (release on request) using the VREL bit in the MAST_CTL register (Table B-160). The default setting is for RWD. ROR means the Universe II releases BBSY* only if a bus request is pending from another VMEbus master and once the channel that is the current owner of the VMEbus Master Interface is done. Ownership of the bus may be assumed by another channel without re-arbitration on the bus if there are no pending requests on any level on the VMEbus. When set for RWD, the VMEbus Master Interface releases BBSY* when the channel accessing the VMEbus Master Interface is done. Note that the MYBBSY status bit in the MISC_STAT register (Table B-164) is set while the Universe II asserts the BBSY* output.

In RWD mode, the VMEbus is released when the channel (for example, the DMA Channel) is done, even if another channel has a request pending (for example, the PCI Target Channel). A re-arbitration of the VMEbus is required for any pending channel requests. Each channel has a set of rules that determine when it is ‘done’ with its VMEbus transaction.

The Interrupt Channel is done when a single interrupt acknowledge cycle is complete.

The PCI Target Channel is done under the following conditions:

- when the TXFIFO is empty (the TXFE bit is set by the Universe II in the MISC_STAT register, Table B-164),
- when the maximum number of bytes per PCI Target Channel tenure has been reached (as programmed with the PWON field in the MAST_CTL register, Table B-160),
- after each posted write, if the PWON is equal to 0b1111, as programmed in the MAST_CTL register, Table B-160
- when the coupled cycle is complete and the Coupled Window Timer has expired,
- if the Coupled Request Timer expires before a coupled cycle is retried by a PCI master, or
- when VMEbus ownership is acquired with the VOWN bit in the MAST_CTL register and then the VOWN bit is cleared (in other words, if the VMEbus is acquired through the use of the VOWN bit, the Universe II does not release BBSY* until the VOWN bit is cleared—see “VME Lock Cycles—Exclusive Access to VMEbus Resources” on page 2-46).

The DMA Channel is done under the following conditions (see “FIFO Operation and Bus Ownership” on page 2-89 and “DMA Error Handling” on page 2-93):

- DMAFIFO full during VMEbus-to-PCI bus transfers,
- DMAFIFO empty during PCI bus to VMEbus transfers,
- if an error is encountered during the DMA operation,
- the DMA VMEbus Tenure Byte Counter has expired, or
- DMA block is complete.

The Universe II does not monitor BCLR* and so its ownership of the VMEbus is not affected by the assertion of BCLR*.

Universe II as VMEbus Master

The Universe II becomes VMEbus master as a result of the following chain of events:

1. a PCI master accesses a Universe II PCI target image (leading to VMEbus access) or the DMA Channel initiates a transaction,
2. either the Universe II PCI Target Channel or the DMA Channel wins access to the VMEbus Master Interface through internal arbitration, and
3. the Universe II Master Interface requests and obtains ownership of the VMEbus.

The Universe II will also become VMEbus master if the VMEbus ownership bit is set (see “VME Lock Cycles—Exclusive Access to VMEbus Resources” on page 2-46) and in its role in VMEbus interrupt handling (see “VMEbus Interrupt Handling” on page 2-68).

The following sections describe the function of the Universe II as a VMEbus master in terms of the different phases of a VMEbus transaction: addressing, data transfer, cycle termination, and bus release.

The Universe I chip is unable to perform a 64-bit (double type) floating point operation on a read from VMEbus address, e.g., T=*Vme and T=T+1.0. This problem does not occur with the Universe II chip. However, to maintain backward compatibility, use two operations.

Addressing Capabilities

Depending upon the programming of the PCI target image (see “PCI Bus Target Images” on page 2-52), the Universe II generates A16, A24, A32, and CR/CSR address phases on the VMEbus. The address mode and type (supervisor/non-privileged and program/data) are also programmed through the PCI target image. Address pipelining is provided except during MBLT cycles, where the VMEbus specification does not permit it.

The address and AM codes that are generated by the Universe II are functions of the PCI address and PCI target image programming (see “PCI Bus Target Images” on page 2-52) or through DMA programming.

The Universe II generates Address-Only-with-Handshake (ADOH) cycles in support of lock commands for A16, A24, and A32 spaces. ADOH cycles must be generated through the Special Cycle Generator (see “The Special Cycle Generator” on page 2-44).

There are two User Defined AM codes that can be programmed through the USER_AM register (Table B-166). The USER_AM register can only be used to generate and accept AM codes 0x10 through 0x1F. These AM codes are designated as USERAM codes in the VMEbus specification. After power-up, the two values in the USER_AM register default to the same VME64 user-defined AM code.

If USER_AM codes are used with the VMEbus Slave Interface, ensure that the cycles use 32-bit addressing, and that only single cycle accesses are used. BLTs and MBLTs with USER_AM codes will lead to unpredictable behavior.

Data Transfer Capabilities

The data transfer between the PCI bus and VMEbus is depicted in Figure 2-4. The Universe II can be seen as a funnel where the mouth of the funnel is the data width of the PCI transaction. The end of the funnel is the maximum VMEbus data width programmed into the PCI target image. For example, consider a 32-bit PCI transaction accessing a PCI target image with VDW set to 16 bits. A data beat with all byte lanes enabled will be broken into two 16-bit cycles on the VMEbus. If the PCI target image is also programmed with block transfers enabled, the 32-bit PCI data beat will result in a D16 block transfer on the VMEbus. Write data is unpacked to the VMEbus and read data is packed to the PCI bus data width.

If the data width of the PCI data beat is the same as the maximum data width of the PCI target image, then the Universe II maps the data beat to an equivalent VMEbus cycle. For example, consider a 32-bit PCI transaction accessing a PCI target image with VDW set to 32 bits. A data beat with all byte lanes enabled is translated to a single 32-bit cycle on the VMEbus.

As the general rule, if the PCI bus data width is less than the VMEbus data width then there is no packing or unpacking between the two buses. The only exception to this is during 32-bit PCI multi-data beat transactions to a PCI target image programmed with maximum VMEbus data width of 64 bits. In this case, packing/unpacking occurs to make maximum use of the full bandwidth on both buses.

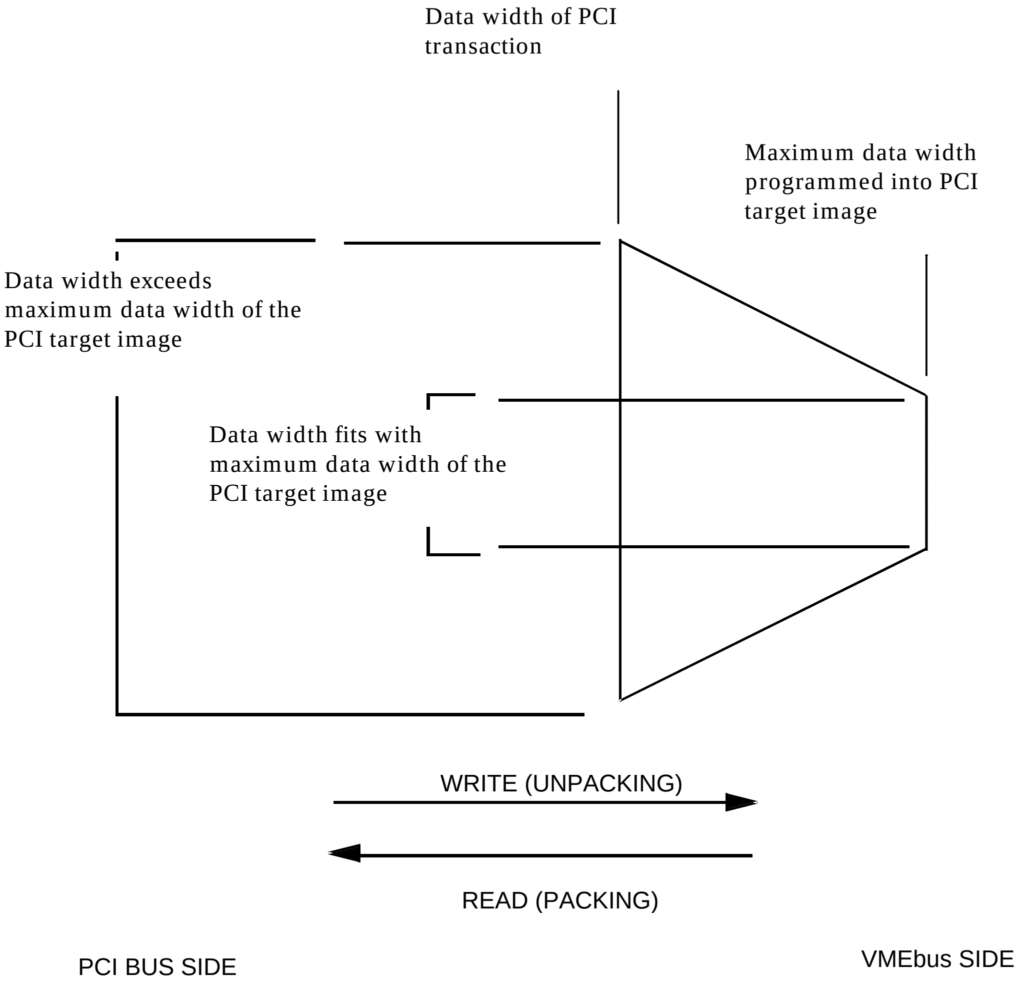
Only aligned VMEbus transactions are generated, so if the requested PCI data beat has unaligned or non-contiguous byte enables, then it is broken into multiple aligned VMEbus transactions no wider than the programmed VMEbus data width. For example, consider a three-byte PCI data beat (on a 32-bit PCI bus) accessing a PCI target image with VDW set to 16 bits. The three-byte PCI data beat will be broken into two aligned VMEbus cycles: a single-byte cycle and a double-byte cycle (the ordering of the two cycles depends on the arrangement of the byte enables in the PCI data beat). If in the above example the PCI target image has a VDW set to 8 bits, then the three-byte PCI data beat will be broken into three single-byte VMEbus cycles.

BLT/MBLT cycles are initiated on the VMEbus if the PCI target image has been programmed with this capacity (see “PCI Bus Target Images” on page 2-52). The length of the BLT/MBLT transactions on the VMEbus will be determined by the initiating PCI transaction or the setting of the PWON field in the MAST_CTL register (Table B-160). For example, a single data beat PCI transaction queued in the TXFIFO results in a single data beat block transfer on the VMEbus. With the PWON field, the user can specify a transfer byte count that will be dequeued from the TXFIFO before the VMEbus Master Interface relinquishes the VMEbus. The PWON field specifies the minimum tenure of the Universe II on the VMEbus. However, tenure is extended if the VOWN bit in the MAST_CTL register is set (see “Using the VOWN bit” on page 2-47).

During DMA operations, the Universe II will attempt block transfers to the maximum length permitted by the VMEbus specification (256 bytes for BLT, 2 Kbytes for MBLT) and as limited by the VON counter (see “DMA VMEbus Ownership” on page 2-79).

The Universe II provides indivisible transactions with the VMEbus lock commands and the VMEbus ownership bit (see “VME Lock Cycles—Exclusive Access to VMEbus Resources” on page 2-46).

Figure 2-4: Influence of Data Width and Target Image Data Width on Data Packing/Unpacking


Data width of PCI transaction
Maximum data width programmed into PCI target image
Data width exceeds maximum data width of the PCI target image
Data width fits with maximum data width of the PCI target image
WRITE (UNPACKING)
READ (PACKING)
PCI BUS SIDE
VMEbus SIDE

Cycle Terminations

The Universe II accepts BERR* or DTACK* as cycle terminations from the VMEbus slave. It does not support RETRY*. The assertion of BERR* indicates that some type of system error occurred and the transaction did not complete properly. A VMEbus BERR* received by the Universe II during a coupled transaction is communicated to the PCI master as a Target-Abort. No information is logged if the Universe II receives BERR* in a coupled transaction. If an error occurs during a posted write to the VMEbus, the Universe II uses the V_AMERR register (Table B-212) to log the AM code of the transaction (AMERR [5:0]), and the state of the IACK* signal (IACK bit, to indicate whether the error occurred during an IACK cycle). The current transaction in the FIFO is purged. The V_AMERR register also records if multiple errors have occurred (with the M_ERR bit), although the actual number of errors is not given. The error log is qualified by the value of the V_STAT bit. The address of the errored transaction is latched in the V_AERR register (Table B-214). When the Universe II receives a VMEbus error during a posted write, it generates an interrupt on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “Interrupt Handling” on page 2-67, Table B-118 and Table B-120).

DTACK* signals the successful completion of the transaction.

Universe as VMEbus Slave

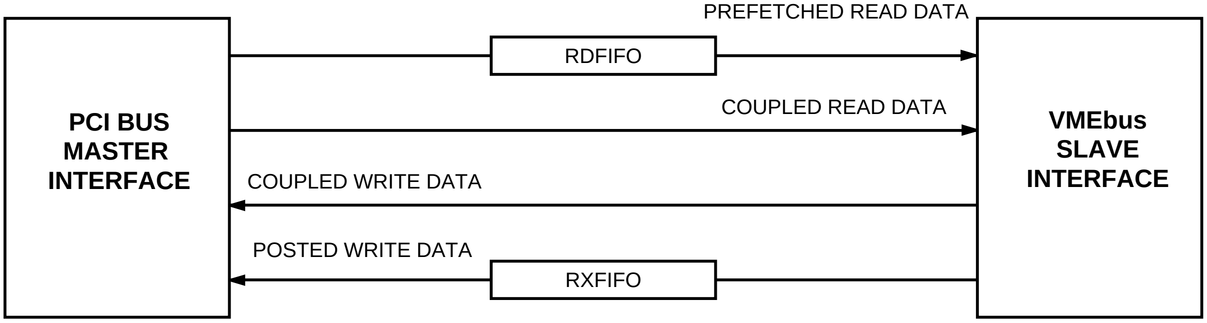
This section describes the VMEbus Slave Channel and other aspects of the Universe II as VMEbus slave. The following topics are discussed:

- Coupled Transfers, on page 2-18
- Posted Writes, on page 2-18
- Prefetched Block Reads, on page 2-20
- VMEbus Lock Commands (ADOH Cycles), on page 2-21
- VMEbus Read-Modify-Write Cycles (RMW Cycles), on page 2-22
- Location Monitors on page 2-22
- Generating PCI Configuration Cycles, on page 2-23.

The Universe II becomes VMEbus slave when one of its eight programmed slave images or register images are accessed by a VMEbus master (note that the Universe II cannot reflect a cycle on the VMEbus and access itself). Depending upon the programming of the slave image, different possible transaction types result (see “VME Slave Images” on page 2-49 for a description of the types of accesses to which the Universe II responds).

For reads, the transaction can be coupled or prefetched. Similarly, write transactions can be coupled or posted. The type of read or write transaction allowed by the slave image depends on the programming of that particular VMEbus slave image (see Figure 2-5 below and “VME Slave Images” on page 2-49). To ensure sequential consistency, prefetched reads, coupled reads, and coupled write operations are only processed once all previously posted write operations have completed (for example, the RXFIFO is empty).

Figure 2-5: VMEbus Slave Channel Dataflow



Incoming cycles from the VMEbus can have data widths of 8-bit, 16-bit, 32-bit, and 64-bit. Although the PCI bus supports only two port sizes (32-bit and 64-bit), the byte lanes on the PCI bus can be individually enabled, which allows each type of VMEbus transaction to be directly mapped to the PCI data bus.

Note

In order for a VMEbus slave image to respond to an incoming cycle, the PCI Master Interface must be enabled (bit BM in the PCI_CSR register, Table B-4). If data is enqueued in the VMEbus Slave Channel FIFO and the PCI BM bit is cleared, the FIFO will empty but no additional transfers will be received.

The VMEbus Interface supports a 32-bit PCI bus only

Coupled Transfers

A coupled transfer means that no FIFO is involved in the transaction and handshakes are relayed directly through the Universe II. Coupled mode is the default setting for the VMEbus slave images. Coupled transfers only proceed once all posted write entries in the RXFIFO have completed (see Posted Writes below).

A coupled cycle with multiple data beats (for example, block transfers) on the VMEbus side is always mapped to single data beat transactions on the PCI bus, where each data beat on the VMEbus is mapped to a single data beat transaction on the PCI bus regardless of data beat size. No packing or unpacking is performed. The only exception to this is when a D64 VMEbus transaction is mapped to D32 on the PCI bus. The data width of the PCI bus depends on the programming of the VMEbus slave image (32-bit or 64-bit, see “VME Slave Images” on page 2-49). The Universe II enables the appropriate byte lanes on the PCI bus as required by the VMEbus transaction. For example, a VMEbus slave image programmed to generate 32-bit transactions on the PCI bus is accessed by a VMEbus D08 BLT read transaction (prefetching is not enabled in this slave image). The transaction is mapped to single data beat 32-bit transfers on the PCI bus with only one byte lane enabled.

Target-Retry from a PCI target is not communicated to the VMEbus master. PCI transactions terminated with Target-Abort or Master-Abort are terminated on the VMEbus with BERR*. Note that the Universe II sets the R_TA or R_MA bits in the PCI_CS register (Table B-4) when it receives a Target-Abort or Master-Abort.

Posted Writes

A posted write involves the VMEbus master writing data into the Universe II’s RXFIFO, rather than directly to the PCI address. Write transactions from the VMEbus are processed as posted if the PWEN bit is set in the VMEbus slave image control register (see “VME Slave Images” on page 2-49). If the bit is cleared (the default setting) the transaction bypasses the FIFO and is performed as a coupled transfer (see above). Incoming posted writes from the VMEbus are queued in the 32-entry deep RXFIFO. (The RXFIFO is the same structure as the RDFIFO. The different names are used for the FIFO’s two roles, only one of which it can implement at once.) Each entry in the RXFIFO can contain 64 address bits, or 64 data bits. Each incoming VMEbus address phase, whether it is 16-bit, 24-bit, or 32-bit, constitutes a single entry in the RXFIFO and is followed by subsequent data entries. The address entry contains the translated PCI address space and command information mapping relevant to the particular VMEbus slave image that has been accessed (see “VME Slave Images” on page 2-49). For this reason, any re-programming of VMEbus slave image attributes will only be reflected in RXFIFO entries queued after the re-programming. Transactions queued before the re-programming are delivered to the PCI bus with the VMEbus slave image attributes that were in use before the re-programming.

Incoming non-block write transactions from the VMEbus require two entries in the RXFIFO: one address entry (with accompanying command information) and one data entry. The size of the data entry corresponds to the data width of the VMEbus transfer. Block transfers require at least two entries: one entry for address and command information, and one or more data entries. The VMEbus Slave Channel packs data received during block transfers to the full 64-bit width of the RXFIFO. For example, a ten data phase D16 BLT transfer (20 bytes in total) does not require ten data entries in the RXFIFO. Instead, eight of the ten data phases (16 bits per data phase for a total of 128 bits) are packed into two 64-bit data entries in the RXFIFO. The final two data phases (32 bits combined) are queued in the next RXFIFO entry. When you add the address entry to the three data entries, this VMEbus block write has been stored in a total of four RXFIFO entries.

Unlike the PCI Target Channel (see page 2-39) the VMEbus Slave Channel does not retry the VMEbus if the RXFIFO does not have enough space to hold an incoming VMEbus write transaction. Instead, the DTACK* response from the VMEbus Slave Interface is delayed until space becomes available in the RXFIFO. Since single transfers require two entries in the RXFIFO, two entries must be freed up before the VMEbus Slave Interface asserts DTACK*. Similarly, the VMEbus Slave Channel requires two available RXFIFO entries before it can acknowledge the first data phase of a BLT or MBLT transfer (one entry for the address phase and one for the first data phase). If the RXFIFO has no available space for subsequent data phases in the block transfer, then the VMEbus Slave Interface delays assertion of DTACK* until a single entry is available for the next data phase in the block transfer.

The PCI Master Interface uses transactions queued in the RXFIFO to generate transactions on the PCI bus. No address phase deletion is performed, so the length of a transaction on the PCI bus corresponds to the length of the queued VMEbus transaction. Non-block transfers are generated on the PCI bus as single data beat transactions. Block transfers are generated as one or more burst transactions, where the length of the burst transaction is programmed by the (PABS field in the MAST_CTL register, Table B-160).

The Universe II always packs or unpacks data from the VMEbus transaction to the PCI bus data width programmed into the VMEbus slave image (with all PCI bus byte lanes enabled). For example, consider a VMEbus slave image programmed for posted writes and a D32 PCI bus that is accessed with a VMEbus D16 block write transaction. The VMEbus D16 write transaction is mapped to a D32 write transaction on the PCI bus with all byte lanes enabled. (However, note that a single D16 transaction from the VMEbus is mapped to the PCI bus as D32 with only two byte lanes enabled).

During block transfers, the Universe II will pack data to the full negotiated width of the PCI bus. This may imply that for block transfers that begin or end on addresses not aligned to the PCI bus width different byte lanes may be enabled during each data beat.

If an error occurs during a posted write to the PCI bus, the Universe II uses the L_CMDERR register (Table B-62) to log the command information for the transaction (CMDERR [3:0]). The L_CMDERR register also records if multiple errors have occurred (with the M_ERR bit) although the actual number is not given. The error log is qualified with the L_STAT bit. The address of the errored transaction is latched in the LAERR register (Table B-64). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see "Bus Error Handling" on page 2-57 and "Interrupt Handling" on page 2-67).

Prefetched Block Reads

Prefetching of read data occurs for VMEbus block transfers (BLT, MBLT) in those slave images that have the prefetch enable (PREN) bit set (see "VME Slave Images" on page 2-49). In the VMEbus Slave Channel, prefetching is not supported for non BLT/MBLT transfers.

Without prefetching, block read transactions from a VMEbus master are handled by the VMEbus Slave Channel as coupled reads. This means that each data phase of the block transfer is translated to a single data beat transaction on the PCI bus. In addition, only the amount of data requested during the relevant data phase is fetched from the PCI bus. For example, a D16 block read transaction with 32 data phases on the VMEbus maps to 32 PCI bus transactions, where each PCI bus transaction has only two byte lanes enabled. Note the VMEbus lies idle during the arbitration time required for each PCI bus transaction, resulting in a considerable performance degradation.

With prefetching enabled, the VMEbus Slave Channel uses a 32-entry deep RDFIFO to provide read data to the VMEbus with minimum latency. (The RXFIFO is the same structure as the RDFIFO. The different names are used for the FIFO's two roles, only one of which it can implement at once.) The RDFIFO is 64 bits wide, with additional bits for control information. If a VMEbus slave image is programmed for prefetching, then a block read access to that image causes the VMEbus Slave Channel to generate aligned burst read transactions on the PCI bus (the size of the burst read transactions is determined by the setting of the aligned burst size, PABS in the MAST_CTL register). These PCI burst read transaction are queued in the RDFIFO and the data is then delivered to the VMEbus. Note that the first data phase provided to the VMEbus master is essentially a coupled read, but subsequent data phases in the VMEbus block read are delivered from the RDFIFO and are essentially decoupled (see "Prefetched Reads" on page 2-58 for the impact on bus error handling).

The data width of the transaction on the PCI bus (32-bit or 64-bit) depends on the setting of the LD64EN bit in the VMEbus slave image control register (e.g. see Table B-168) and the capabilities of the accessed PCI target. Internally, the prefetched read data is packed to 64 bits, regardless of the width of the PCI bus or the data width of the original VMEbus block read (no address information is stored with the data). Once one entry is queued in the RDFIFO, the VMEbus Slave Interface delivers the data to the VMEbus, unpacking the data as necessary to fit with the data width of the original VMEbus block read (e.g. D16, or D32). The VMEbus Slave Interface continuously delivers data from the RDFIFO to the VMEbus master performing the block read transaction. Because PCI bus data transfer rates exceed those of the VMEbus, it is unlikely that the RDFIFO will ever be unable to deliver data to the VMEbus master. For this reason, block read performance on the VMEbus will be similar to that observed with block writes. However, should the RDFIFO be unable to deliver data to the VMEbus master (which may happen if there is considerable traffic on the PCI bus or the PCI bus target has a slow response) the VMEbus Slave Interface delays DTACK* assertion until an entry is queued and is available for the VMEbus block read.

On the PCI side, prefetching continues as long as there is room for another transaction in the RDFIFO and the initiating VMEbus block read is still active. The space required in the RDFIFO for another PCI burst read transaction is determined by the setting of the PCI aligned burst size (PABS in the MAST_CTL register, Table B-160). If PABS is set for 32 bytes, there must be four entries available in the RDFIFO; for aligned burst size set to 64 bytes, eight entries must be available, for aligned burst size set to 128 bytes, there must be 16 entries available. When there is insufficient room in the RDFIFO to hold another PCI burst read, the read transactions on the PCI bus are terminated and only resume if room becomes available for another aligned burst AND the original VMEbus block read is still active. When the VMEbus block transfer terminates, any remaining data in the RDFIFO is purged.

Reading on the PCI side will not cross a 1024-byte boundary. The PCI Master Interface will release FRAME# and the VMEbus Slave Channel will relinquish internal ownership of the PCI Master Interface when it reaches this boundary. The VMEbus Slave Channel will re-request internal ownership of the PCI Master Interface as soon as possible, in order to continue reading from the external PCI target. (As described elsewhere, the PABS setting determines how much data must be available in the RDFIFO before the VMEbus Slave Channel continues reading.)

Regardless of the read request, the data width of prefetching on the PCI side is full width with all byte lanes enabled. If the request is unaligned, then the first PCI data beat will have only the relevant byte lanes enabled. Subsequent data beats will have full data width with all byte lanes enabled. If LD64EN is set in the VMEbus Slave image, the Universe II requests D64 on the PCI bus by asserting REQ64# during the address phase. If the PCI target does not respond with ACK64#, subsequent data beats are D32.

If an error occurs on the PCI bus, the Universe II does not translate the error condition into a BERR* on the VMEbus. Indeed, the Universe II does not directly map the error. By doing nothing, the Universe II forces the external VMEbus error timer to expire.

The VMEbus Interface supports a 32-bit PCI bus only

VMEbus Lock Commands (ADOH Cycles)

The Universe II supports VMEbus lock commands as described in the VME64 specification. Under the specification, ADOH cycles are used to execute the lock command (with a special AM code). Any resource locked on the VMEbus cannot be accessed by any other resource during the bus tenure of the VMEbus master.

When the Universe II receives a VMEbus lock command, it asserts LOCK# to the addressed resource on the PCI bus. The PCI Master Interface processes this as a read transfer (with no data). All subsequent slave VMEbus transactions are coupled while the Universe II owns PCI LOCK#. Note that the VMEbus Slave Channel has dedicated access to the PCI Master Interface during the locked transaction. The Universe II holds the PCI bus lock until the VMEbus lock command is terminated, for example, when BBSY* is negated.

The Universe II accepts ADOH cycles in any of the slave images when the Universe II PCI Master Interface is enabled (BM bit in PCI_CSR register) and the images are programmed to map transactions into PCI Memory Space.

In the event that a Target-Abort or a Master-Abort occurs during a locked transaction on the PCI bus, the Universe II will relinquish its ownership of LOCK# in accord with the PCI bus Specification. It is the responsibility of the user to verify the R_MA and R_TA status bits of the PCI_CSR status register to determine whether or not ownership of LOCK# was lost.

Once an external VMEbus masters locks the PCI bus, the Universe II DMA will not perform transfers on the PCI bus until the bus is unlocked.

VMEbus Read-Modify-Write Cycles (RMW Cycles)

A read-modify-write (RMW) cycle allows a VMEbus master to read from a VMEbus slave and then write to the same resource without relinquishing bus tenure between the two operations. Each of the Universe II slave images can be programmed to map RMW transactions to PCI locked transactions. If the LLRMW enable bit is set in the selected VMEbus slave image control register (e.g. Table B-168), then every non-block slave read is mapped to a coupled PCI locked read. LOCK# will be held on the PCI bus until AS* is negated on the VMEbus. Every non-block slave read is assumed to be a RMW since there is no possible indication from the VMEbus master that the single cycle read is just a read or the beginning of a RMW.

If the LLRMW enable bit is not set and the Universe II receives a VMEbus RMW cycle, the read and write portions of the cycle will be treated as independent transactions on the PCI bus: for example, a read followed by a write. The write may be coupled or decoupled depending on the state of the PWEN bit in the accessed slave image.

Note

There may be an adverse performance impact for reads that are processed through a RMW-capable slave image; this may be accentuated if LOCK# is currently owned by another PCI master.

RMW cycles are not supported with unaligned or D24 cycles.

When an external VMEbus Master begins a RMW cycle, at some point a read cycle will appear on the PCI bus. During the time between when the read cycle occurs on the PCI bus and when the associated write cycle occurs on the PCI bus, no DMA transfers will occur on the PCI bus.

Register Accesses

See “Universe II Registers” on page 2-97 for a full description of register mapping and register access.

Location Monitors

Universe II has four location monitors to support a VMEbus broadcast capability. The location monitors’ image is a 4-Kbyte image in A16, A24 or A32 space on the VMEbus. If enabled, an access to a location monitor causes the PCI Master Interface to generate an interrupt.

The Location Monitor Control Register (LM_CTL, Table B-200) controls the Universe II’s location monitoring. The EN field of the LM_CTL register enables the capability. The PGM[1:0] field sets the Program/Data AM code. The SUPER[1:0] field of the LM_CTL register sets the Supervisor/User AM code to which the Universe II responds. The VAS[3:0] field of the LM_CTL register specifies the address space that is monitored. The BS[31:12] field of the location monitor Base Address Register (LM_BS, Table B-202) specifies the lowest address in the 4 Kbyte range that will be decoded as a location monitor access. While the Universe II is said to have four location monitors, they all share the same LM_CTL and LM_BS registers.

In address spaces A24 and A16, the respective upper address bits are ignored.

When an access to a location monitor is detected, an interrupt is generated on the PCI bus. VMEbus address bits [4:3] determine which Location Monitor will be used, and hence which of four PCI interrupts to generate. (See “Location Monitors” on page 2-73 for details on mapping the interrupts from the location monitor.)

The location monitors do not store write data. Read data from the location monitors is undefined. Location monitors do not support BLT or MBLT transfers.

Each Universe II on the VMEbus should be programmed to monitor the same 4 Kbytes of addresses on the VMEbus. Note that the Universe II may access its own location monitor. If the Universe II accesses its own (enabled) location monitor, the same Universe II generates DTACK* on the VMEbus and thereby terminates its own cycle. This removes the necessity of the system integrator ensuring that there is another card enabled to generate DTACK*. The generation of DTACK* happens after the Universe II has decoded and responded to the cycle. If the location monitor is accessed by a different master, the Universe II does not respond with DTACK*.

Generating PCI Configuration Cycles

PCI Configuration cycles can be generated by accessing a VMEbus slave image whose Local Address Space field (LAS) is set for Configuration Space.

Both Type 0 and Type 1 cycles are generated and handled through the same mechanism. Once a VMEbus cycle is received and mapped to a configuration cycle, the Universe II compares bits [23:16] of the incoming address with the value stored in the MAST_CTL Register’s Bus Number field (BUS_NO[7:0] in Table B-160). If the bits are the same as the BUS_NO field, then a TYPE 0 access is generated. If they are not the same, a Type 1 configuration access is generated. The PCI bus-generated address then becomes an unsigned addition of the incoming VMEbus address and the VMEbus slave image translation offset.

Generating Configuration Type 0 Cycles

The Universe II asserts one of AD[31:11] on the PCI bus to select a device during a configuration Type 0 access. To perform a configuration Type 0 cycle on the PCI bus:

- Program the LAS field of VSIx_CTL for Configuration Space,
- Program the VSIx_BS, VSIx_BD registers to some suitable value,
- Program the VSIx_TO register to 0, and
- Program the BUS_NO field of the MAST_CTL register to some value.

Perform a VMEbus access where:

- VA[7:2] identifies the PCI Register Number and will be mapped directly to AD[7:2],
- VA[10:8] identifies the PCI Function Number and will be mapped directly to AD[10:8],
- VA[15:11] selects the device on the PCI bus and will be mapped to AD[31:12] according to Table 2-2 on page 65,
- VA[23:16] matches the BUS_NO in MAST_CTL register, and
- Other address bits are irrelevant—they are not mapped to the PCI bus

Table 2-2: PCI Address Line Asserted as a Function of VA[15:11]

VA[15:11][1]	PCI Address Line Asserted[2]
00000	11
00001	12
00010	13
00011	14
00100	15
00101	16
00110	17
00111	18
01000	19
01001	20
01010	21
01011	22
01100	23
01101	24
01110	25
01111	26
10000	27
10001	28
10010	29
10011	30
10100	31

[1] *The other values of VA[15:11] are not defined and must not be used.*
[2] *Only one of AD[31:11] is asserted; the other address lines in AD[31:11] are negated.*

Generating Configuration Type 1 Cycles

To generate a configuration Type 1 cycle on the VMEbus:

- Program LAS field of VSIx_CTL to Configuration Space,
- Program the VSIx_BS, VSIx_BD registers to some suitable value,
- Program the VSIx_TO register to 0 and
- Program the BUS_NO field of the MAST_CTL register to some value.

Perform a VMEbus access where:

- VMEbus Address [7:2] identifies the PCI Register Number,
- VMEbus Address [10:8] identifies the PCI Function Number,
- VMEbus Address [15:11] identifies the PCI Device Number,
- VMEbus Address [23:16] does not match the BUS_NO in MAST_CTL register, and
- VMEbus Address [31:24] are mapped directly through to the PCI bus.

VMEbus Configuration

The Universe II provides the following functions to assist in the initial configuration of the VMEbus system:

- First Slot Detector,
- Register Access at Power-up, and
- Auto Slot ID (two methods).

These are described separately below.

First Slot Detector

As specified by the VME64 specification the First Slot Detector module on the Universe II samples BG3IN* immediately after reset to determine whether the Universe II’s host board resides in slot 1. The VMEbus specification requires that BG[3:0]* lines be driven high after reset. This means that if a card is preceded by another card in the VMEbus system, it will always sample BG3IN* high after reset. BG3IN* can only be sampled low after reset by the first card in the system (there is no preceding card to drive BG3IN* high). If BG3IN* is sampled at logic low immediately after reset (due to the Universe II’s internal pull-down), then the Universe II’s host board is in slot 1 and the Universe II becomes SYSCON: otherwise, the SYSCON module is disabled. This mechanism may be overridden by software through clearing or setting the SYSCON bit in the MISC_CTL register (Table B-162).

The Universe II monitors IACK* (rather than IACKIN*) when it is configured as SYSCON. This permits it to operate as SYSCON in a VMEbus chassis slot other than slot 1, provided there are only empty slots to its left. The slot with SYSCON in it becomes a virtual slot 1.

VMEbus Register Access at Power-up

The Universe II provides a VMEbus slave image that allows access to all Universe II Control and Status Registers (UCSR). The base address for this slave image is programmed through the VRAI_BS register (Table B-207). At power-up, the Universe II can program the VRAI_BS and VRAI_CTL (Table B-204) registers with information specifying the UCSR slave image (see “Power-Up Options” on page 2-112).

Register access at power-up would be used in systems where the Universe II’s card has no CPU, or where register access for that card needs to be independent of the local CPU.

Automatic Slot Identification

The Universe II supports two types of Auto-ID functionality. One type uses the Auto Slot ID technique as described in the VME64 specification. The other type uses a proprietary method developed by DY4 Systems and implemented in the Tundra SCV64. Neither system identifies geographical addressing, only the relative position amongst the boards present in the system (for example, fourth board versus fourth slot).

Auto-ID prevents the need for jumpers to uniquely identify cards in a system. This can:

- increase the speed of system level repairs in the field,
- reduce the possibility of incorrect configurations, and
- reduce the number of unique spare cards that must be stocked.

Both methods of Auto ID employed by the Universe II are described below.

Auto Slot ID: VME64 Specified

The VME64 auto ID cycle (described in the VME64 Specification) requires at power-up that the Auto ID slave

- generate IRQ2*, and
- negate SYSFAIL*.

When the Auto ID slave responds to the Monarch’s IACK cycle, it will

- enable accesses to its CR/CSR space,
- provide a Status/ID to the Monarch indicating the interrupt is an Auto-ID request,
- assert DTACK*, and
- release IRQ2*.

The Universe II participates in the VME64 auto ID cycle in either an automatic or semi-automatic mode. In its fully automatic mode, it holds SYSFAIL* asserted until SYSRST* is negated. When SYSRST* is negated, the Universe II asserts IRQ2* and releases SYSFAIL*. In its semi-automatic mode, the Universe II still holds SYSFAIL* asserted until SYSRST* is negated. However, when SYSRST* is negated, the local CPU performs diagnostics and local logic sets the AUTOID bit in the MISC_CTL register (Table B-162). This asserts IRQ2* and releases SYSFAIL*.

After SYSFAIL* is released and the Universe II detects a level 2 IACK cycle, it responds with the STATUS/ID stored in its level 2 STATID register (which defaults to 0xFE).

The Universe II can be programmed so that it will not release SYSFAIL* until the SYSFAIL bit in the VCSR_CLR register (Table B-248) is cleared by local logic (SYSFAIL* is asserted if the SYSFAIL bit in the VCSR_SET register, Table B-250, is set at power-up). Since the system Monarch does not service the Auto-ID slave until after SYSFAIL* is negated, not clearing the SYSFAIL bit allows the Auto-ID process to be delayed until the CPU completes local diagnostics. Once local diagnostics are complete, the CPU clears the SYSFAIL bit and the Auto-ID cycle proceeds.

The Monarch can perform CR/CSR reads and writes at A[23:19]= 0x00 in CR/CSR space and relocate the Universe II's CR/CSR base address.

Universe II and the Auto-ID Monarch

At power-up an Auto-ID Monarch waits to run a level 2 IACK cycle until after SYSFAIL* goes high. After the IACK cycle is performed and it has received a Status/ID indicating an Auto-ID request, the monarch software

- masks IRQ2* (so that it will not service other interrupters at that interrupt level until current Auto-ID cycle is completed),
- performs an access at 0x00 in CR/CSR space to get information about Auto-ID slave,
- moves the CR/CSR base address to a new location, and
- unmasks IRQ2* (to allow it to service the next Auto-ID slave).

The Universe II supports monarch activity through its capability to be a level 2 interrupt handler. All other activity must be handled through software residing on the board.

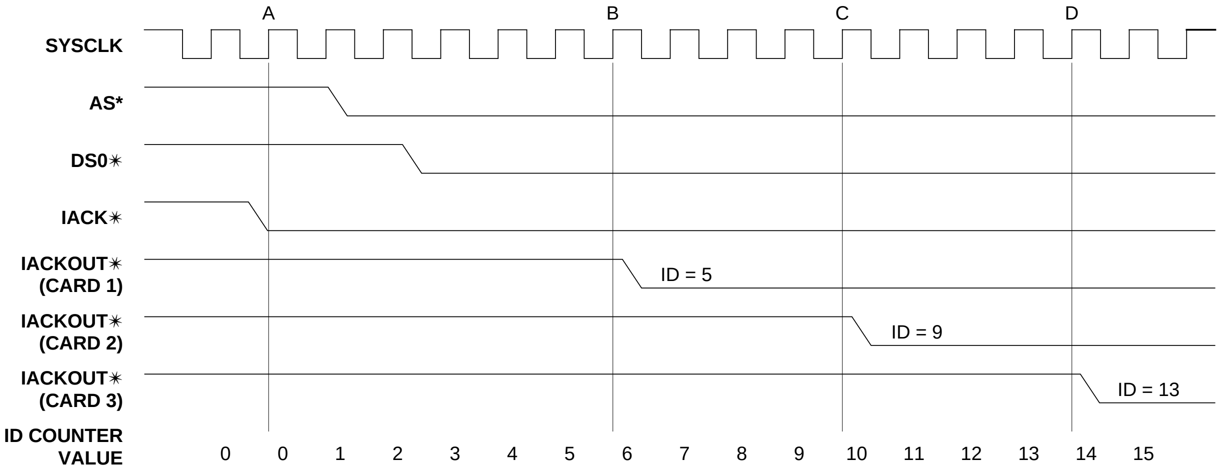
Auto-ID: A Proprietary Tundra Method

The Universe II uses a proprietary Auto-ID scheme if this is selected as a power-up option (see Auto-ID on page 163). The Tundra proprietary Auto-ID function identifies the relative position of each board in the system, without using jumpers or on-board information. The ID number generated by Auto-ID can then be used to determine the board's base address.

After any system reset (assertion of SYSRST*), the Auto-ID logic responds to the first level one IACK cycle on the VMEbus.

After the level one IACK* signal has been asserted (either through IRQ1* or with a synthesized version), the Universe II in slot 1 counts five clocks from the start of the cycle and then asserts IACKOUT* to the second board in the system (see Figure 2-6). All other boards continue counting until they receive IACKIN*, then count four more clocks and assert IACKOUT* to the next board. Finally, the last board asserts IACKOUT* and the bus pauses until the data transfer time-out circuit ends the bus cycle by asserting BERR*.

Figure 2-6: Timing for Auto-ID Cycle



Because all boards are four clocks “wide”, the value in the clock counter is divided by four to identify the slot in which the board is installed; any remainder is discarded. Note that since the start of the IACK cycle is not synchronized to SYSCLK, a one count variation from the theoretical value of the board can occur. However, in all cases the ID value of a board is greater than that of a board in a lower slot number. The result is placed in the DY4AUTOID [7:0] field and the DY4DONE bit is set (both are located in the MISC_STAT register, Table B-164).

System Controller Functions

When located in Slot 1 of the VMEbus system (see “First Slot Detector”), the Universe II assumes the role of SYSCON and sets the SYSCON status bit in the MISC_CTL register (Table B-162). In accordance with the VME64 specification, as SYSCON the Universe II provides:

- a system clock driver,
- an arbitration module,
- an IACK Daisy Chain Driver (DCD), and
- a bus timer.

System Clock Driver

The Universe II provides a 16 MHz SYSCLK signal derived from CLK64 when configured as SYSCON.

VMEbus Arbiter

When the Universe II is SYSCON, the Arbitration Module is enabled. The Arbitration Module supports the following arbitration modes:

- Fixed Priority Arbitration Mode (PRI),
- Single Level Arbitration (SGL) (a subset of PRI), or
- Round Robin Arbitration Mode (RRS) (default setting).

These are set with the VARB bit in the MISC_CTL register (Table B-162).

Fixed Priority Arbitration Mode (PRI)

In this mode, the order of priority is VRBR#[3], VRBR#[2], VRBR#[1], and VRBR#[0] as defined by the VME64 specification. The Arbitration Module issues a Bus Grant (VBGO [3:0]#) to the highest requesting level.

If a Bus Request of higher priority than the current bus owner becomes asserted, the Arbitration Module asserts VBCLR# until the owner releases the bus (VRBBSY# is negated).

Single Level Arbitration Mode (SGL)

In this mode, a subset of priority mode, all requests and grants are made exclusively on level 3. Set the Universe II in PRI mode to use this mode.

Round Robin Arbitration Mode (RRS)

This mode arbitrates all levels in a round robin mode, repeatedly scanning from levels 3 to 0. Only one grant is issued per level and one owner is never forced from the bus in favor of another requester (VBCLR# is never asserted). Since only one grant is issued per level on each round robin cycle, several scans will be required to service a queue of requests at one level.

VMEbus Arbiter Time-out

The Universe II’s VMEbus arbiter can be programmed to time-out if the requester does not assert BBSY* within a specified period. This allows BGOUT to be negated so that the arbiter may continue with other requesters. The timer is programmed using the VARBTO field in the MISC_CTL register (Table B-162), and can be set to 16 µs, 256 µs, or disabled. The default setting for the timer is 16 µs. The arbitration time-out timer has a granularity of 8 µs; setting the timer for 16 µs means the timer may timeout in as little as 8 µs.

IACK Daisy-Chain Driver Module

The IACK Daisy-Chain Driver module is enabled when the Universe II becomes system controller. This module guarantees that IACKIN* will stay high for at least 30 ns as specified in rule 40 of the VME64 specification.

VMEbus Time-out

A programmable bus timer allows users to select a VMEbus time-out period. The time-out period is programmed through the VBTO field in the MISC_CTL register (Table B-162) and can be set to 16µs, 32µs, 64µs, 128 µs, 256 µs, 512 µs, 1024 µs, or disabled. The default setting for the timer is 64 µs. The VMEbus Timer module asserts VXBERR# if a VMEbus transaction times out (indicated by one of the VMEbus data strobes remaining asserted beyond the time-out period).

The Universe II time-out timer functions only when the interface is the system controller. The VMEbus Interface provides a non-slot 1 bus timeout timer. Refer to Chapter 3 for timer information.

BI-Mode

BI-Mode® (Bus Isolation Mode) is a mechanism for logically isolating the Universe II from the VMEbus. This mechanism is useful for the following purposes:

- implementing hot-standby systems. A system may have two identically configured boards, one in BI-Mode. If the board that is not in BI-Mode fails, it can be put in BI-Mode while the spare board is removed from BI-Mode.
- system diagnostics for routine maintenance, or
- fault isolation in the event of a card failure, even if a spare board is not provided, at least the faulty board can be isolated.

While in BI-Mode, the Universe II data channels cannot be used to communicate between VMEbus and PCI (Universe II mailboxes do provide a means of communication). The only traffic permitted is to Universe II registers either through configuration cycles, the PCI register image, the VMEbus register image, or CR/CSR space. No IACK cycles will be generated or responded to. No DMA activity will occur. Any access to other PCI images will result in a Target-Retry. Access to other VMEbus images will be ignored.

Entering BI-Mode has the following effects.

- The VMEbus Master Interface becomes inactive. PCI Target Channel coupled accesses will thereafter be retried. The PCI Target Channel Posted Writes FIFO will continue to accept transactions but will eventually fill and no further posted writes will be accepted. The DMA FIFO will eventually empty or fill and no further DMA activity will take place on the PCI bus. The Universe II VMEbus Master will not service interrupts while in BI-Mode.
- The Universe II will not respond as a VMEbus slave, except for accesses to the register image and CR/CSR image.

- The Universe II will not respond to any interrupt it had outstanding. All VMEbus outputs from the Universe II will be tri-stated, so that the Universe II will not be driving any VMEbus signals. The only exception to this is the IACK and BG daisy chains which must remain in operation as before.

There are four ways to cause the Universe II to enter BI-Mode. The Universe II is put into BI-Mode:

1. if the BI-Mode power-up option is selected (See “Power-up Option Descriptions” on page 2-114 and Table 2-24),
2. when SYSRST* or RST# is asserted any time after the Universe II has been powered-up in BI-Mode,
3. when VRIRQ# [1] is asserted, provided that the ENGBI bit in the MISC_CTL register has been set, or
4. when the BI bit in the MISC_CTL register is set.

Note that when the Universe II is put in BI-Mode, the BI bit in the MISC_CTL register (Table B-162) is set. Clearing this bit ends Bi-Mode. There are two ways to remove the Universe II from BI-Mode:

1. power-up the Universe II with the BI-Mode option off , or
2. clear the BI bit in the MISC_CTL register, which will be effective only if the source of the BI-Mode is no longer active. That is, if VRIRQ# [1] is still being asserted while the ENGBI bit in the MISC_CTL register is set, then attempting to clear the BI bit in the MISC_CTL register will not be effective.

The BI-Mode power-up option is not supported in this product.

PCI Bus Interface

The PCI Bus Interface is organized as follows:

- PCI Cycles—Overview, below
- Universe II as PCI Master, on page 2-35
- Universe II as PCI Target, on page 2-39.

The Universe II PCI Bus Interface is electrically and logically directly connected to the PCI bus. For information concerning the different types of PCI accesses available, see “PCI Bus Target Images”.

PCI Cycles—Overview

The PCI bus port of the Universe II operates as a PCI compliant port with a 64-bit multiplexed address/data bus. The Universe II PCI bus Interface is configured as little-endian using address invariant translation when mapping between the VMEbus and the PCI bus. Address invariant translation preserves the byte ordering of a data structure in a little-endian memory map and a big-endian memory map (see Appendix F).

The Universe II has all the PCI signals described in the PCI specification with the exception of SBO# and SDONE (since the Universe II does not provide cache support).

Universe II PCI cycles are synchronous, meaning that bus and control input signals are externally synchronized to the PCI clock (CLK). PCI cycles are divided into four phases:

1. request,
2. address phase,
3. data transfer, and
4. cycle termination.

32-Bit Versus 64-Bit PCI

The Universe II is configured with a 32-bit or 64-bit PCI data bus at power-up (see “PCI Bus Width” on page 2-116 for directions on how to configure the PCI bus width.)

The VMEbus Interface has been configured for 32-bit PCI operation and does not support 64-bit PCI operation. No attempt should be made to perform 64-bit PCI accesses.

Each of the Universe II’s VMEbus slave images can be programmed so that VMEbus transactions are mapped to a 64-bit data bus on the PCI Interface (with the LD64EN bit, e.g. see Table B-168).

If the VMEbus slave image is programmed with a 64-bit PCI bus data width and if the Universe II powered up in a 64-bit PCI environment, then the Universe II asserts REQ64# during the address phase of the PCI transaction. If the PCI target is 64-bit capable, then it will respond with ACK64# and the Universe II will pack data to the full width (64 bits) of the PCI bus. If the PCI target is not

64-bit capable, then it does not assert ACK64# and the Universe II will pack data to a 32-bit PCI bus.

Note

REQ64# will be asserted if LD64EN is set in a 64-bit PCI system independent of whether the Universe II has a full 64 bits to transfer. This may result in a performance degradation because of the extra clocks required to assert REQ64# and to sample ACK64#. Also, there can be some performance degradation when accessing 32-bit targets with LD64EN set. Do not set this bit unless there are 64-bit targets in the slave image window.

If the VMEbus slave images are not programmed for a 64-bit wide PCI data bus, then the Universe operates transparently in a 32-bit PCI environment.

Independent of the setting of the LD64EN bit, the Universe II will never attempt a 64-bit cycle on the PCI bus if it is powered up as 32-bit.

PCI Bus Request and Parking

The Universe II supports bus parking. If the Universe II requires the PCI bus it will assert REQ# only if its GNT# is not currently asserted. When the PCI Master Module is ready to begin a transaction and its GNT# is asserted, the transfer begins immediately. This eliminates a possible one clock cycle delay before beginning a transaction on the PCI bus which would exist if the Universe II did not implement bus parking. Bus parking is described in Section 3.4.3 of the PCI Specification (Rev. 2.1).

Address Phase

PCI transactions are initiated by asserting FRAME# and driving address and command information onto the bus. In the VMEbus Slave Channel, the Universe II calculates the address for the PCI transaction by adding a translation offset to the VMEbus address (see “Universe as VMEbus Slave” on page 2-17).

The command signals (on the C/BE# lines) contain information about Memory space, cycle type and whether the transaction is read or write. Table 2-3 below gives PCI the command type encoding implemented with the Universe II.

Table 2-3: Command Type Encoding for Transfer Type

C/BE# [3:0] for PCI,C/BE# [7:4] for non-multiplexed	Command Type	Universe II Capability
0000	Interrupt Acknowledge	N/A
0001	Special Cycle	N/A
0010	I/O Read	Target/Master
0011	I/O Write	Target/Master
0100	Reserved	N/A

Table 2-3: Command Type Encoding for Transfer Type (continued)

0101	Reserved	N/A
0110	Memory Read	Target/Master
0111	Memory Write	Target/Master
1000	Reserved	N/A
1001	Reserved	N/A
1010	Configuration Read	Target/Master
1011	Configuration Write	Target/Master
1100	Memory Read Multiple	(See Text)
1101	Dual Address Cycle	N/A
1110	Memory Read Line	(See Text)
1111	Memory Write and Invalidate	(See Text)

Memory Read Multiple and Memory Read Line transactions are aliased to Memory Read transactions when the Universe II is accessed as a PCI target with these commands. Likewise, Memory Write and Invalidate is aliased to Memory Write. As a PCI initiator, the Universe II may generate Memory Read Multiple but never Memory Read Line.

PCI targets are expected to assert DEVSEL# if they have decoded the access. During a Configuration cycle, the target is selected by its particular IDSEL. If a target does not respond with DEVSEL# within 6 clocks, a Master-Abort is generated. The role of configuration cycles is described in the PCI 2.1 Specification.

Data Transfer

Acknowledgment of a data phase occurs on the first rising clock edge after both IRDY# and TRDY# are asserted by the master and target, respectively. REQ64# may be driven during the address phase to indicate that the master wishes to initiate a 64-bit transaction. The PCI target asserts ACK64# if it is able to respond to the 64-bit transaction.

Wait cycles are introduced by either the master or the target by deasserting IRDY# or TRDY#. For write cycles, data is valid on the first rising edge after IRDY# is asserted. Data is acknowledged by the target on the first rising edge with TRDY# asserted. For read cycles, data is transferred and acknowledged on first rising edge with both IRDY# and TRDY# asserted.

A single data transfer cycle is repeated every time IRDY# and TRDY# are both asserted. The transaction only enters the termination phase when FRAME# is deasserted (master-initiated termination) or if STOP# is asserted (target-initiated). When both FRAME# and IRDY# are deasserted (final data phase is complete), the bus is defined as idle.

Termination Phase

The PCI Bus Interface permits all four types of PCI terminations:

Master-Abort: the PCI bus master negates FRAME# when no target responds (DEVSEL# not asserted) after 6 clock cycles.

Target-Disconnect: a termination is requested by the target (STOP# is asserted) because it is unable to respond within the latency requirements of the PCI specification or it requires a new address phase. Target-disconnect means that the transaction is terminated after data is transferred. The Universe II will deassert REQ# for at least two clock cycles if it receives STOP# from the PCI target.

Target-Retry: termination is requested (STOP# is asserted) by the target because it cannot currently process the transaction. Retry means that the transaction is terminated after the address phase without any data transfer.

Target-Abort: is a modified version of target-disconnect where the target requests a termination (asserts STOP#) of a transaction which it will never be able to respond to, or during which a fatal error occurred. Although there may be a fatal error for the initiating application, the transaction completes gracefully, ensuring normal PCI operation for other PCI resources.

Parity Checking

The Universe II both monitors and generates parity information using the PAR signal. The Universe II monitors PAR when it accepts data as a master during a read or a target during a write. The Universe II drives PAR when it provides data as a target during a read or a master during a write. The Universe II also drives PAR during the address phase of a transaction when it is a master and monitors PAR during an address phase when it is the PCI target. In both address and data phases, the PAR signal provides even parity for C/BE#[7:0] and AD[63:0]. The Universe II continues with a transaction independent of any parity error reported during the transaction.

The Universe II can also be programmed to report address parity errors. It does this by asserting the SERR# signal and setting a status bit in its registers. No interrupt is generated, and regardless of whether assertion of SERR# is enabled, the Universe II does not respond to the errored access. If powered up in a 64-bit PCI environment, the Universe II uses PAR64 in the same way as PAR, except for AD[63:32] and C/BE[7:4].

Universe II as PCI Master

The Universe II requests PCI bus mastership through its PCI Master Interface. The PCI Master Interface is available to either the VMEbus Slave Channel (access from a remote VMEbus master) or the DMA Channel.

The VMEbus Slave Channel makes an internal request for the PCI Master Interface when:

- the RXFIFO contains a complete transaction,
- sufficient data exists in the RXFIFO to generate a transaction of length defined by the programmable aligned burst size (PABS), or
- there is a coupled cycle request.

The DMA Channel makes an internal request for the PCI Master Interface when:

- the DMAFIFO has room for 128 bytes to be read from PCI,
- the DMAFIFO has queued 128 bytes to be written to PCI, or
- the DMA block is completely queued during a write to the PCI bus.

Arbitration between the two channels for the PCI Master Interface follows a round robin protocol. Each channel is given access to the PCI bus for a single transaction. Once that transaction completes, ownership of the PCI Master Interface is granted to the other channel if it requires the bus. The VMEbus Slave Channel and the DMA Channel each have a set of rules that determine when it is 'done' with the PCI Master Interface. The VMEbus Slave Channel is done under the following conditions:

- an entire transaction (no greater in length than the programmed aligned burst size) is emptied from the RXFIFO, or
- the coupled cycle is complete.

The DMA Channel is done when:

- the boundary programmed into the PCI aligned burst size is emptied from the DMAFIFO during writes to the PCI bus, or
- the boundary programmed into the PCI aligned burst size is queued to the DMAFIFO during reads from the PCI bus.

As discussed elsewhere, access from the VMEbus may be either coupled or decoupled. For a full description of the operation of these data paths, see “Universe as VMEbus Slave” on page 2-17.

The PCI Master Interface can generate the following command types:

- I/O Read,
- I/O Write,
- Memory Read,
- Memory Read Multiple,
- Memory Write,
- Configuration Read (Type 0 and 1), and
- Configuration Write (Type 0 and 1).

The type of cycle the Universe II generates on the PCI bus depends on which VMEbus slave image is accessed and how it is programmed. For example, one slave image might be programmed as an I/O space, another as Memory space and another for Configuration space (see “VME Slave Images” on page 2-49). When generating a memory transaction, the implied addressing is either 32-bit or 64-bit aligned, depending upon the PCI target. When generating an I/O transaction, the implied addressing is 32-bit aligned and all incoming transactions are coupled.

PCI Burst Transfers

The Universe II generates aligned burst transfers of some maximum alignment, according to the programmed PCI aligned burst size (PABS field in the MAST_CTL register). The PCI aligned burst size can be programmed at 32, 64 or 128 bytes. Burst transfers will not cross the programmed boundaries. For example, when programmed for 32-byte boundaries, a new burst will begin at XXXX_XX20, XXXX_XX40, etc. If necessary, a new burst will begin at an address with the programmed alignment. To optimize PCI bus usage, the Universe II always attempts to transfer data in aligned bursts at the full negotiated width of the PCI bus.

The Universe II can perform a 64-bit data transfer over the AD [63:0] lines, if operated in a 64-bit PCI environment or against a 64-bit capable target or initiator. The LD64EN bit must be set if the access is being made through a VMEbus slave image; the LD64EN bit must be set if the access is being performed with the DMA.

The Universe II generates burst cycles on the PCI bus if it is:

- emptying the RXFIFO (the RXFE status bit in the MISC_STAT register is set when the RXFIFO empties),
- filling the RDFIFO (receives a block read request from a VMEbus master to an appropriately programmed VMEbus slave image), or
- performing DMA transfers

All other accesses are treated as single data beat transactions on the PCI bus.

During PCI burst transactions, the Universe II dynamically enables byte lanes on the PCI bus by changing the BE# signals during each data phase.

Termination

The Universe II performs a Master-Abort if the target does not respond within 6 clock cycles. Coupled PCI transactions terminated with Target-Abort or Master-Abort are terminated on the VMEbus with BERR*. The R_TA or R_MA bits in the PCI_CS register (Table B-4) are set when the Universe II receives a Target-Abort or generates a Master-Abort independent of whether the transaction was coupled, decoupled, prefetched, or initiated by the DMA.

If the Universe II receives a retry from the PCI target, then it relinquishes the PCI bus and re-requests within 2-3 PCI clock cycles. No other transactions are processed by the PCI Master Interface until the retry condition is cleared. The Universe II can be programmed to perform a maximum number of retries using the MAXRTRY field in the MAST_CTL register (Table B-160). When this number of retries has been reached, the Universe II responds in the same way as it does to a Target-Abort on the PCI bus. That is, the Universe II may issue a BERR* signal on the VMEbus. Target-Aborts are discussed in the next two paragraphs. All VMEbus slave coupled transactions and decoupled transactions will encounter a delayed DTACK once the FIFO fills until the condition clears either due to success or a retry time-out.

If the error occurs during a posted write to the PCI bus (see also “Bus Error Handling” on page 2-57), the Universe II uses the L_CMDERR register (Table B-62) to log the command information for the transaction (CMDERR [3:0]) and the address of the errored transaction is latched in the LAERR register (Table B-64). The L_CMDERR register also records if multiple errors occur (with the M_ERR bit) although the number of errors is not given. The error log is qualified with the L_STAT bit. The rest of the transaction will be purged from the RXFIFO if some portion of the write encounters an error. An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “Interrupt Handling” on page 2-67).

If an error occurs on the PCI bus, the Universe II does not translate the error condition into a BERR* on the VMEbus. Indeed, the Universe II does not directly map the error. By doing nothing, the Universe II forces the external VMEbus error timer to expire.

Parity

The Universe II monitors PAR when it accepts data as a master during a read and drives PAR when it provides data as a master during a write. The Universe II also drives PAR during the address phase of a transaction when it is a master. In both address and data phases, the PAR signal provides even parity for C/BE#[3:0] and AD[31:0]. If the Universe II is powered up in a 64-bit PCI environment, then PAR64 provides even parity for C/BE#[7:4] and AD[63:32].

The PERESP bit in the PCI_CS register (Table B-4) determines whether or not the Universe II responds to parity errors as PCI master. Data parity errors are reported through the assertion of PERR# if the PERESP bit is set. Regardless of the setting of these two bits, the D_PE (Detected Parity Error) bit in the PCI_CS register is set if the Universe II encounters a parity error as a master. The DP_D (Data Parity Detected) bit in the same register is only set if parity checking is enabled through the PERESP bit and the Universe II detects a parity error while it is PCI master (for example, it asserts PERR# during a read transaction or receives PERR# during a write).

No interrupts are generated by the Universe II in response to parity errors reported during a transaction. Parity errors are reported by the Universe II through assertion of PERR# and by setting the appropriate bits in the PCI_CS register. If PERR# is asserted to the Universe II while it is PCI master, the only action it takes is to set the DP_D. The Universe II continues with a transaction independent of any parity errors reported during the transaction.

As a master, the Universe II does not monitor SERR#. It is expected that a central resource on the PCI bus will monitor SERR# and take appropriate action.

Universe II as PCI Target

This section covers the following aspects of the Universe as PCI bus target:

- Data Transfer, on page 2-40
- Coupled Transfers, on page 2-41
- Posted Writes, on page 2-43
- The Special Cycle Generator, on page 2-44
- Using the VOWN bit, on page 2-47
- Terminations, on page 2-47.

Overview

The Universe II becomes PCI bus target when one of its eight programmed PCI target images or one of its registers is accessed by a PCI bus master (the Universe II cannot be that PCI bus master). Register accesses are discussed elsewhere (see “Universe II Registers” on page 2-97); this section describes only those accesses destined for the VMEbus.

When one of its PCI target images is accessed, the Universe II responds with DEVSEL# within two clocks of FRAME# (making the Universe II a medium speed device, as reflected by the DEVSEL field in the PCI_CS register).

As PCI target, the Universe II responds to the following command types:

- I/O Read,
- I/O Write,
- Memory Read,
- Memory Write,
- Configuration Read (Type 0),
- Configuration Write (Type 0),
- Memory Read Multiple (aliased to Memory Read),
- Memory Line Read (aliased to Memory Read),
- Memory Write and Invalidate (aliased to Memory Write).

Type 0 Configuration accesses can only be made to the Universe II’s PCI configuration registers. The PCI target images do not accept Type 0 accesses.

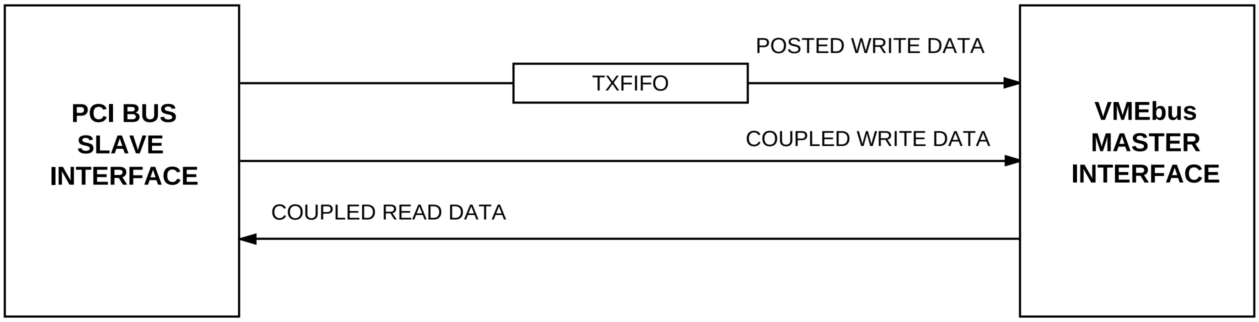
Address parity errors are reported if both PERESP and SERR_EN are set in the PCI_CS register (Table B-4). Address parity errors are reported by the Universe II by asserting the SERR# signal and setting the S_SERR (Signalled SERR#) bit in the PCI_CS register. Assertion of SERR# can be disabled by clearing the SERR_EN bit in the PCI_CS register. No interrupt is generated, and regardless of whether assertion of SERR# is enabled or not, the Universe II does not respond to the access with DEVSEL#. Typically the master of the transaction times out with a Master-Abort.

If the Universe II is accessed validly with REQ64# in Memory space as a 64-bit target, then it responds with ACK64# if it is powered up as a 64-bit device.

Data Transfer

Read transactions are always coupled (as opposed to VMEbus slave reads, which may be pre-fetched; see “Universe as VMEbus Slave” on page 2-17). Write transactions can be coupled or posted (see Figure 2-7 and “PCI Bus Target Images” on page 2-52). To ensure sequential consistency, coupled operations (reads or writes) are only processed once all previously posted write operations have completed (for example, the TXFIFO is empty).

Figure 2-7: PCI Bus Target Channel Dataflow



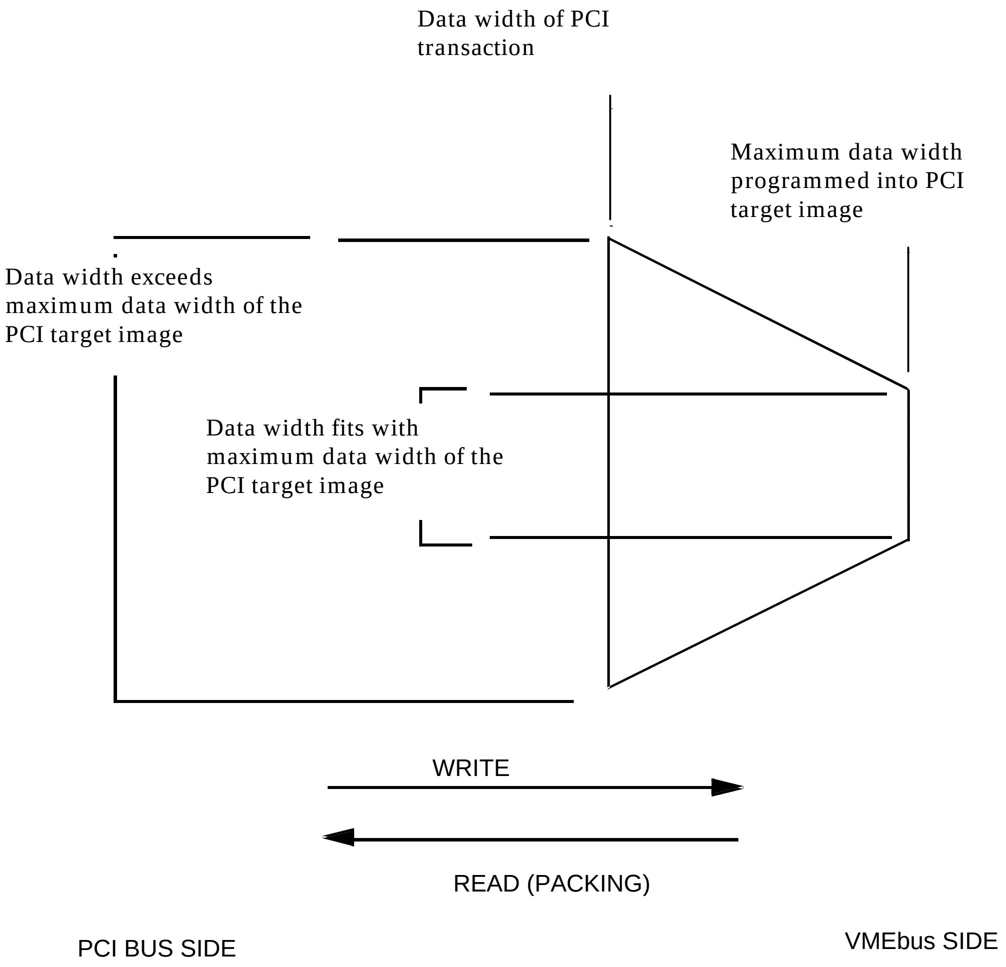
The data transfer between the PCI bus and VMEbus is perhaps best explained by Figure 2-8. The Universe II can be seen as a funnel where the mouth of the funnel is the data width of the PCI transaction. The end of the funnel is the maximum VMEbus data width programmed into the PCI target image (VDW bit in the PCI target image control register). For example, consider a 32-bit PCI transaction accessing a PCI target image with VDW set to 16 bits. A data beat with all byte lanes enabled will be broken into two 16-bit cycles on the VMEbus. If the PCI target image is also programmed with block transfers enabled, the 32-bit PCI data beat will result in a D16 block transfer on the VMEbus. Write data is unpacked to the VMEbus and read data is packed to the PCI bus data width.

If the data width of the PCI data beat is the same as the maximum data width of the PCI target image, then the Universe II maps the data beat to an equivalent VMEbus cycle. For example, consider a 32-bit PCI transaction accessing a PCI target image with VDW set to 32 bits. A data beat with all byte lanes enabled is translated to a single 32-bit cycle on the VMEbus.

As the general rule, if the PCI bus data width is less than the VMEbus data width then there is no packing or unpacking between the two buses. The only exception to this is during 32-bit PCI multi-data beat transactions to a PCI target image programmed with maximum VMEbus data width of 64 bits. In this case, packing/unpacking occurs to make maximum use of the full bandwidth on both buses.

Only aligned VMEbus transactions are generated, so if the requested PCI data beat has unaligned or non-contiguous byte enables, then it is broken into multiple aligned VMEbus transactions no wider than the programmed VMEbus data width. For example, consider a three-byte PCI data beat (on a 32-bit PCI bus) accessing a PCI target image with VDW set to 16 bits. The three-byte PCI data beat will be broken into two aligned VMEbus cycles: a single-byte cycle and a double-byte cycle (the ordering of the two cycles depends on the arrangement of the byte enables in the PCI data beat). If in the above example the PCI target image has a VDW set to 8 bits, then the three-byte PCI data beat will be broken into three single-byte VMEbus cycles.

Figure 2-8: Influence of Transaction Data Width and Target Image Data Width on Data Packing/Unpacking


Data width of PCI transaction
Maximum data width programmed into PCI target image
Data width exceeds maximum data width of the PCI target image
Data width fits with maximum data width of the PCI target image
WRITE
READ (PACKING)
PCI BUS SIDE
VMEbus SIDE

Coupled Transfers

The PCI Target Channel supports “coupled transfers”. In a nutshell, a coupled transfer through the PCI Target Channel is a transfer between PCI and VME where the Universe II maintains ownership of the VMEbus from the beginning to the end of the transfer on the PCI bus (and possibly longer), and where the termination of the cycle on the VMEbus is relayed directly to the PCI initiator in the normal manner (for example, Target-Abort, or Target Completion), rather than through error-logging and interrupts.

By default, all PCI target images are set for coupled transfers. Coupled transfers typically cause the Universe II to go through three phases: The Coupled Request Phase, the Coupled Data-Transfer Phase, and then the Coupled Wait Phase. When an external PCI Master attempts a data transfer through a slave image programmed for coupled cycles, then:

- If the Universe II currently owns the VMEbus, the PCI Target Channel moves directly to the Coupled Data-Transfer Phase; otherwise,
- the Universe II moves to the Coupled Request Phase. These three phases are described below.

Note that once the Coupled Request phase has begun, posted writes may traverse the PCI Target Channel without affecting coupled transfers.

Coupled Request Phase

During the Coupled Request Phase, the Universe II will attempt to acquire the VMEbus. But first it must empty any posted writes pending in the TXFIFO, and obtain ownership of the internal VMEbus Master Interface (see “VMEbus Release” on page 2-11 for more details on how the Universe II shares the VMEbus between channels.) The PCI Target Channel retries the PCI master until the PCI Target Channel obtains ownership of the VMEbus. Every time it issues such a retry, the Universe II restarts the Coupled Request Timer, which counts down a period of 215 PCI clock cycles. The Coupled Request Timer co-determines how long the Universe II maintains the VMEbus since the last time the Universe II issued a Target-Retry during a Coupled Request Phase: the Universe II will release (or terminate its attempt to obtain) the VMEbus if a coupled transfer is not attempted before the Coupled Request Timer expires.

Usually, an external PCI Master will attempt a coupled cycle once the Universe II has acquired the VMEbus during its Coupled Request Phase. In this case the Universe will proceed to the “Coupled Data-Transfer Phase”. No address matching is performed to verify whether the current coupled cycle matches the initiating coupled cycle. If an external PCI Master requests a PCI I/O or RMW transfer with an illegal byte lane combination, the Universe II will exit the “Coupled Request Phase.”

Coupled Data-Transfer Phase

At the beginning of the Coupled Data-Transfer Phase, the Universe II latches the PCI command, byte enable, address and (in the case of a write) data. Regardless of the state of FRAME#, the Universe II retries the master, and then performs the transaction on the VMEbus. The Universe II continues to signal Target-Retry to the external PCI master until the transfer completes (normally or abnormally) on the VMEbus.

If the transfer completes normally on the VMEbus, then in the case of a read, the data is transmitted to the PCI bus master. If a data phase of a coupled transfer requires packing or unpacking on the VMEbus, acknowledgment of the transfer is not given to the PCI bus master until all data has been packed or unpacked on the VMEbus. Successful termination is signalled on the PCI bus—the data beat is acknowledged with a Target-Disconnect, forcing all multi-beat transfers into single beat. At this point, the Universe II enters the Coupled Wait Phase.

If a bus error is signalled on the VMEbus or an error occurs during packing or unpacking, then the transaction is terminated on the PCI bus with Target-Abort.

See also “Data Transfer” on page 2-40.

Coupled Wait Phase

The Coupled Wait Phase is entered after the successful completion of a Coupled Data-Transfer phase. The Coupled Wait Phase allows consecutive coupled transactions to occur without releasing the VMEbus. If a new coupled transaction is attempted while the Universe II is in the Coupled Wait Phase, the Universe II will move directly to the Coupled Data-Transfer Phase without re-entering the Coupled Request Phase.

The Coupled Window Timer determines the maximum duration of the Coupled Wait Phase. When the Universe II enters the Coupled Wait Phase, the Coupled Window Timer starts. The period of this timer is specified in PCI clocks and is programmable through the CWT field of the LMISC register (Table B-58). If this field is programmed to 0000, the Universe II will do an early release of BBSY* during the coupled transfer on the VMEbus and will not enter the “Coupled Wait Phase.” In this case, VMEbus ownership is relinquished immediately by the PCI Target Channel after each coupled cycle.

Once the timer associated with the Coupled Wait Phase expires, the Universe II will release the VMEbus if release mode is set for RWD, or the release mode is set for ROR and there is a pending (external) request on the VMEbus.

Posted Writes

Posted writes are enabled for a PCI target image by setting the PWEN bit in the control register of the PCI target image (see “PCI Bus Target Images” on page 2-52). Write transactions are relayed from the PCI bus to the VMEbus through a 32-entry deep TXFIFO. The TXFIFO allows each entry to contain 32 address bits (with extra bits provided for command information), or up to 64 data bits. For each posted write transaction received from the PCI bus, the PCI Target Interface queues an address entry in the FIFO. This entry contains the translated address space and mapped VMEbus attributes information relevant to the particular PCI target image that has been accessed. For this reason, any re-programming of PCI bus target image attributes will only be reflected in TXFIFO entries queued after the re-programming. Transactions queued before the re-programming are delivered to the VMEbus with the PCI bus target image attributes that were in use before the re-programming.

Caution

Care should be taken before reprogramming target images from one bus while that image is being accessed from the opposite bus. If there is a chance the image may be accessed while being reprogrammed, disable the image first before changing image attributes.

Once the address phase is queued in one TXFIFO entry, the PCI Target Interface may pack the subsequent data beats to a full 64-byte width before queuing the data into new entries in the TXFIFO.

For 32-bit PCI transfers in the Universe II, the TXFIFO will accept a single burst of one address phase and 59 data phases when it is empty. For 64-bit PCI, the TXFIFO will accept a single burst of one address phase and 31 data phases when it is empty. To improve PCI bus utilization, the TXFIFO does not accept a new address phase if it does not have room for a burst of one address phase and 128 bytes of data. If the TXFIFO does not have enough space for an aligned burst, then the posted write transaction is terminated with a Target-Retry immediately after the address phase.

When an external PCI Master posts writes to the PCI Target Channel of the Universe II, the Universe II will issue a disconnect if the implied address will cross a 256-byte boundary.

Before a transaction can be delivered to the VMEbus from the TXFIFO, the PCI Target Channel must obtain ownership of the VMEbus Master Interface. Ownership of the VMEbus Master Interface is granted to the different channels on a round robin basis (see “VMEbus Release” on page 2-11). Once the PCI Target Channel obtains the VMEbus through the VMEbus Master Interface, the manner in which the TXFIFO entries are delivered depends on the programming of the VMEbus attributes in the PCI target image (see “PCI Bus Target Images” on page 2-52). For example, if the VMEbus data width is programmed to 16 bits, and block transfers are disabled, then each data entry in the TXFIFO corresponds to four transactions on the VMEbus.

If block transfers are enabled in the PCI target image, then each transaction queued in the TXFIFO, independent of its length, is delivered to the VMEbus as a block transfer. This means that if a single data beat transaction is queued in the TXFIFO, it appears on the VMEbus as a single data phase block transfer.

Any PCI master attempting coupled transactions is retried while the TXFIFO contains data. If posted writes are continually written to the PCI Target Channel, and the FIFO does not empty, coupled transactions in the PCI Target Channel will not proceed and will be continually retried. This presents a potential starvation scenario.

The Special Cycle Generator

The Special Cycle Generator in the PCI Target Channel of the Universe II can be used in conjunction with one of the PCI Target Images to generate read-modify-write (RMW) and Address Only With Handshake (ADOH) cycles.

The address programmed into the SCYC_ADDR register (Table B-50), in the address space specified by the LAS field of the SCYC_CTL register (Memory or I/O), must appear on the PCI bus during the address phase of a transfer for the Special Cycle Generator to perform its function. Whenever this address on the PCI bus matches the address in the SCYC_ADDR register, the Universe II does not respond with ACK64# (since the Special Cycle Generator only processes up to 32-bit cycles).

The cycle that is produced on the VMEbus (if any) will use attributes programmed into the Image Control Register of the image that contains the address programmed in the SCYC_ADDR register.

The Special Cycle Generator is configured through the register fields shown in Table 2-4 below.

Table 2-4: Register Fields for the Special Cycle Generator

Field	Register Bits	Description
32-bit address	ADDR in Table B-50	specifies PCI bus target image address
PCI Address Space	LAS in Table B-48	specifies whether the address specified in the ADDR field lies in PCI memory or I/O space
Special cycle	SCYC[1:0] in Table B-48	disabled, RMW or ADOH
32-bit enable	EN [31:0] in Table B-52	a bit mask to select the bits to be modified in the VMEbus read data during a RMW cycle
32-bit compare	CMP [31:0] in Table B-54	data which is compared to the VMEbus read data during a RMW cycle
32-bit swap	SWP [31:0] in Table B-56	data which is swapped with the VMEbus read data and written to the original address during a RMW cycle

The following sections describe the specific properties for each of the transfer types: RMW and ADOH.

Read-Modify-Write

When the SCYC field is set to RMW, any PCI bus read access to the specified PCI bus address (SCYC_ADDR register) will result in a RMW cycle on the VMEbus (provided the constraints listed below are satisfied). RMW cycles on the VMEbus consist of a single read followed by a single write operation. The data from the read portion of the RMW on the VMEbus is returned as the read data on the PCI bus.

RMW cycles make use of three 32-bit registers (see Table 2-4 above). The bit enable field is a bit mask which lets the user specify which bits in the read data are compared and modified in the RMW cycle. This bit enable setting is completely independent of the RMW cycle data width, which is determined by the data width of the initiating PCI transaction. During a RMW, the VMEbus read data is bitwise compared with the SCYC_CMP and SCYC_EN registers. The valid compared and enabled bits are then swapped using the SCYC_SWP register.

Each enabled bit that compares true is swapped with the corresponding bit in the 32-bit swap field. A false comparison results in the original bit being written back.

Once the RMW cycle completes, the VMEbus read data is returned to the waiting PCI bus master and the PCI cycle terminates.

Certain restrictions apply to the use of RMW cycles. If a write transaction is initiated to the VMEbus address when the special cycle field (SCYC in Table B-48) is set for RMW, then a standard write occurs with the attributes programmed in the PCI target image (in other words, the special cycle generator is not used). The Universe II performs no packing and unpacking of data on the VMEbus during a RMW operation. The following constraints must also be met.

1. The Special Cycle Generator will only generate a RMW if it is accessed with an 8-bit, aligned 16-bit, or aligned 32-bit read cycle.
2. The Special Cycle Generator will only generate a RMW if the size of the request is less than or equal to the programmed VMEbus Maximum Datawidth.
3. The destination VMEbus address space must be one of A16, A24 or A32.

In the event that the Special Cycle Generator is accessed with a read cycle that does not meet the three criteria described above, the Universe II generates a Target-Abort. Thus it is the user’s responsibility to ensure that the Universe II is correctly programmed and accessed with correct byte-lane information.

VME Lock Cycles—Exclusive Access to VMEbus Resources

The VME Lock cycle is used in combination with the VOWN bit in the MAST_CTL register to lock resources on the VMEbus. The VME Lock cycle can be used by the Universe II to inform the resource that a locked cycle is intended (so that the VMEbus slave can prevent accesses from other masters on a different bus). The VOWN bit in the MAST_CTL register can be set to ensure that when the Universe II acquires the VMEbus, it is the only master given access to the bus (until the VOWN bit is cleared). It may also be necessary for the PCI master to have locked the Universe II using the PCI LOCK# signal.

When the SCYC field is set to VME Lock, any write access to the specified VMEbus address will result in a VME Lock cycle on the VMEbus. A VME Lock cycle is coupled: the cycle does not complete on the PCI bus until it completes on the VMEbus. Reads to the specified address translate to VMEbus reads in the standard fashion. The data during writes is ignored. The AM code generated on the VMEbus is determined by the PCI target image definition for the specified VMEbus address (see Table 2-12).

However, after the VME Lock cycle is complete, there is no guarantee that the Universe II will remain VMEbus master unless it has set the VOWN bit. If the Universe II loses VMEbus ownership, then the VMEbus resouce will no longer remain locked.

The following procedure is required to lock the VMEbus via an ADOH cycle:

- (If there is more than one master on the PCI bus, it may be necessary to use PCI LOCK# to ensure that the PCI master driving the ADOH cycle has sole PCI access to the Universe II registers and the VMEbus,)
- program the VOWN bit in the MAST_CTL register to a value of 1 (see Using the VOWN bit below),
- wait until the VOWN_ACK bit in the MAST_CTL register is a value of 1,
- generate an ADOH cycle with the Special Cycle Generator,
- perform transactions to be locked on the VMEbus,

- release the VMEbus by programming the VOWN bit in the MAST_CTL register to a value of 0, and
- wait until the VOWN_ACK bit in the MAST_CTL register is a value of 0.

In the event that BERR* is asserted on the VMEbus once the Universe II has locked and owns the VMEbus, it is the responsibility of the user to release ownership of the VMEbus by programming the VOWN bit in the MAST_CTL register to a value of 0.

The following restrictions apply to the use of VME Lock cycles:

1. All byte lane information is ignored for VME Lock cycles,
2. The Universe II will generate an VME Lock cycle on the VMEbus only if the PCI Target Image which subsumes the special cycle has posted writes disabled,
3. The Universe II Special Cycle Generator will not generate VME Lock cycles if the address space is not one of A16, A24 or A32. Instead it produces regular cycles.

Using the VOWN bit

The Universe II provides a VMEbus ownership bit (VOWN bit in the MAST_CTL register, Table B-160) to ensure that the Universe II has access to the locked VMEbus resource for an indeterminate period. The Universe II can be programmed to assert an interrupt on the PCI bus when it acquires the VMEbus and the VOWN bit is set (VOWN enable bit in the LINT_EN register, Table B-112). While the VMEbus is held using the VOWN bit, the Universe II sets the VOWN_ACK bit in the MAST_CTL register. The VMEbus Master Interface maintains bus tenure while the ownership bit is cleared. This function is important for the following two reasons.

If the VMEbus Master Interface is programmed for RWD (VREL bit in MAST_CTL register), it may release the VMEbus when the PCI Target Channel has completed a transaction (definition of ‘done’ for the PCI Target Channel, see “VMEbus Release” on page 2-11). Therefore, if exclusive access to the VMEbus resource is required for multiple transactions, then the VMEbus ownership bit will hold the bus until the exclusive access is no longer required.

Alternatively, if the VMEbus Master Interface is programmed for ROR, the VMEbus ownership bit will ensure VMEbus tenure even if other VMEbus requesters require the VMEbus.

Terminations

The Universe II performs the following terminations as PCI target:

1. Target-Disconnect
   - when registers are accessed with FRAME# asserted (no bursts allowed to registers),
   - after the first data beat of every coupled cycle, or
   - after the first data phase of a PCI Memory command (with FRAME# asserted) if AD[1:0] is not equal to 00, as recommended in Revision 2.1 of the PCI Specification (page 28).
2. Target-Retry

- for 64-bit PCI, when a new posted write is attempted and the TXFIFO does not have room for a burst of one address phase and sixteen 64-bit data phases,
- when a coupled transaction is attempted and the Universe II does not own the VMEbus,
- when a coupled transaction is attempted while the TXFIFO has entries to process, or
- when a master attempts to access the Universe II's registers while a VMEbus master owns the Register Channel (e.g., through a RMW access or another type of access).

3. Target-Abort

   - when the Universe II receives BERR* on the VMEbus during a coupled cycle (BERR* translated as Target-Abort on the PCI side and the S_TA bit is set in the PCI_CS register, Table B-4).

Whether to terminate a transaction or for retry purposes, the Universe II keeps STOP# asserted until FRAME# is deasserted, independent of the logic levels of IRDY# and TRDY#. If STOP# is asserted while TRDY# is deasserted, it means that the Universe II will not transfer any more data to the master.

If an error occurs during a posted write to the VMEbus, the Universe II uses the V_AMERR register (Table B-212) to log the AM code of the transaction (AMERR [5:0]), and the state of the IACK* signal (IACK bit, to indicate whether the error occurred during an IACK cycle). The FIFO entries for the offending cycle are purged. The V_AMERR register also records whether multiple errors have occurred (with the M_ERR bit) although the number is not given. The error log is qualified with the V_STAT bit (logs are valid if the V_STAT bit is set). The address of the errored transaction is latched in the VAERR register (Table B-214). When the Universe II receives a VMEbus error during a posted write, it generates an interrupt on the VMEbus and/or PCI bus depending upon whether the VERR and VERR interrupts are enabled (see Interrupt Handling on page 2-67).

Slave Image Programming

The Universe recognizes two types of accesses on its bus interfaces: accesses destined for the other bus, and accesses decoded for its own register space. Address decoding for the Universe's register space is described in Universe II Registers on page 146. This section describes the slave images used to map transactions between the PCI bus and VMEbus.

VME Slave Images

The Universe II accepts accesses from the VMEbus within specific programmed slave images. Each VMEbus slave image opens a window to the resources of the PCI bus and, through its specific attributes, allows the user to control the type of access to those resources. The tables below describe programming for the VMEbus slave images by dividing them into VMEbus, PCI bus and Control fields.

Table 2-5: VMEbus Fields for VMEbus Slave Image

Field	Register Bits	Description
Base	BS[31:12] or BS[31:16] in VSIx_BS	multiples of 4 or 64 Kbytes (base to bound: maximum of 4 GBytes)
Bound	BD[31:12] or BD[31:16] in VSIx_BD	
address space	VAS in VSIx_CTL	A16, A24, A32, User 1, User 2
Mode	SUPER in VSIx_CTL	supervisor and/or non-privileged
Type	PGM in VSIx_CTL	program and/or data

Table2-6: PCI Bus Fields for VMEbus Slave Image

Field	Register Bits	Description
translation offset	TO[31:12] or TO[31:16] in VSIx_TO	offsets VMEbus slave address to a selected PCI address
address space	LAS in VSIx_CTL	Memory, I/O, Configuration
RMW	LLRMW in VSIx_CTL	RMW enable bit

Table 2-7: Control Fields for VMEbus Slave Image

Field	Register Bits	Description
image enable	EN in VSIx_CTL	enable bit
posted write	PWEN in VSIx_CTL	posted write enable bit
prefetched read	PREN in VSIx_CTL	prefetched read enable bit
enable PCI D64	LD64EN in VSIx_CTL	enables 64-bit PCI bus transactions

Note that the Bus Master Enable (BM) bit of the PCI_CS register must be set in order for the image to accept posted writes from an external VMEbus master. If this bit is cleared while there is data in the VMEbus Slave Posted Write FIFO, the data will be written to the PCI bus but no further data will be accepted into this FIFO until the bit is set.

Tundra recommends that the attributes in a slave image not be changed while data is enqueued in the Posted Writes FIFO. To ensure data is dequeued from the FIFO, check the RXFE status bit in the MISC_STAT register (Table B-164) or perform a read from that image. If the programming for an image is changed after the transaction is queued in the FIFO, the transaction's attributes are not changed. Only subsequent transactions are affected by the change in attributes.

Caution

Do not set bit LD64EN to '1' in any of the following registers: VSIO_CTL, VSI1-CTL,VSI2_CTL, or VSI3_CTL. 64-bit PCI bus operaton is not supported by the VMIVME-7698.

VMEbus Fields

Decoding for VMEbus accesses is based on the address, and address modifiers produced by the VMEbus master. Before responding to an external VMEbus master, the address must lie in the window defined by the base and bound addresses, and the Address Modifier must match one of those specified by the address space, mode, and type fields.

The Universe II's eight VMEbus slave images (images 0 to 7) are bounded by A32 space. The first and fifth of these images (VMEbus slave image 0 and 5) have a 4 Kbyte resolution while VMEbus slave images 1 to 3 and 6 to 8 have 64-Kbyte resolution (maximum image size of 4 GBytes). Typically, image 0 or 5 would be used as an A16 image since they provide the finest granularity of the eight images.

Caution

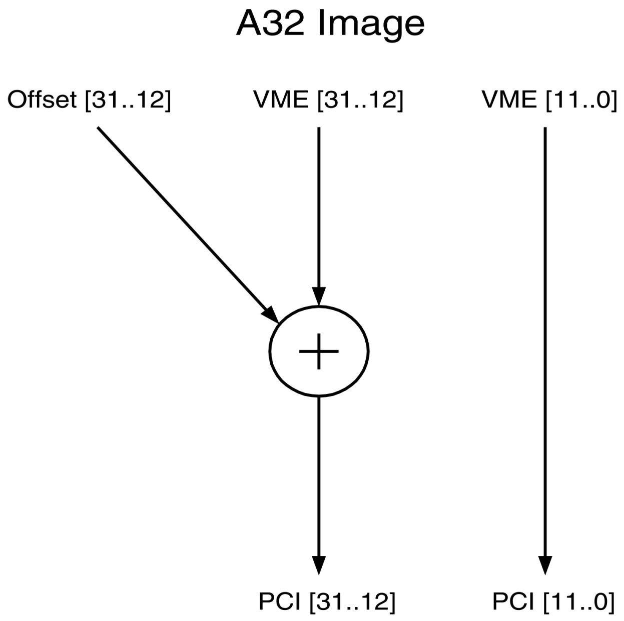
The address space of a VMEbus slave image must not overlap with the address space for the Universe II's control and status registers.

PCI Bus Fields

The PCI bus fields specify how the VMEbus transaction is mapped to the appropriate PCI bus transaction. The translation offset field allows the user to translate the VMEbus address to a different address on the PCI bus. The translation of VMEbus transactions beyond 4 GBytes results in wrap-around to the low portion of the address range.

The PAS field controls generation of the PCI transaction command. The LLRMW bit allows indivisible mapping of incoming VMEbus RMW cycles to the PCI bus via the PCI LOCK# mechanism Refer to VMEbus Read-Modify-Write Cycles (RMW Cycles) on page 62. When the LLRMW bit is set, single cycle reads will always be mapped to single data beat locked PCI transactions. Setting this bit has no effect on non-block writes: they can be coupled or decoupled. However, note that only accesses to PCI Memory Space are decoupled, accesses to I/O or Configuration Space are always coupled.

Figure 2-9: Address Translation Mechanism for VMEbus-to-PCI Bus Transfers



Control Fields

The control fields allow the user to enable a VMEbus slave image (using the EN bit), as well as specify how reads and writes will be processed. At power-up, all images are disabled and are configured for coupled reads and writes.

If the PREN bit is set, the Universe II will prefetch for incoming VMEbus block read cycles. It is the user's responsibility to ensure that prefetched reads are not destructive and that the entire image contains prefetchable resources.

If the PWEN bit is set, incoming write data from the VMEbus is loaded into the RXFIFO (see Posted Writes on page 2-18). Note that posted write transactions can only be mapped to Memory space on the PCI bus. Setting the PAS bit in the PCI fields to I/O or Configuration Space will force all incoming cycles to be coupled independent of this bit.

If the LD64EN bit is set, the Universe II will attempt to generate 64-bit transactions on the PCI bus by asserting REQ64#. The REQ64# line is asserted during the address phase in a 64-bit PCI system, and is the means of determining whether the PCI target is a 64-bit port. If the target asserts ACK64# with DEVSEL#, then the Universe II uses the 64-bit data bus. If the target does not assert ACK64# with DEVSEL#, then the Universe II uses a 32-bit data bus. However, note that use of REQ64# requires extra clocks internally. Therefore, if no 64-bit targets are expected on the PCI bus then performance can be improved by disabling LD64EN on the VMEbus slave images.

Note

In order for a VMEbus slave image to respond to an incoming cycle, the PCI Master Interface must be enabled (bit BM in the PCI_CSR register, Table B-4).

PCI Bus Target Images

The Universe II accepts accesses from the PCI bus with specific programmed PCI target images. Each image opens a window to the resources of the VMEbus and allows the user to control the type of access to those resources. The tables below describe programming for the eight standard PCI bus target images (numbered 0 to 7) by dividing them into VMEbus, PCI bus and Control fields. One special PCI target image separate from the four discussed below is described in “Special PCI Target Image” on page 2-54.

Table 2-8: PCI Bus Fields for the PCI Bus Target Image

Field	Register Bits	Description
base	BS[31:12] or BS[31:16] in LSIx_BS	multiples of 4 or 64 Kbytes (base to bound: maximum of 4 GBytes)
bound	BD[31:12] or BD[31:16] in LSIx_BD	
address space	LAS in LSIx_CTL	Memory or I/O

Table 2-9: VMEbus Fields for the PCI Bus Target Image

Field	Register Bits	Description
translation offset	TO[31:12] or TO[31:16] in LSIx_TO	translates address supplied by PCI master to a specified VMEbus address
maximum data width	VDW in LSIx_CTL	8, 16, 32, or 64 bits
address space	VAS in LSIx_CTL	A16, A24, A32, CR/CSR, User1, User2
mode	SUPER in LSIx_CTL	supervisor or non-privileged
type	PGM in LSIx_CTL	program or data
cycle	VCT in LSIx_CTL	single or block

Table 2-10: Control Fields for PCI Bus Target Image

Field	Register Bits	Description
image enable	EN in LSIx_CTL	enable bit
posted write	PWEN in LSIx_CTL	enable bit

Tundra recommends that the attributes in a target image not be changed while data is enqueued in the Posted Writes FIFO. To ensure data is dequeued from the FIFO, check the TXFE status bit in the MISC_STAT register (Table B-164) or perform a read from that image. If the programming for an image is changed after the transaction is queued in the FIFO, the transaction’s attributes are not changed. Only subsequent transactions are affected by the change in attributes.

PCI Bus Fields

All decoding for VMEbus accesses are based on the address and command information produced by a PCI bus master. The PCI Target Interface claims a cycle if there is an address match and if the command matches certain criteria.

All of the Universe II's eight PCI target images are A32-capable only. The first and fifth of them (for example, PCI target images 0 and 4) have a 4 Kbyte resolution while PCI target images 1 to 3 and 5 to 8 have 64 Kbyte resolution. Typically, image 0 or image 4 would be used for an A16 image since they have the finest granularity.

Caution

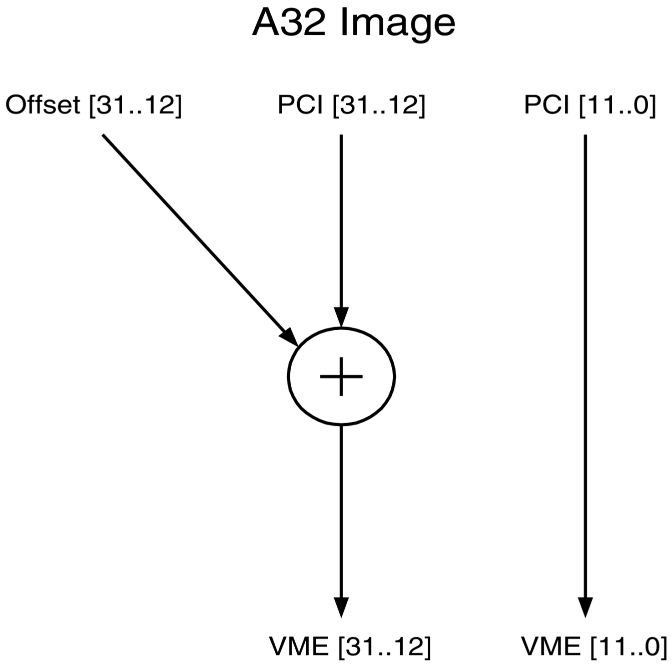
The address space of a VMEbus slave image must not overlap with the address space for the Universe II's control and status registers.

VMEbus Fields

The VMEbus fields map PCI transactions to a VMEbus transaction, causing the Universe II to generate the appropriate VMEbus address, AM code, and cycle type. Some invalid combinations exist within the PCI target image definition fields. For example, A16 and CR/CSR spaces do not support block transfers, and A16 space does not support 64-bit transactions. Note that the Universe II does not attempt to detect or prevent these invalid programmed combinations, and that use of these combinations may cause illegal activity on the VMEbus.

The 21-bit translation offset allows the user to translate the PCI address to a different address on the VMEbus. The figure below illustrates the translation process:.

Figure 2-10: Address Translation Mechanism for PCI Bus to VMEbus Transfers



Translations beyond the 4 Gbyte limit will wrap around to the low address range.

The AM code generated by the Universe II is a function of the VMEbus fields, the data width, and alignment generated by the PCI bus master. For RMW and ADOH cycles, the AM code also depends on the settings for the Special Cycle Generator (see “The Special Cycle Generator” on page 2-44).

The address space, mode, type, and cycle fields control the VMEbus AM code for most transactions. Setting the cycle field to BLT enables the BLT cycle generation. MBLT cycles are generated when the maximum width is set to 64, the BLT bit is set, and the PCI master queues a transaction with at least 64 bits (aligned) of data.

The Universe II provides support for user defined AM codes. The USER_AM register (Table B-166) contains AM codes identified as User1 and User2. The USER_AM register can only be used to generate and accept AM codes 0x10 through 0x1F. These AM codes are designated as USERAM codes in the VMEbus specification. If the user selects one of these two, then the corresponding AM code from the global register is generated on the VMEbus. This approach results in standard single cycle transfers to A32 VMEbus address space independent of other settings in the VMEbus fields.

The VCT bits in the LSIx_CTL registers determine whether or not the VMEbus Master Interface will generate BLT transfers. The VCT bit will only be used if the VAS field is programmed for A24 or A32 space and the VDW bits are programmed for 8, 16, or 32 bits. If VAS bits of the control register are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

Control Fields

The control fields allow the user to enable a PCI target image (the EN bit), as well as specify how writes are processed. If the PWEN bit is set, then the Universe II will perform posted writes when that particular PCI target image is accessed. Posted write transactions are only decoded within PCI Memory space. Accesses from other spaces will result in coupled cycles independent of the setting of the PWEN bit.

Special PCI Target Image

The Universe II provides a special PCI target image located in Memory or I/O space. Its base address is aligned to 64-Mbyte boundaries and its size is fixed at 64 Mbytes (decoded using PCI address lines [31:26]). The Special PCI Target Image is divided into four 16Mbyte regions numbered 0 to 3 (see Figure 2-11). These separate regions are selected with PCI address bits AD [25:24]. For example, if AD[25:24] = 01, then region 1 is decoded. Within each region, the upper 64Kbytes map to VMEbus A16 space, while the remaining portion of the 16 Mbytes maps to VMEbus A24 space. Note that no offsets are provided, so address information from the PCI transaction is mapped directly to the VMEbus.

The general attributes of each region are programmed according to the following tables.

Table 2-11: PCI Bus Fields for the Special PCI Target Image

Field	Register Bits	Description
Base	BS[5:0] in Table B-60 on page 246	64 Mbyte aligned base address for the image
address space	LAS [1:0] in Table B-60 on page 246	Places image in Memory or I/O

Table 2-12: VMEbus Fields for the Special PCI Bus Target Image

Field	Register Bits	Description
maximum data width	VDW in Table B-60 on page 246	separately sets each region for 16 or 32 bits
Mode	SUPER in Table B-60 on page 246	separately sets each region as supervisor or non-privileged
Type	PGM in Table B-60 on page 246	separately sets each region as program or data

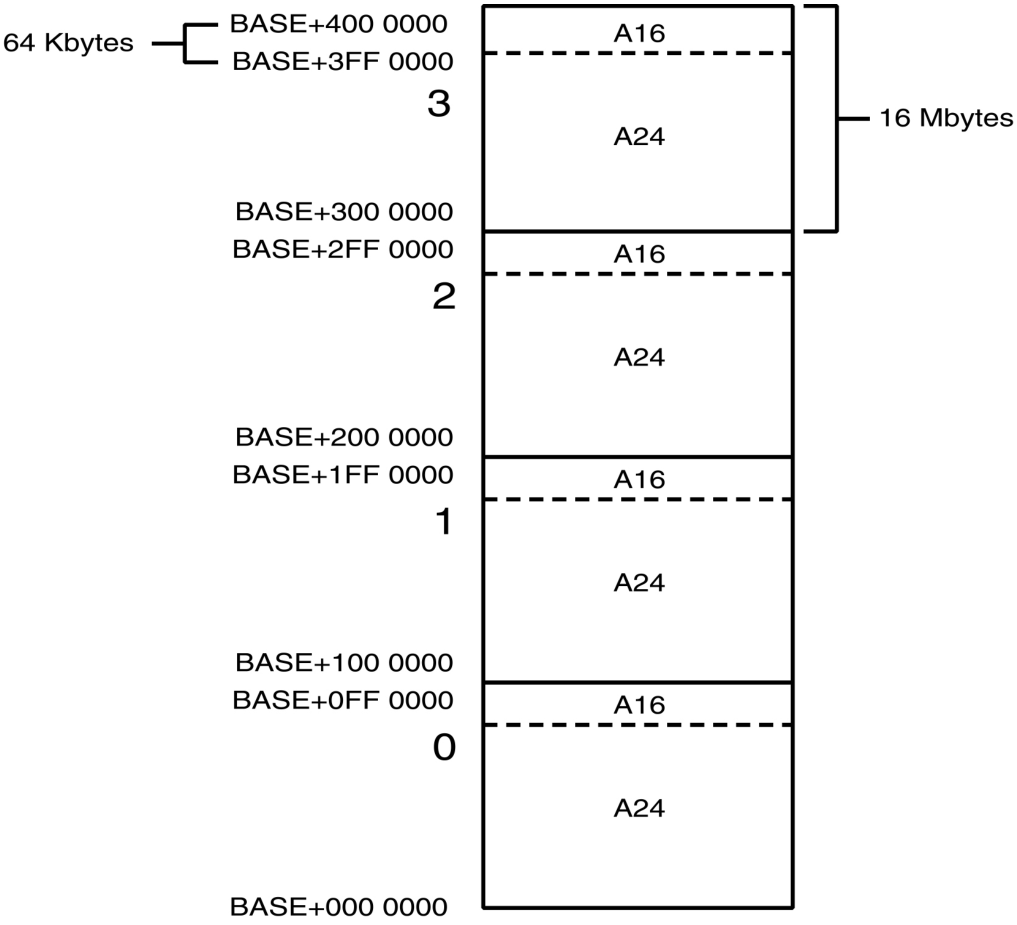
Table 2-13: Control Fields for the Special PCI Bus Target Image

Field	Register Bits	Description
image enable	EN in Table B-60 on page 246	enable bit for the image
posted write	PWEN in Table B-60 on page 246	enable bit for posted writes for the image

The special PCI target image provides access to all of A16 and most of A24 space (all except the upper 64 Kbytes). By using the special PCI target image for A16 and A24 transactions, it is possible to free the eight standard PCI target images (see “PCI Bus Target Images” on page 2-52), which are typically programmed to access A32 space.

Note that some address space redundancy is provided in A16 space. The VMEbus specification requires only two A16 spaces, while the special PCI target image allows for four A16 address spaces.

Figure 2-11: Memory Mapping in the Special PCI Target Image



Bus Error Handling

There are two fundamentally different conditions under which bus errors may occur with the Universe II during coupled cycles or during decoupled cycles. In a coupled transaction, the completion status is returned to the transaction master, which may then take some action. However, in a decoupled transaction, the master is not involved in the data acknowledgment at the destination bus and higher level protocols are required.

The error handling provided by the Universe II is described for both coupled and decoupled transactions below.

Coupled Cycles

During coupled cycles, the Universe II provides immediate indication of an errored cycle to the originating bus. VMEbus-to-PCI transactions terminated with Target-Abort or Master-Abort are terminated on the VMEbus with BERR*. The R_TA or R_MA bits in the PCI_CSR register (Table B-4) are set when the Universe II receives a Target-Abort or Master-Abort. For PCI-to-VMEbus transactions, a VMEbus BERR* received by the Universe II is communicated to the PCI master as a Target-Abort and the S_TA bit is set (Table B-4). No information is logged in either direction nor is an interrupt generated.

Decoupled Transactions

Posted Writes

The Universe II provides the option of performing posted writes in both the PCI Target Channel and the VMEbus Slave Channel. Once data is written into the RXFIFO or TXFIFO by the initiating master (VMEbus or PCI bus respectively), the Universe II provides immediate acknowledgment of the cycle's termination. When the data in the FIFO is written to the destination slave by the Universe II, the Universe II may subsequently receive a bus error instead of a normal termination. The Universe II handles this situation by logging the errored transactions in one of two error logs and generating an interrupt. Each error log (one for VMEbus errors and one for PCI bus errors) is comprised of two registers: one for address and one for command or address space logging.

If the error occurs during a posted write to the VMEbus, the Universe II uses the V_AMERR register (Table B-212) to log the AM code of the transaction (AMERR [5:0]). The state of the IACK* signal is logged in the IACK bit, to indicate whether the error occurred during an IACK cycle. The address of the errored transaction is latched in the V_AERR register (Table B-214 on page 314). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and VERR interrupts are enabled (see Interrupt Handling on page 2-67). The remaining entries of the offending transaction are purged from the FIFO.

If the error occurs during a posted write to the PCI bus, the Universe II uses the L_CMDERR register (Table B-62) to log the command information for the transaction (CMDERR [3:0]). The address of the errored transaction is latched in the L_AERR register (Table B-64). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “Interrupt Handling” on page 2-67).

Under either of the above conditions (VMEbus-to-PCI, or PCI-to-VMEbus), the address that is stored in the log represents the most recent address the Universe II generated before the bus error was encountered. For single cycle transactions, the address represents the address for the actual errored transaction. However, for multi-data beat transactions (block transfers on the VMEbus or burst transactions on the PCI bus) the log only indicates that an error occurred somewhere after the latched address. For a VMEbus block transfer, the logged address will represent the start of the block transfer. In the PCI Target Channel, the Universe II generates block transfers that do not cross 256-byte boundaries, the error will have occurred from the logged address up to the next 256-byte boundary. In the VMEbus Slave Channel, the error will have occurred anywhere from the logged address up to the next burst aligned address.

In the case of PCI-initiated transactions, all data from the errored address up to the end of the initiating transaction is flushed from the TXFIFO. Since the Universe II breaks PCI transactions at 256-byte boundaries (or earlier if the TXFIFO is full), the data is not flushed past this point. If the PCI master is generating bursts that do not cross the 256-byte boundary, then (again) only data up to the end of that transaction is flushed.

In a posted write from the VMEbus, all data subsequent to the error in the transaction is flushed from the RXFIFO. However, the length of a VMEbus transaction differs from the length of the errored PCI bus transaction. For non-block transfers, the length always corresponds to one so only the errored data beat is flushed. However, if an error occurs on the PCI bus during a transaction initiated by a VMEbus block transfer, all data subsequent to the errored data beat in the block transfer is flushed from the RXFIFO. In the case of BLTs, this implies that potentially all data up to the next 256-byte boundary may be flushed. For MBLTs, all data up to the next 2-KByte boundary may be flushed.

Once an error is captured in a log, that set of registers is frozen against further errors until the error is acknowledged. The log is acknowledged and made available to latch another error by clearing the corresponding status bit in the VINT_STAT or LINT_STAT registers. Should a second error occur before the CPU has the opportunity to acknowledge the first error, another bit in the logs is set to indicate this situation (M_ERR bit).

The VMEbus Interface provides auxiliary BERR handler logic. When disabled, the Universe II is responsible for handling BERR. Chapter 3 discusses the function of the auxiliary BERR handler.

Prefetched Reads

In response to a block read from the VMEbus, the Universe II initiates prefetching on the PCI bus (if the VMEbus slave image is programmed with this option, see Slave Image Programming on page 2-49). The transaction generated on the PCI bus is an aligned memory read transaction with multiple data beats extending to the aligned burst boundary (as programmed by PABS in the MAST_CTL register, Table B-162). Once an acknowledgment is given for the first data beat, an acknowledgment is sent to the VMEbus initiator by the assertion of DTACK*. Therefore, the first data beat of a prefetched read is coupled while all subsequent reads in the transaction are decoupled.

If an error occurs on the PCI bus, the Universe II does not translate the error condition into a BERR* on the VMEbus. Indeed, the Universe II does not directly map the error. By doing nothing, the Universe II forces the external VMEbus error timer to expire.

DMA Errors

How the Universe II responds to a bus error during a transfer controlled by the DMA Channel is described in DMA Error Handling on page 2-93.

Parity Errors

The Universe II both monitors and generates parity information using the PAR signal. The Universe II monitors PAR when it accepts data as a master during a read or as a target during a write. The Universe II drives PAR when it provides data as a target during a read or a master during a write. The Universe II also drives PAR during the address phase of a transaction when it is a master and monitors PAR during an address phase when it is the PCI target. In both address and data phases, the PAR signal provides even parity for C/BE#[3:0] and AD[31:0]. If the Universe II is powered up in a 64-bit PCI environment, then PAR64 provides even parity for C/BE#[7:4] and AD[63:32].

The PERESP and SERR_EN bits in the PCI_CS register (Table B-4) determine whether or not the Universe II responds to parity errors. Data parity errors are reported through the assertion of PERR# if the PERESP bit is set. Address parity errors, reported through the SERR# signal, are reported if both PERESP and SERR_EN are set. Regardless of the setting of these two bits, the D_PE (Detected Parity Error) bit in the PCI_CS register is set if the Universe II encounters a parity error as a master or as a target. The DP_D (Data Parity Detected) bit in the same register is only set if parity checking is enabled through the PERESP bit and the Universe II detects a parity error while it is PCI master (for example, it asserts PERR# during a read transaction or receives PERR# during a write).

No interrupts are generated by the Universe II either as a master or as a target in response to parity errors reported during a transaction. Parity errors are reported by the Universe II through assertion of PERR# and by setting the appropriate bits in the PCI_CS register. If PERR# is asserted to the Universe II while it is PCI master, the only action it takes is to set the DP_D. Regardless of whether the Universe II is the master or target of the transaction, and regardless which agent asserted PERR#, the Universe II does not take any action other than to set bits in the PCI_CS register. The Universe II continues with a transaction independent of any parity errors reported during the transaction.

Similarly, address parity errors are reported by the Universe II (if the SERR_EN bit and the PERESP bit are set) by asserting the SERR# signal and setting the S_SERR (Signalled SERR#) bit in the PCI_CS register. Assertion of SERR# can be disabled by clearing the SERR_EN bit in the PCI_CS register. No interrupt is generated, and regardless of whether assertion of SERR# is enabled or not, the Universe II does not respond to the access with DEVSEL#. Typically the master of the transaction times out with a Master-Abort. As a master, the Universe II does not monitor SERR#. It is expected that a central resource on the PCI bus will monitor SERR# and take appropriate action.

Interrupter

The VMEbus Interface has two sources of PCI interrupts: the Universe II chip and the BERR interrupt logic.

The LINT#2, LINT#3, LINT#4, LINT#5, LINT#6, and LINT#7 are not supported by the VMEbus Interface and should not be used. Refer to the following Universe II chip specific material for a detailed description of PCI interrupt handling.

Interrupt Generation

This is the first of two sections in this chapter which describe the Universe II's interrupt capabilities. This section describes the Universe II as a generator of interrupts. The following section, Interrupt Handling on page 2-67, describes the Universe II as an interrupt handler. Each of these sections has subsections which detail interrupt events on the PCI bus and VMEbus (for example, how the Universe II can generate interrupts on the PCI bus and the VMEbus, and how the Universe II can respond to interrupt sources on the PCI bus and the VMEbus).

The Interrupt Channel handles the prioritization and routing of interrupt sources to interrupt outputs on the PCI bus and VMEbus. The interrupt sources are:

- the PCI LINT#[7:0] lines,
- the VMEbus IRQ*[7:1] lines,
- ACFAIL* and SYSFAIL*
- various internal events

These sources can be routed to either the PCI LINT# [7:0] lines or the VMEbus IRQ* [7:1] lines. Each interrupt source is individually maskable and can be mapped to various interrupt outputs. Most interrupt sources can be mapped to one particular destination bus. The PCI sources, LINT#[7:0], can only be mapped to the VMEbus interrupt outputs, while the VMEbus sources, VIRQ[7:1], can only be mapped to the PCI interrupt outputs. Some internal sources (for example, error conditions or DMA activity) can be mapped to either bus.

Figure 2-12: Universe Interrupt Circuitry

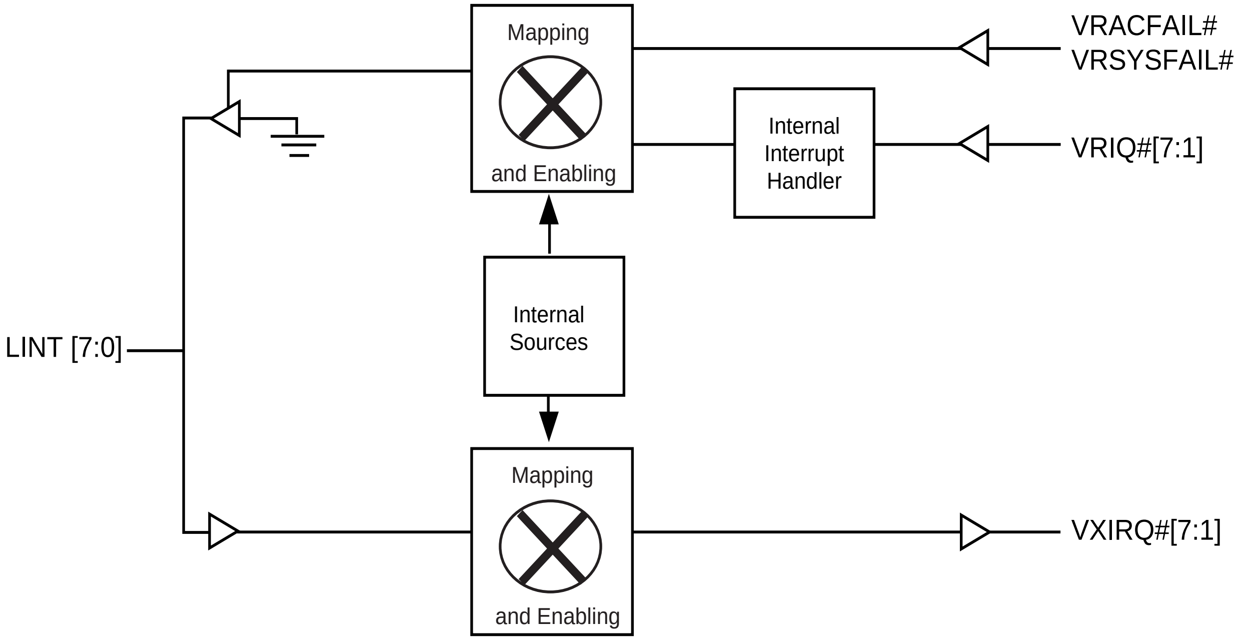


Figure 2-12 above illustrates the circuitry inside the Universe II Interrupt Channel. The PCI hardware interrupts are listed on the left, and the VMEbus interrupt inputs and outputs are on the right. Internal interrupts are also illustrated. The figure shows that the interrupt sources may be mapped and enabled. The Internal Interrupt Handler is a block within the Universe II that detects assertion of the VRIRQ#[7:1] pins and generates the VME IACK through the VME Master. Upon completion of the IACK cycle, the Internal Interrupt Handler notifies the Mapping Block which in turn asserts the local LINT#, if enabled. (Whereas the Internal Interrupt Handler implies a delay between assertion of an interrupt condition to the Universe II and the Universe's mapping of the interrupt, all other interrupt sources get mapped immediately to their destination—assertion of LINT# immediately causes an IRQ, assertion of ACFAIL immediately causes an LINT#, etc.) This is described in more detail in the following sections.

PCI Interrupt Generation

The Universe II expands on the basic PCI specification which permits “single function” devices to assert only a single interrupt line. Eight PCI interrupt outputs provide maximum flexibility, although if full PCI compliancy is required, the user may route all interrupt sources to a single PCI interrupt output.

Note

Only one of the PCI interrupt outputs, LINT#[0], has the drive strength to be fully compliant with the PCI specification. The other seven may require buffering if they are to be routed to PCI compliant interrupt lines. For most applications, however, the drive strength provided should be sufficient.

PCI interrupts may be generated from multiple sources:

- VMEbus sources of PCI interrupts
    - IRQ*[7:1]
    - SYSFAIL*
    - ACFAIL*
- internal sources of PCI interrupts
    - DMA
    - VMEbus bus error encountered
    - PCI Target-Abort or Master-Abort encountered
    - VMEbus ownership has been granted while the VOWN bit is set (see VME Lock Cycles—Exclusive Access to VMEbus Resources on page 2-46)
    - software interrupt
    - mailbox access
    - location monitor access
    - VMEbus IACK cycle performed in response to a software interrupt

Each of these sources may be individually enabled in the LINT_EN register (Table B-112) and mapped to a single LINT# signal through the LINT_MAP0, LINT_MAP1, and LINT_MAP2 registers (Table B-116, Table B-118, and Table B-144). When an interrupt is received on any of the enabled sources, the Universe II asserts the appropriate LINT# pin and sets a matching bit in the LINT_STAT register (Table B-122). See Table 2-15 below for a list of the enable, mapping and status bits for PCI interrupt sources.

Table 2-14: Source, Enabling, Mapping, and Status of PCI Interrupt Output

Interrupt Source	Enable Bit in LINT_EN (Table B-112)	Mapping Field in LINT_MAPx (Table B-116, Table B-118, Table B-144)	Status Bit in LINT_STAT (Table B-114)
ACFAIL*	ACFAIL	ACFAIL	ACFAIL
SYSFAIL*	SYSFAIL	SYSFAIL	SYSFAIL
PCI Software Interrupt	SW_INT	SW_INT	SW_INT
VMEbus Software IACK	SW_IACK	SW_IACK	SW_IACK
VMEbus Error	VERR	VERR	VERR
PCI Target-Abort or Master-Abort	LERR	LERR	LERR
DMA Event	DMA	DMA	DMA
VMEbus Interrupt Input	VIRQ7-1	VIRQ7-1	VIRQ7-1
Location Monitor	LM3-0	LM3-0	LM3-0
Mailbox Access	MBOX3-0	MBOX3-0	MBOX3-0
VMEbus Ownership	VOWN	VOWN	VOWN

The LINT_STAT register shows the status of all sources of PCI interrupts, independent of whether that source has been enabled. This implies that an interrupt handling routine must mask out those bits in the register that do not correspond to enabled sources on the active LINT# pin.

Except for SYSFAIL* and ACFAIL*, all sources of PCI interrupts are edge-sensitive. Enabling of the ACFAIL* or SYSFAIL* sources (ACFAIL and SYSFAIL bits in the LINT_EN register) causes the status bit and mapped PCI interrupt pin to assert synchronously with the assertion of the ACFAIL* or SYSFAIL* source. The PCI interrupt is negated once the ACFAIL or SYSFAIL status bit is cleared. The status bit cannot be cleared if the source is still active. Therefore, if SYSFAIL* or ACFAIL* is still asserted while the interrupt is enabled the interrupt will continue to be asserted. Both of these sources are synchronized and filtered with multiple edges of the 64 MHz clock at their inputs.

All other sources of PCI interrupts are edge-sensitive. Note that the VMEbus source for PCI interrupts actually comes out of the VMEbus Interrupt Handler block and reflects acquisition of a VMEbus STATUS/ID. Therefore, even though VMEbus interrupts externally are level-sensitive as required by the VMEbus specification, they are internally mapped to edge-sensitive interrupts (see VMEbus Interrupt Handling on page 2-68).

The interrupt source status bit (in the LINT_STAT register) and the mapped LINT# pin remain asserted with all interrupts. The status bit and the PCI interrupt output pin are only released when the interrupt is cleared by writing a “one” to the appropriate status bit.

LINT#0 and LINT#1 are supported. LINT#0 maps to PCI INTA#. LINT#1 maps to PCI SERR#.

Auxiliary BERR Interrupt Generation

The VMEbus Interface has an external BERR* circuit which, when enabled is capable of generating a BERR* interrupt to the CPU via PCI INTA#.

The circuit captures the address and address modifier code of the VMEbus BERR* cycle and sets the BERR status bit in the Endian Conversion register located at $D800E in memory. The BERR status bit will in turn generate a PCI interrupt, if enabled. The BERR interrupt is mapped to PCI INTA#. Once a BERR cycle has been captured, it will remain captured until the BERR status bit has been processed by writing a 1 to the BERR status bit. The BERR captured address is available in the BERR address log register located at $D8010. The BERR address modifier code is available at $D8014.

The BERR interrupt is enabled by setting the BERR Latch Enable bit and BERR Interrupt EN bit in the endian Conversion register located at $D800E in memory.

LINT#0 and LINT#1 are supported. LINT#0 maps to PCI INTA#. LINT#1 maps to PCI SERR#.

VMEbus Interrupt Generation

This section details the conditions under which the Universe II generates interrupts to the VMEbus.

Interrupts may be generated on any combination of VMEbus interrupt lines (IRQ*[7:1]) from multiple sources:

- PCI sources of VMEbus interrupts
  - LINT#[7:0]
- Internal sources of VMEbus interrupts
  - DMA
  - VMEbus bus error encountered
  - PCI Target-Abort or Master-Abort encountered
  - Mailbox register access
  - software interrupt

Each of these sources may be individually enabled through the VINT_EN register (Table B-120) and mapped to a particular VMEbus Interrupt level using the VINT_MAPx registers (Table B-124, Table B-126, and Table B-146). Multiple sources may be mapped to any VMEbus level. Mapping interrupt sources to level 0 effectively disables the interrupt.

Once an interrupt has been received from any of the sources, the Universe II sets the corresponding status bit in the VINT_STAT register (Table B-122), and asserts the appropriate VMEbus interrupt output signal (if enabled). When a VMEbus interrupt handler receives the interrupt, it will perform an IACK cycle at that interrupt level. When the Universe II decodes that IACK cycle together with IACKIN* asserted, it provides the STATUS/ID previously stored in the STATID register (Table B-

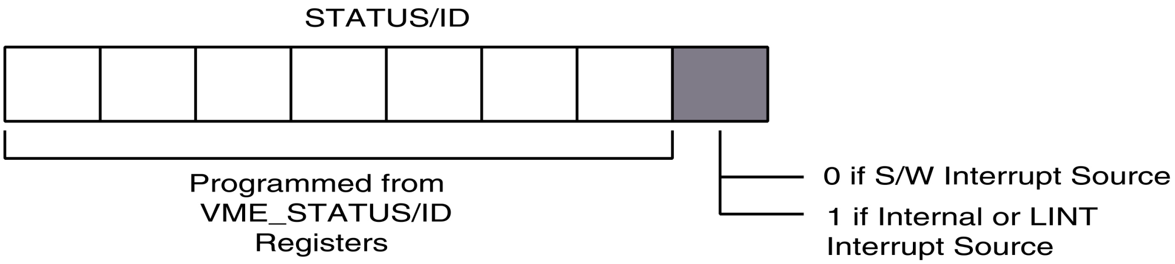
128), unless it is configured as SYSCON in which case it does not monitor IACKIN*. See Table 2-15 below for a list of the enable, mapping and status bits for VMEbus interrupt sources.

Table 2-15: Source, Enabling, Mapping, and Status of VMEbus Interrupt Outputs

Interrupt Source	Enable Bit in VINT_EN (Table B-120)	Mapping Field in VINT_MAPx (Table B-124, Table B-126, Table B-146)	Status Bit in VINT_STAT (Table B-122)
VMEbus Software Interrupt	SW_INT7-1	N/A	SW_INT7-1
VMEbus Error	VERR	VERR (Table B-126)	VERR
PCI Target-Abort or Master-Abort	LERR	LERR (Table B-126)	LERR
DMA Event	DMA	DMA (Table B-126)	DMA
Mailbox Register	MBOX3-0	MBOX3-0 (Table B-146)	MBOX3-0
PCI bus Interrupt Input	LINT7-0	LINT7-0 (Table B-146)	LINT7-0
VMEbus Software Interrupt (mappable)	SW_INT	SW_INT(Table B-126)	SW_INT

For all VMEbus interrupts, the Universe II interrupter supplies a pre-programmed 8-bit STATUS/ID: a common value for all interrupt levels. The upper seven bits are programmed in the STATID register. The lowest bit is cleared if the source of the interrupt was the software interrupt, and is set for all other interrupt sources. If a software interrupt source and another interrupt source are active and mapped to the same VMEbus interrupt level, the Universe II gives priority to the software source.

Figure 2-13: STATUS/ID Provided by Universe II


STATUS/ID
Programmed from VME_STATUS/ID Registers
0 if S/W Interrupt Source
1 if Internal or LINT Interrupt Source

Once the Universe II has provided the STATUS/ID to an interrupt handler during a software initiated VMEbus interrupt, it generates an internal interrupt, SW_IACK. If enabled, this interrupt feeds back to the PCI bus (through one of the LINT# pins) to signal a process that the interrupt started through software has been completed.

All VMEbus interrupts generated by the Universe II are RORA, except for the software interrupts which are ROAK. This means that if the interrupt source was a software interrupt, then the VMEbus interrupt output is automatically negated when the Universe II receives the IACK cycle. However, for any other interrupt, the VMEbus interrupt output remains asserted until cleared by a register access. Writing a “one” to the relevant bit in the VINT_STAT register clears that interrupt source. However, since PCI interrupts are level-sensitive, if an attempt is made to clear the VMEbus interrupt while the LINT# pin is still asserted, the VMEbus interrupt remains asserted. This causes a second interrupt to be generated to the VMEbus. For this reason, a VMEbus interrupt handler should clear the source of the PCI interrupt before clearing the VMEbus interrupt.

Since software interrupts are ROAK, the respective bits in the VINT_STAT register are cleared automatically on completion of the IACK cycle, simultaneously with the negation of the IRQ.

Interrupt Handling

This is the second of two sections in this chapter which describe the Universe II's interrupt capabilities. The previous section, Interrupt Generation on page 2-60, described the interrupt outputs of the Universe II on the PCI bus and the VMEbus. The current section describes how the Universe II responds to interrupt sources. In other words, this section describes the Universe II as an interrupt handler.

This section is broken down as follows:

- PCI Interrupt Handling below explains how the Universe II can respond to hardware interrupts on the PCI bus,
- VMEbus Interrupt Handling on page 2-68 explains how the Universe II can respond to hardware interrupts (or SYSFAIL* and ACFAIL*) on the VMEbus,
- Internal Interrupt Handling on page 2-69 explains how internal states of the Universe II can trigger interrupts.

PCI Interrupt Handling

This section explains how the Universe II can respond to hardware interrupts on the PCI bus.

All eight PCI interrupt lines, LINT#[7:0], can act as interrupt inputs to the Universe II. They are level-sensitive and, if enabled in the VINT_EN register (Table B-120), immediately generate an interrupt to the VMEbus. It is expected that when a VMEbus interrupt handler receives the Universe II's STATUS/ID from the Universe II, the interrupt handler will clear the VMEbus interrupt by first clearing the source of the interrupt on the PCI bus, and then clearing the VMEbus interrupt itself (by writing a “one” to the appropriate bit in the VINT_STAT register, Table B-122).

Note that since PCI interrupts are level-sensitive, if an attempt is made to clear the VMEbus interrupt while the LINT# pin is still asserted, the VMEbus interrupt remains asserted. This causes a second interrupt to be generated to the VMEbus. For this reason, a VMEbus interrupt handler should clear the source of the PCI interrupt before clearing the VMEbus interrupt.

Since all the interrupt sources share a common PCI interrupt, the interrupt handler needs to poll all enabled sources and process them as required by the application. The Universe II based interrupt sources are mapped to the PCI bus by programming the LINT_MAP2/1/0 registers as follows in Table 2-16.

Table 2-16: PCI bus LINT_MAP Registers

Interrupt Source Register Mapping	Corresponding PCI Interrupt
LINT#0	PCI INTA#
LINT#1	PCI SERR#

The LINT#2, LINT#3, LINT#4, LINT#5, LINT#6, and LINT#7 signals are not supported by the VMEbus Interface and should not be used. Refer to the following Universe II chip specific material for a detailed description of PCI interrupt handling.

Do not map any VMEbus interrupt to LINT#(2-7). The VMEbus Interface only supports LINT#0 and LINT#1.

VMEbus Interrupt Handling

This section explains how the Universe II can respond to hardware events on the VMEbus as an interrupt handler.

As a VMEbus interrupt handler, the Universe II can monitor any or all of the VMEbus interrupt levels. It can also monitor SYSFAIL* and ACFAIL*, although IACK cycles are not generated for these inputs. Each interrupt is enabled through the LINT_EN register (Table B-112).

Once enabled, assertion of any of the VMEbus interrupt levels, IRQ[7:1]*, causes the internal interrupt handler circuitry to request ownership of the Universe II's VMEbus Master Interface on the level programmed in the MAST_CTL register (see VMEbus Requester on page 2-10). This interface is shared between several channels in the Universe II: the PCI Target Channel, the DMA Channel, and the Interrupt Channel. The Interrupt Channel has the highest priority over all other channels and, if an interrupt is pending, assumes ownership of the VMEbus Master Interface when the previous owner has relinquished ownership.

The Universe II latches the first interrupt that appears on the VMEbus and begins to process it immediately. Thus if an interrupt at a higher priority is asserted on the VMEbus before BBSY* is asserted the Universe II will perform an interrupt acknowledge for the first interrupt it detected. Upon completion of that IACK cycle, the Universe II will then perform IACK cycles for the higher of any remaining active interrupts.

There may be some latency between reception of a VMEbus interrupt and generation of the IACK cycle. This arises because of the latency involved in the Interrupt Channel gaining control of the VMEbus Master Interface, and because of possible latency in gaining ownership of the VMEbus if the VMEbus Master Interface is programmed for release-when-done. In addition, the Universe II only generates an interrupt on the PCI bus once the IACK cycle has completed on the VMEbus. Because of these combined latencies (time to acquire VMEbus and time to run the IACK cycle), systems should be designed to accommodate a certain worst case latency from VMEbus interrupt generation to its translation to the PCI bus.

When the Universe II receives a STATUS/ID in response to an IACK cycle, it stores that value in one of seven registers. These registers, V1_STATID through V7_STATID (Table B-130 to Table B-142), store the STATUS/ID corresponding to each IACK level (in the STATID field). Once an IACK cycle has been generated and the resulting STATUS/ID is latched, another IACK cycle will not be run on that level until the level has been re-armed by writing a "one" to the corresponding status bit in the VINT_STAT register (Table B-122). If other interrupts (at different levels) are pending while the interrupt is waiting to be re-armed, IACK cycles are run on those levels in order of priority and the STATUS/IDs stored in their respective registers.

Once the IACK cycle is complete and the STATUS/ID stored, an interrupt is generated to the PCI bus on one of LINT#[7:0] depending on the mapping for that VMEbus level in the LINT_MAP0 register. The interrupt is cleared and the VMEbus interrupt level is re-armed by clearing the correct bit in the LINT_STAT register.

Bus Error During VMEbus IACK Cycle

A bus error encountered on the VMEbus while the Universe II is performing an IACK cycle is handled by the Universe II in two ways. The first is through the error logs in the VMEbus Master Interface. These logs store address and command information whenever the Universe II encounters a bus error on the VMEbus (see Bus Error Handling on page 2-57f the error occurs during an IACK cycle, the IACK# bit is set in the V_AMERR register (Table B-212). The VMEbus Master Interface also generates an internal interrupt to the Interrupt Channel indicating a VMEbus error occurred. This internal interrupt can be enabled and mapped to either the VMEbus or PCI bus.

As well as generating an interrupt indicating an error during the IACK cycle, the Universe II also generates an interrupt as though the IACK cycle completed successfully. If an error occurs during the fetching of the STATUS/ID, the Universe II sets the ERR bit in the Vx_STATID register (Table B-130 to Table B-142), and generates an interrupt on the appropriate LINT# pin (as mapped in the LINT_MAP0 register, Table B-122). The PCI resource, upon receiving the PCI interrupt, is expected to read the STATUS/ID register, and take appropriate actions if the ERR bit is set. Note that the STATUS/ID cannot be considered valid if the ERR bit is set in the STATUS/ID register.

It is important to recognize that the IACK cycle error may generate two PCI interrupts: one through the VMEbus master bus error interrupt and another through the standard PCI interrupt translation. Should an error occur during acquisition of a STATUS/ID, the VINT_STAT register (Table B-122) will show that both VIRQx, and VERR are active.

Internal Interrupt Handling

The Universe II’s internal interrupts are routed from several processes in the device. There is an interrupt from the VMEbus Master Interface to indicate a VMEbus error, another from the PCI Master Interface to indicate an error on that bus, another from the DMA to indicate various conditions in that channel, along with several others as indicated in Table 2-17 below. Table 2-17 shows to which bus each interrupt source may be routed (some sources may be mapped to both buses, but we recommend that you map interrupts to a single bus).

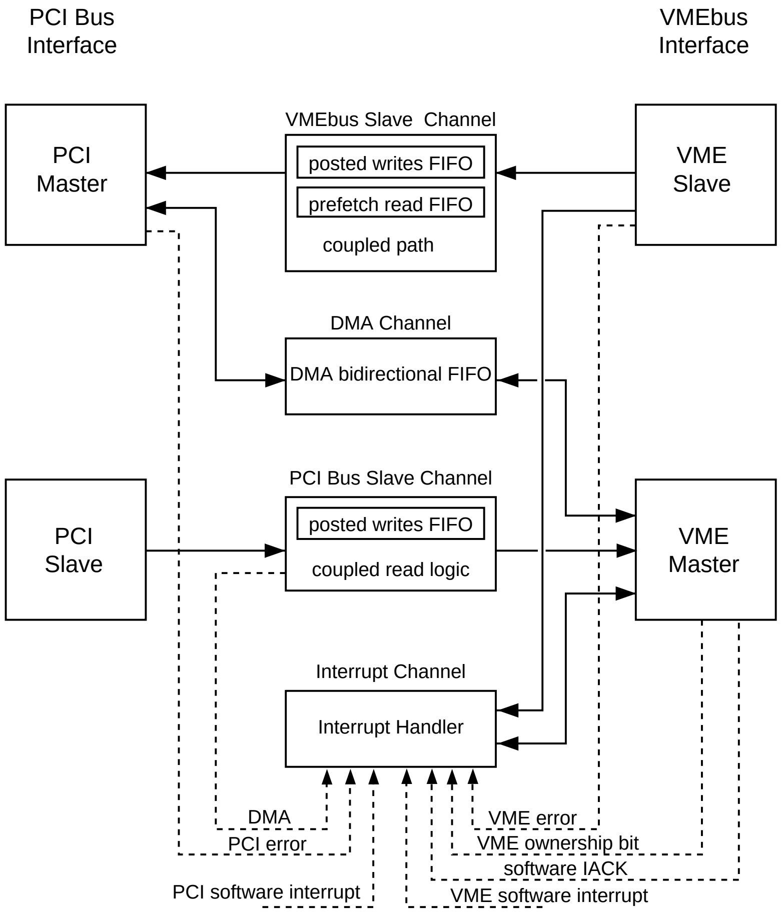
Table 2-17: Internal Interrupt Routing

Interrupt Source	May be Routed to:	
	VMEbus	PCI Bus
PCI s/w interrupt		√
VMEbus s/w interrupt	√	
IACK cycle complete for s/w interrupt		√
DMA event	√	√
Mailbox access	√	√
Location monitor		√
PCI Target-Abort or Master-Abort	√	√
VMEbus bus error	√	√
VMEbus bus ownership granted		√

Figure 2-14 shows the sources of interrupts, and the interfaces from which they originate.

Interrupt handling for each one of these sources is described in the following subsections.

Figure 2-14: Sources of Internal Interrupts



VMEbus and PCI Software Interrupts

It is possible to interrupt the VMEbus and the PCI bus through software. These interrupts may be triggered by writing a “one” to the respective enable bits.

Interrupting the VMEbus through software

There are two methods of triggering software interrupts on the VMEbus. The second method is provided for compatibility with the Universe I.

1. The first method for interrupting the VMEbus through software involves writing “one” to one of the SW_INT7-1 bits in the VINT_EN register (Table B-120) while the mask bit is zero. This causes an interrupt to be generated on the corresponding IRQ7-1 line. That is, setting the SW_INT1 bit triggers VXIRQ1, setting the SW_INT2 bit triggers VXIRQ2, etc.
2. The second method for interrupting the VMEbus through software involves an extra step. Writing a “one” to the SW_INT bit in the VINT_EN register when this bit is “zero” (Table B-120) triggers one (and only one) interrupt on the VMEbus on the level programmed in the VINT_MAP1 register (Table B-126). Notice that this method requires that the user specify in the VINT_MAP1 register to which line the interrupt is to be generated. When the SW_INT interrupt (method 2) is active at the same level as one of SW_INT7-1 interrupts (method 1), the SW_INT interrupt (method 2) takes priority. While this interrupt source is active, the SW_INT status bit in the VINT_STAT register is set.

With both methods, the mask bit (SW_INTx or SW_INT) in the VINT_EN register must be zero in order for writing “one” to the bit to have any effect.

Regardless of the software interrupt method used, when an IACK cycle is serviced on the VMEbus, the Universe II can be programmed to generate an interrupt on the PCI bus by setting the SW_IACK enable bit in the LINT_EN register (see Software IACK Interrupt on page 2-72).

Interrupting the PCI bus through software

On the PCI bus, there is only one method of directly triggering a software interrupt. (This method is analogous to the second method described in the previous section.) Causing a “zero” to “one” transition in the SW_INT in the LINT_EN (Table B-112) register generates an interrupt to the PCI bus. While this interrupt source is active, the SW_INT status bit in LINT_STAT is set. The SW_INT field in the LINT_MAP1 register (Table B-118) determines which interrupt line is asserted on the PCI interface.

Termination of software interrupts

Any software interrupt may be cleared by clearing the respective bit in the VINT_EN or LINT_EN register. However, this method is not recommend for software VME bus interrupts because it may result in a spurious interrupt on that bus. That is, the Universe II will then not respond to the interrupt handler's IACK cycle, and the handler will be left without a STATUS/ID for the interrupt.

Since the software interrupt is edge-sensitive, the software interrupt bit in the VINT_EN or LINT_EN register should be cleared any time between the last interrupt finishing ant the generation of another interrupt. It is recommended that the appropriate interrupt handler clear this bit once it has completed its operations. Alternatively, the process generating a software interrupt could clear this bit before re-asserting it.

Software interrupts on the VMEbus have priority over other interrupts mapped internally to the same level on the VMEbus. When a VMEbus interrupt handler generates an IACK cycle on a level mapped to both a software interrupt and another interrupt, the Universe II always provides the STATUS/ID for the software interrupt (bit zero of the Status/ID is cleared). If there are no other active interrupts on that level, the interrupt is automatically cleared upon completion of the IACK cycle (since software interrupts are ROAK).

While the software interrupt STATUS/ID has priority over other interrupt sources, the user can give other interrupt sources priority over the software interrupt. This is done by reading the LINT_STAT register (Table B-122) when handling a Universe II interrupt. This register indicates all active interrupt sources. Using this information, the interrupt handler can then handle the interrupt sources in any system-defined order.

Software IACK Interrupt

The Universe II generates an internal interrupt when it provides the software STATUS/ID to the VMEbus. This interrupt can only be routed to a PCI interrupt output. A PCI interrupt will be generated upon completion of an IACK cycle that had been initiated by the Universe II's software interrupt if:

- the SW_IACK bit in the LINT_EN register (Table B-112) is set, and
- the SW_IACK field in the LINT_MAP1 register (Table B-118) is mapped to a corresponding PCI interrupt line.

This interrupt could be used by a PCI process to indicate that the software interrupt generated to the VMEbus has been received by the slave device and acknowledged.

Like other interrupt sources, this interrupt source can be independently enabled through the LINT_EN register (Table B-112) and mapped to a particular LINT# pin using the LINT_MAP1 register (Table B-118). A status bit in the LINT_STAT register (Table B-122) indicates when the interrupt source is active, and is used to clear the interrupt once it has been serviced.

VMEbus Ownership Interrupt

The VMEbus ownership interrupt is generated when the Universe II acquires the VMEbus in response to programming of the VOWN bit in the MAST_CTL register (Table B-160). This interrupt source can be used to indicate that ownership of the VMEbus is ensured during an exclusive access (see “VME Lock Cycles—Exclusive Access to VMEbus Resources”on page 2-46). The interrupt is cleared by writing a one to the matching bit in the LINT_STAT register (Table B-122).

DMA Interrupt

The DMA module provides six possible interrupt sources:

- if the DMA is stopped (INT_STOP),
- if the DMA is halted (INT_HALT),
- if the DMA is done (INT_DONE),
- for PCI Target-Abort or Master-Abort (INT_LERR),
- for VMEbus errors (INT_VERR), or
- if there is a PCI protocol error or if the Universe II is not enabled as PCI master (INT_P_ERR).

All of these interrupt sources are OR’ed to a single DMA interrupt output line. When an interrupt comes from the DMA module, software must read the DMA status bits (Table B-108) to discover the originating interrupt source. The DMA interrupt can be mapped to either the VMEbus or one of the PCI interrupt output lines. See “DMA Interrupts” on page 2-92.

Mailbox Register Access Interrupts

The Universe II can be programmed to generate an interrupt on the PCI bus and/or the VMEbus when any one of its mailbox registers is accessed (see Mailbox Registers on page 2-105). The user may enable or disable an interrupt response to the access of any mailbox register (Table B-112). Each register access may be individually mapped to a specific interrupt on the PCI bus (LINT_MAP2, Table B-144) and/or the VMEbus (VINT_MAP2, Table B-146). The status of the PCI interrupt and the VMEbus are recorded in the LINT_STAT (Table B-114) and VINT_STAT registers (Table B-122), respectively.

Location Monitors

The Universe II can be programmed to generate an interrupt on the PCI bus when one of its four location monitors is accessed (see Location Monitors on page 2-22).

In order for an incoming VMEbus transaction to activate the location monitor of the Universe II, the location monitor must be enabled, the access must be within 4 kbytes of the location monitor base address (LM_BS, Table B-202), and it must be in the specified address space.

When an access to a location monitor is detected, an interrupt may be generated on the PCI bus (if the location monitor is enabled). There are four location monitors:

- VA[4:3] = 00 selects Location Monitor 1
- VA[4:3] = 01 selects Location Monitor 2
- VA[4:3] = 10 selects Location Monitor 3
- VA[4:3] = 11 selects Location Monitor 4

The user may enable or disable an interrupt response to the access of any location monitor with bits in the LINT_EN register (Table B-112). Access to each location monitor may be individually mapped to a specific interrupt on the PCI bus (LINT_MAP2, Table B-144)—not to the VMEbus bus. The status of the PCI interrupt is logged in (LMn bit of the LINT_STAT, Table B-114).

PCI and VMEbus Error Interrupts

Interrupts from VMEbus errors, PCI Target-Aborts or Master-Aborts are generated only when bus errors arise during decoupled writes. The bus error interrupt (from either a PCI or VMEbus error) can be mapped to either a VMEbus or PCI interrupt output line.

VME64 Auto-ID

The Universe II includes a power-up option for participation in the VME64 Auto-ID process. When this option is enabled, the Universe II generates a level 2 interrupt on the VMEbus before release of SYSFAIL*. When the level 2 IACK cycle is run by the system Monarch, the Universe II responds with the Auto-ID Status/ID, 0xFE, and enables access to a CR/CSR image at base address 0x00_0000.

When the Monarch detects an Auto-ID STATUS/ID on level 2, it is expected to access the enabled CR/CSR space of the interrupter. From there it completes identification and configuration of the card. The Monarch functionality is typically implemented in software on one card in the VMEbus system. See Automatic Slot Identification on page 2-26.

DMA Controller

The Universe II has a DMA controller for high performance data transfer between the PCI bus and VMEbus. It is operated through a series of registers that control the source and destination for the data, length of the transfer and the transfer protocol to be used. There are two modes of operation for the DMA: Direct Mode, and Linked List Mode.   In direct mode, the DMA registers are programmed directly by the external PCI master. In linked list mode, the registers are loaded from PCI memory by the Universe II, and the transfer described by these registers is executed. A block of DMA registers stored in PCI memory is called a command packet. A command packet may be linked to another command packet, such that when the DMA has completed the operations described by one command packet, it automatically moves on to the next command packed in the linked-list of command packets.

This section is broken into the following major sub-sections

- DMA Registers Outline, below, describes in detail how the DMA is programmed from a register perspective.
- Direct Mode Operation on page 2-81 describes how to operate the DMA when directly programming the DMA registers.
- Linked-List Operation on page 2-83 describes how to operate the DMA when a linked-list of command packets describing DMA transfers is stored in PCI memory.
- FIFO Operation and Bus Ownership on page 2-89 describes internally how the DMA makes use of its FIFO and how this affects ownership of the VMEbus and PCI bus.
- DMA Interrupts on page 2-92 describes the interrupts generated by the DMA
- Interactions with Other Channels on page 2-92 discusses the relations between the DMA Channel and the other data channels.
- DMA Error Handling on page 2-93 describes how to handle errors encountered by the DMA

DMA Registers Outline

The DMA registers reside in a block starting at offset 0x200. They describe a single DMA transfer: where to transfer data from; where to transfer data to; how much data to transfer; and the transfer attributes to use on the PCI bus and VMEbus. A final register contains status and control information for the transfer. While the DMA is active, the registers are locked against any changes so that any writes to the registers will have no impact.

In direct-mode operation, these registers would be programmed directly by the user. In linked-list operation, they are repeatedly loaded by the Universe II from command packets residing in PCI memory until the end of the linked-list is reached (see Linked-List Operation on page 2-83).

Source and Destination Addresses

The source and destination addresses for the DMA reside in two registers: the DMA PCI bus Address Register (DLA register, Table B-102), and the DMA VMEbus Address Register (DVA register in Table B-104). The determination of which is the source address, and which is the destination is made by the L2V bit in the DCTL register (Table B-98). When set, the DMA transfers data from the PCI to the VMEbus. Hence DLA becomes the PCI source register and DVA becomes the VMEbus destination register. When cleared, the DMA transfers data from the VMEbus-to-PCI bus and DLA becomes the PCI destination register; DVA becomes the VMEbus source register.

The PCI address may be programmed to any byte address in PCI Memory space. It cannot transfer to or from PCI I/O or Configuration spaces.

The VMEbus address may also be programmed to any byte address, and can access any VMEbus address space from A16 to A32 in supervisory or non-privileged space, and data or program space. The setting of address space, A16, A24 or A32, is programmed in the VAS field of the DCTL register (Table B-98). The sub-spaces are programmed in the PGM and SUPER fields of the same register.

Note

Although the PCI and VMEbus addresses may be programmed to any byte aligned address, they must be 8-byte aligned to each other (for example, the low three bits of each must be identical). If not programmed with aligned source and destination addresses and an attempt to start the DMA is made, the DMA will not start, it will set the protocol error bit (P_ERR) in the DCSR register (Table B-108), and if enabled to, generate an interrupt. Linked-list operations will cease.

In direct mode the user must reprogram the source and destination address registers (DMA, DLA) before each transfer. These registers are not updated in direct mode. In linked-list mode, these registers are updated by the DMA when (and only when) the DMA is stopped, halted, or at the completion of processing a command packet. If read during DMA activity, the y will return the number of bytes remaining to transfer on the PCI side. All of the DMA registers are locked against any changes by the user while the DMA is active. When stopped due to an error situation, the DLA and DVA registers should not be used, but the DTBC is valid (see “DMA Error Handling” on page 2-93 for details). At the end of a successful linked-list transfer, the DVA and DLA registers will point to the next address at the end of the transfer block, and the DTBC register will be zero.

Transfer Size

The DMA may be programmed through the DMA Transfer Byte Count register (DTBC register in Table B-100) to transfer any number of bytes from 1 byte to 16 MBytes. There are no alignment requirements to the source or destination addresses. Should the width of the data turnovers (8- through 64-bit on VMEbus and 32- or 64-bit on PCI) not align to the length of the transfer or the source/destination addresses, the DMA will insert transfers of smaller width on the appropriate bus. For example, if a 15-byte transfer is programmed to start at address 0x1000 on the VMEbus, and the width is set for D32, the DMA will perform three D32 transfers, followed by a D16 transfer, followed by a D08 transfer. The Universe II does not generate unaligned transfers. On a 32-bit PCI bus, if the start address was 0x2000, the DMA would generate three data beats with all byte lanes enabled, and a fourth with three byte lanes enabled.

The DTBC register is not updated while the DMA is active (indicated by the ACT bit in the DGCS register). At the end of a transfer it will contain zero. However, if stopped by the user (via the STOP bit in the DGCS register) or the DMA encounters an error, the DTBC register contains the number of bytes remaining to transfer on the source side. See DMA Error Handling on page 141.

Starting the DMA while DTBC=0 will result in one of two situations. If the CHAIN bit in the DGCS register (Table B-108) is not set, the DMA will not start; it will perform no action. If the CHAIN bit is set, then the DMA loads the DMA registers with the contents of the command packet pointed to by the DCPP register (Table B-106 on page 265), and starts the transfers described by that packet. Note that the DCPP[31:5] field of the DCPP register implies that the command packets be 32-byte aligned (bits 4:0 of this register must be 0).

Transfer Data Width

The VMEbus and PCI bus data widths are determined by three fields in the DCTL register (Table B-98). These fields affect the speed of the transfer. They should be set for the maximum allowable width that the destination device is capable of accepting.

On the VMEbus, the DMA supports the following data widths:

- D08(EO)
- D16
- D16BLT
- D32
- D64
- D32BLT
- D64BLT (MBLT)

The width of the transfer is set with the VDW field in the DCTL register. The VCT bit determines whether or not the Universe II VMEbus Master will generate BLT transfers. The value of this bit only has meaning if the address space is A24 or A32 and the data width is not 64 bits. If the data width is 64 bits the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The Universe II may perform data transfers smaller than that programmed in the VDW field in order to bring itself into alignment with the programmed width. For example if the width is set for D32 and the starting VMEbus address is 0x101, the DMA will perform a D08 cycle followed by a D16 cycle. Only once it has achieved the alignment set in the VDW field does it start D32 transfers. At the end of the transfer, the DMA will also have to perform more low-width transfers if the last address is not aligned to VDW. Similarly, if the VCT bit is set to enable block transfers, the DMA may perform non-block transfers to bring itself into alignment.

On the PCI bus, the DMA provides the option of performing 32- or 64-bit PCI transactions through the LD64EN bit in the DCTL register (Table B-98). If the Universe II has powered-up on a 32-bit bus (see “Power-up Option Descriptions” on page 2-114), this bit will have no effect. If powered-up on a 64-bit bus, this bit can provide some performance improvements when accessing 32-bit targets on that bus. Following the PCI specification, before a 64-bit PCI initiator starts a 64-bit transaction, it engages in a protocol with the intended target to determine if it is 64-bit capable. This protocol typically consumes one clock period. To save bandwidth, the LD64EN bit can be cleared to bypass this protocol when it is known that the target is only 32-bit capable

DMA Command Packet Pointer

The DMA Command Packet Pointer (DCPP in Table B-106) points to a 32-byte aligned address location in PCI Memory space that contains the next command packet to be loaded once the transfer currently programmed into the DMA registers has been successfully completed. When it has been completed (or the DTBC register is zero when the GO bit is set) the DMA reads the 32-byte command packet from PCI memory and executes the transfer it describes.

DMA Control and Status

The DMA General Control/Status Register (DGCS in Table B-108) contains a number of fields that control initiation and operation of the DMA as well as actions to be taken on completion.

DMA Initiation

Once all the parameters associated with the transfer have been programmed (source/destination addresses, transfer length and data widths, and if desired, linked lists enabled), the DMA transfer is started by setting the GO bit in the DGCS register. This causes the DMA first to examine the DTBC register. If it is non-zero, it latches the values programmed into the DCTL, DTBC, DLA, and DVA registers and initiates the transfer programmed into those registers. If DTBC=0, it checks the CHAIN bit in the DGCS register and if that bit is cleared it assumes the transfer to have completed and stops. Otherwise, if the CHAIN bit is set, it loads into the DMA registers the command packet pointed to by the DCPP register and initiates the transfer described there.

If the GO bit is set, but the Universe II has not been enabled as a PCI master with the BM (bus master enable) bit in the PCI_CSR register, or if the DVA and DLA contents are not 64-bit aligned to each other, the transfer does not start, a protocol error is indicated by the P_ERR bit in the DGCS register and, if enabled, an interrupt is generated.

If the DMA has been terminated for some reason (stopped, halted, or error), all DMA registers contain values indicating where the DMA terminated. Once all status bits have been cleared, the DMA may be restarted from where it left off by simply setting the GO bit. The GO bit will only have an effect if all status bits have been cleared. These bits include STOP, HALT, DONE, LERR, VERR, and P_ERR; all in the DGCS register. These bits are all cleared by writing “one” to them, either before or while setting the GO bit.

The GO bit always returns a zero when read independent of the DMA's current state. Clearing the bit has no impact at any time. The ACT bit in the DGCS register indicates whether the DMA is currently active. It is set by the DMA once the GO bit is set, and cleared when the DMA is idle. Generally, when the ACT bit is cleared, one of the other status bits in the DGCS register is set (DONE, STOP, HALT, LERR, VERR, or P_ERR), indicating why the DMA is no longer active.

DMA VMEbus Ownership

Two fields in the DGCS register determine how the DMA will share the VMEbus with the other two potential masters in the Universe II (PCI Target Channel, and Interrupt Channel), and with other VMEbus masters on the bus. These fields are: VON and VOFF.

VON affects how much data the DMA will transfer before giving the opportunity to another master (either the Universe II or an external master) to assume ownership of the bus. The VON counter is used to temporarily stop the DMA from transferring data once a programmed number of bytes have been transferred (256 bytes, 512 bytes, 1K, 2K, 4K, 8K, or 16K). When performing MBLT transfers on the VMEbus, the DMA will stop performing transfers within 2048 bytes after the programmed VON limit has been reached. When not performing MBLT transfers, the DMA will stop performing transfers within 256 bytes once the programmed limit has been reached. When programmed for Release-When-Done operation, the Universe II will perform an early release of BBSY* when the VON counter reaches its programmed limit. VON may be disabled by setting the field to zero. When set as such, the DMA will continue transferring data as long as it is able.

There are other conditions under which the DMA may relinquish bus ownership. See "FIFO Operation and Bus Ownership" on page 2-89 for details on the VMEbus request and release conditions for the DMA.

VOFF affects how long the DMA will wait before re-requesting the bus after the VON limit has been reached. By setting VOFF to zero, the DMA will immediately re-request the bus once the VON boundary has been reached. Since the DMA operates in a round-robin fashion with the PCI Target Channel, and in a priority fashion with the Interrupt Channel, if either of these channels require ownership of the VMEbus, they will receive it at this time.

VOFF is only invoked when VMEbus tenure is relinquished due to encountering the VON boundary. When the VMEbus is released due to other conditions (e.g., the DMAFIFO has gone full while reading from the VMEbus), it will be re-requested as soon as that condition is cleared. The VOFF timer can be programmed to various time intervals from 0μs to 1024μs. See "FIFO Operation and Bus Ownership" on page 2-89 for details on the VMEbus request and release conditions for the DMA.

See "Interactions with Other Channels" on page 2-92 for information on other mechanisms which may delay the DMA Channel from acquiring the VMEbus or the PCI bus.

DMA Completion and Termination

Normally, the DMA will continue processing its transfers and command packets until either it completes everything it has been requested to, or it encounters an error. There are also two methods for the user to interrupt this process and cause the DMA to terminate prematurely: Stop and Halt. Stop causes the DMA to terminate immediately, while halt causes the DMA to terminate when it has completed processing the current command packet in a linked list.

When the STOP_REQ bit in the DGCS register is set by the user, it tells the DMA to cease its operations on the source bus immediately. Remaining data in the FIFO continues to be written to the destination bus until the FIFO is empty. Once the FIFO is empty, the STOP bit in the same register is set and, if enabled, an interrupt generated. The DMA registers will contain the values that the DMA stopped at: the DTBC register contains the number of bytes remaining in the transfer, the source and destination address registers contain the next address to be read/written, the DCPP register contains the next command packet in the linked-list, and the DCTL register contains the transfer attributes.

If read transactions are occurring on the VMEbus, then setting a stop request can be affected by the VOFF timer. If the STOP_REQ bit is set while the DMA is lying idle waiting for VOFF to expire before recommencing reads, then the request remains pending until the VOFF timer has expired and the bus has been granted.

Halt provides a mechanism to interrupt the DMA at command packet boundaries during a linked-list transfer. In contrast, a stop requests the DMA to be interrupted immediately, while halt takes effect only when the current command packet is complete. A halt is requested of the DMA by setting the HALT_REQ bit in the DGCS register. This causes the DMA to complete the transfers defined by the current contents of the DMA registers and, if the CHAIN bit is set, load in the next command packet. The DMA then terminates, the HALT bit in the DGCS register is set, and, if enabled, an interrupt generated.

After a stop or halt, the DMA can be restarted from the point it left off by setting the GO bit; but before it can be re-started, the STOP and HALT bits must both be cleared.

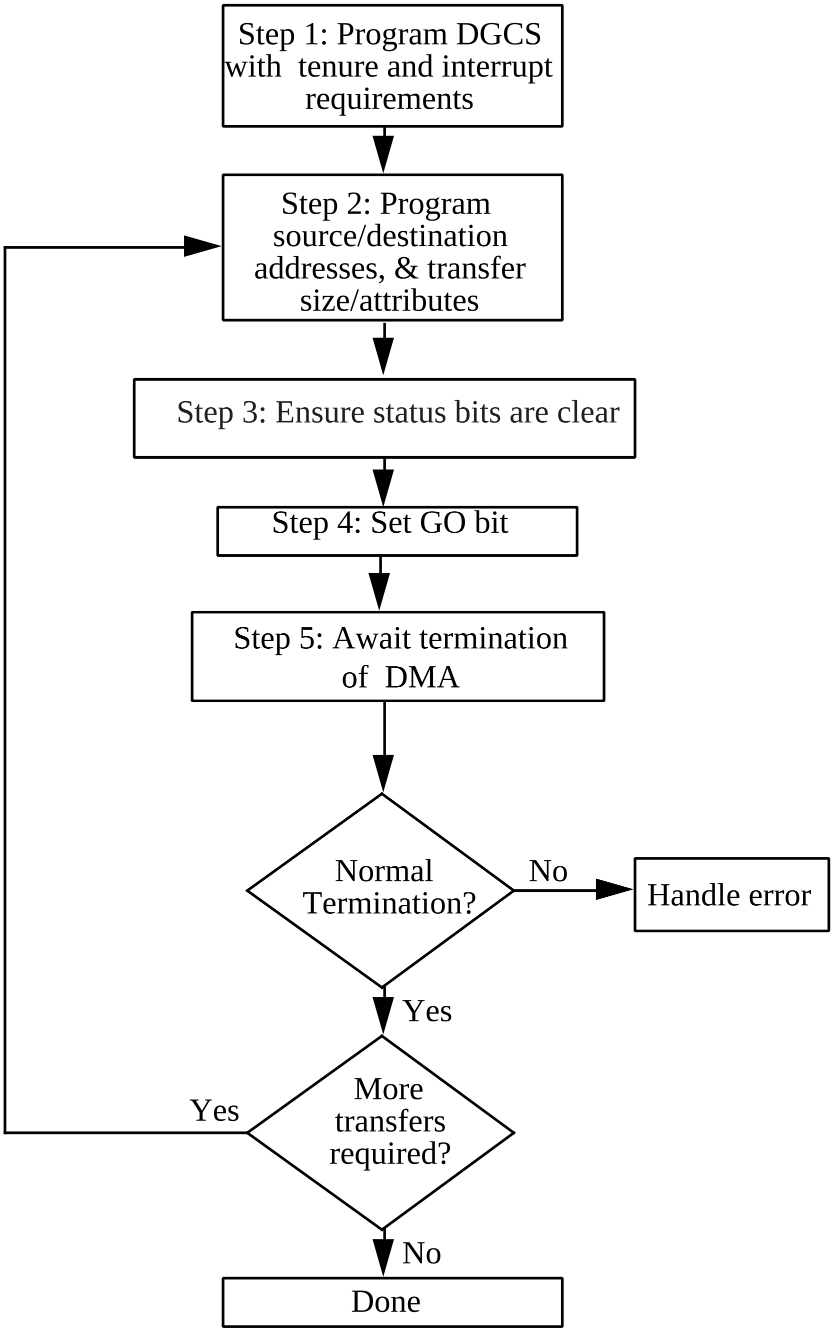
Regardless of how the DMA stops—whether normal, bus error or user interrupted—the DMA will indicate in the DGCS register why it stopped. The STOP and HALT bits get set in response to a stop or halt request. The DONE bit gets set when the DMA has successfully completed the DMA transfer, including all entries in the linked-list if operating in that mode. There are also three bits that are set in response to error conditions: LERR in the case of Target-Abort encountered on the PCI bus; VERR in the case of a bus error encountered on the VMEbus; and P_ERR in the case that the DMA has not been properly programmed (the DMA was started with the BM bit in the PCI_CSR register not enabled, or the DLA and DVA registers were not 64-bit aligned, (see “Source and Destination Addresses” on page 2-76).   Before the DMA can be restarted, each of these status bits must be cleared.

When the DMA terminates, an interrupt may be generated to VMEbus or PCI bus. The user has control over which DMA termination conditions will cause the interrupt through the INT_STOP, INT_HALT, INT_DONE, INT_LERR, INT_VERR, and INT_P_ERR bits in the DGCS register.

Direct Mode Operation

When operated in direct mode, the Universe II DMA is set through manual register programming. Once the transfer described by the DVA, DLA, DTBC and DCTL registers has been completed, the DMA sits idle awaiting the next manual programming of the registers. Figure 2-16 describes the steps involved in operating the DMA in direct mode.

Figure 2-15: Direct Mode DMA Transfers


Step 1: Program DGCS with tenure and interrupt requirements
Step 2: Program source/destination addresses, & transfer size/attributes
Step 3: Ensure status bits are clear
Step 4: Set GO bit
Step 5: Await termination of DMA
Normal Termination?
No
Handle error
Yes
More transfers required?
Yes
No
Done

In Step 1, the DGCS register is set up: the CHAIN bit is cleared, VON and VOFF are programmed with the appropriate values for controlling DMA VMEbus tenure, and the interrupt bits (INT_STOP, INT_HALT, INT_DONE, INT_LERR, INT_VERR, and INT_P_ERR) are programmed to enable generation of interrupts based on DMA termination events. DMA interrupt enable bits in the LINT_EN or VINT_EN bits should also be enabled as necessary (see "PCI Interrupt Generation" on page 2-61 and "VMEbus Interrupt Generation" on page 2-64 for details on generating interrupts).

In Step 2, the actual transfer is programmed into the DMA: source and destination start addresses into the DLA and DVA registers, transfer count into the DTBC register, and transfer width, direction and VMEbus address space into the DCTL register. These should be reprogrammed after each transfer.

In Step 3, ensure that if any status bits (DONE, STOP, HALT, LERR, VERR, or P_ERR) remain set from a previous transfer they are cleared. P_ERR must not be updated at the same time as Step 4, otherwise the P_ERR that may be generated by setting GO may be missed (see Step 4). These bits may be cleared as part of Step 1.

In Step 4, with the transfer programmed, the GO bit in DGCS must be set. If the DMA has been improperly programmed, either because the BM bit in the PCI_CSR has not been set to enable PCI bus mastership, or the source and destination start addresses are not aligned, then P_ERR will be asserted. Otherwise, the ACT bit will be set, and the DMA will then start transferring data, sharing ownership of the VMEbus with the PCI Target and Interrupt channels and the PCI bus with the VMEbus Slave Channel.

In Step 5, one waits for termination of the DMA transfers. The DMA will continue with the transfers until it:

- completes all transfers,
- is terminated early with the STOP_REQ bit, or
- encounters an error on the PCI bus or VMEbus.

Each of these conditions will cause the ACT bit to clear, and a corresponding status bit to be set in the DGCS register. If enabled in Step 1, an interrupt will also be generated. Once the software has set the GO bit, the software can monitor for DMA completion by either waiting for generation of an interrupt, or by polling the status bits. It is recommended that a background timer also be initiated to time-out the transfer. This will ensure that the DMA has not been hung up by a busy VMEbus, or other such system issues.

If an early termination is desired, perhaps because a higher priority operation is required, the STOP_REQ bit in the DGCS register can be set. This will stop all DMA operations on the source bus immediately, and set the STOP bit in the same register when the last piece of queued data in the DMA FIFO has been written to the destination bus. Attempting to terminate the transfer with the HALT_REQ bit will have no effect in direct mode operation since this bit only requests the DMA to stop between command packets in linked-list mode operation.

When the software has detected completion, it should verify the status bits in the DGCS register to see the reason for completion. If one of the error bits have been set, it proceeds into an error handling routine (see “DMA Error Handling” on page 2-93). If the STOP bit was set, the software should take whatever actions were desired when it set the STOP_REQ bit. For example, if it was stopped for a higher priority transfer, it might record the DLA, DVA and DTBC registers, and then reprogram them with the higher priority transfer. When that has completed, it can restore the DVA, DLA and DTBC registers to complete the remaining transfers.

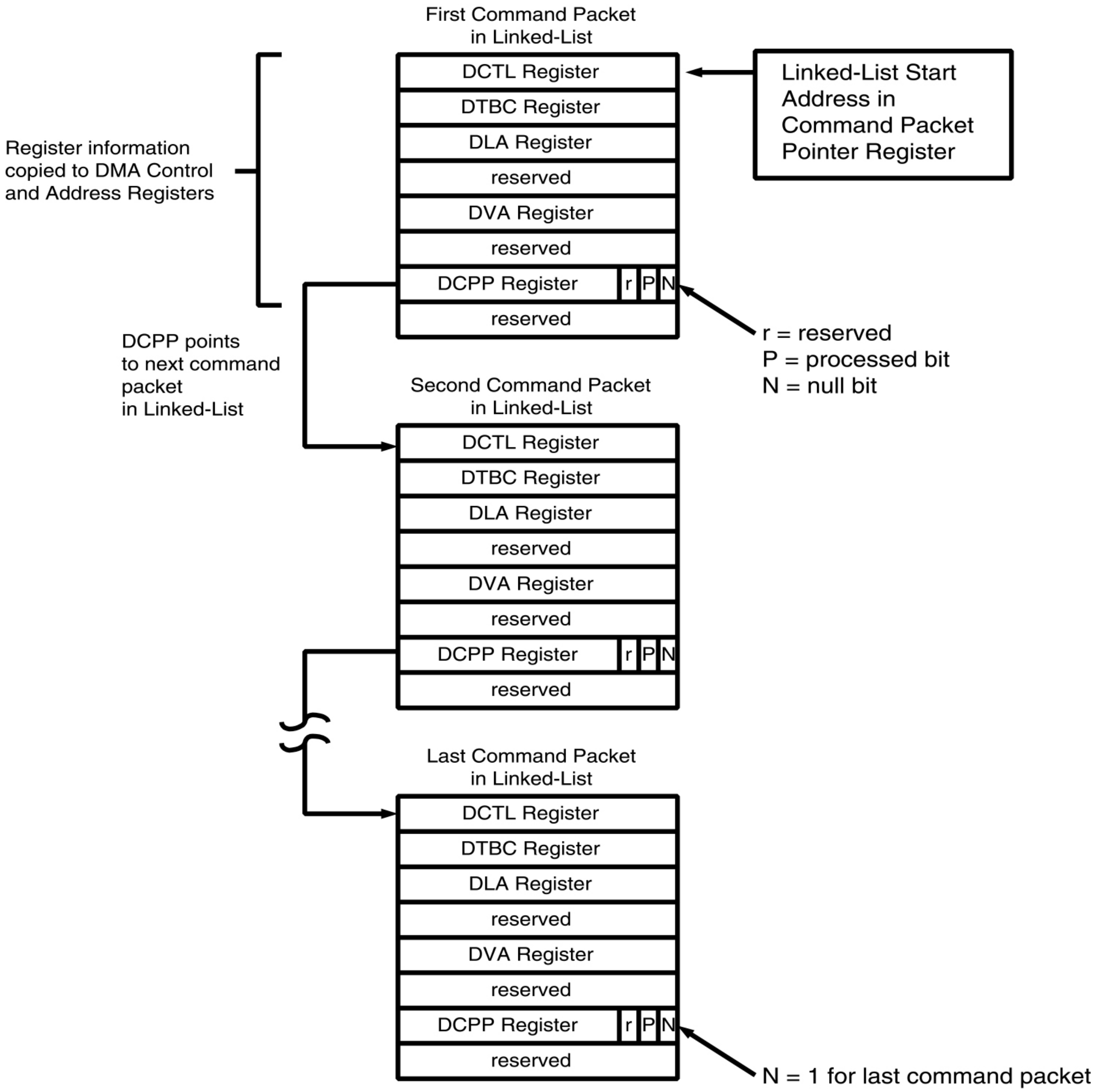
If the DONE bit was set, it indicates that the DMA completed its requested transfer successfully, and if more transfers are required, the software can proceed to Step 2 to start a new transfer.

Linked-List Operation

Unlike direct mode, in which the DMA performs a single block of data at a time, linked-list mode allows the DMA to transfer a series of non-contiguous blocks of data without software intervention. Each entry in the linked-list is described by a command packet which parallels the DMA register layout. The data structure for each command packet is the same (see Figure 2-17), and contains all the necessary information to program the DMA address and control registers. It could be described in software as a record of eight 32-bit data elements. Four of the elements represent the four core registers required to define a DMA transfer: DCTL, DTBC, DVA, and DLA. A fifth element represents the DCPP register which points to the next command packet in the list. The least two significant bits of the DCPP element (the PROCESSED and NULL bits) provide status and control information for linked list processing.

The PROCESSED bit indicates whether a command packet has been PROCESSED or not. When the DMA processes the command packet and has successfully completed all transfers described by this packet, it sets the PROCESSED bit to “1” before reading in the next command packet in the list. This implies that the PROCESSED bit must be initially set for “0” by the user for it to be of use. This bit, when set to 1, indicates that this command packet has been disposed of by the DMA and its memory can be de-allocated or reused for another transfer description.

Figure 2-16: Command Packet Structure and Linked List Operation

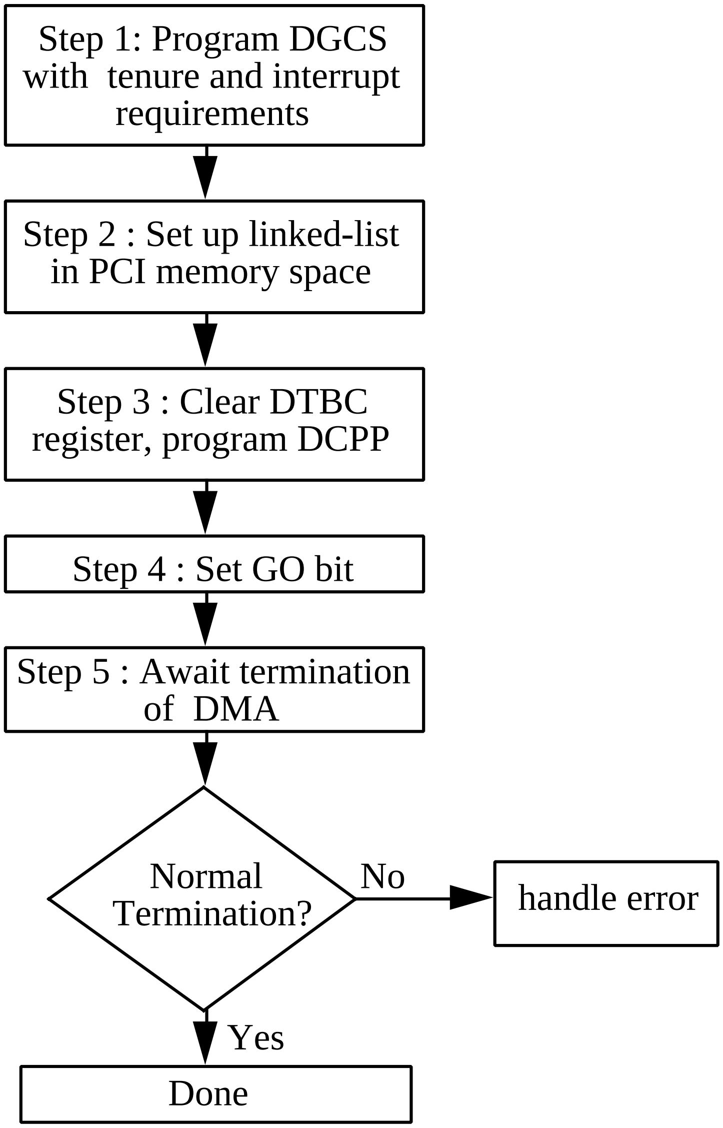


The NULL bit indicates the termination of the entire linked list. If the NULL bit is set to “0”, the DMA processes the next command packet pointed to by the command packet pointer. If the NULL bit is set to “1” then the address in the command packet pointer is considered invalid and the DMA stops at the completion of the transfer described by the current command packet.

Figure 2-17 outlines the steps in programming the DMA for linked-list operation.

In Step 1, the DGCS register is set up: the CHAIN bit is set, VON and VOFF are programmed with the appropriate values for controlling DMA VMEbus tenure, and the interrupt bits (INT_STOP, INT_HALT, INT_DONE, INT_LERR, INT_VERR, and INT_P_ERR) are programmed to enable generation of interrupts based on DMA termination events. DMA interrupt enable bits in the LINT_EN or VINT_EN bits should also be enabled as necessary (“PCI Interrupt Generation” on page 2-61 and “VMEbus Interrupt Generation” on page 2-64).

Figure 2-17: DMA Linked List Operation


Step 1: Program DGCS with tenure and interrupt requirements
Step 2 : Set up linked-list in PCI memory space
Step 3 : Clear DTBC register, program DCPP
Step 4 : Set GO bit
Step 5 : Await termination of DMA
Normal Termination?
No
handle error
Yes
Done

In Step 2, the linked-list structure is programmed with the required transfers. The actual structure may be set up at any time with command packet pointers pre-programmed and then only the remaining DMA transfer elements need be programmed later. One common way is to set up the command packets as a circular queue: each packet points to the next in the list, and the last points to the first. This allows continuous programming of the packets without having to set-up or tear down packets later.

Once the structure for the linked-list is established, the individual packets are programmed with the appropriate source and destination addresses, transfer sizes and attributes.

In Step 3, Clear the DTBC register and program the DCPP register to point to the first command packet in the list.

Caution

When using the DMA to perform linked-list transfers, it is important to ensure that the DTBC register contains a value of zero before setting the GO bit of the DGCS register. Otherwise, the DMA may not read the first command packet but instead perform a direct mode transfer based on the contents of the DCTL, DTBC, DLA, DVA and DGCS registers. After this direct mode transfer is completed, the PROCESSED bit of the first command packet is programmed with a value of 1 even though the packet was not actually processed. The DMA continues as expected with the next command packet.

In Step 4, to start the linked-list transfer, set the GO bit in the DGCS register. The DMA will first perform the transfers defined by the current contents of the DCTL, DTBC, DVA and DLA registers. Once that is complete it will then start the transfers defined by the linked-list pointed to in the DCPP register.

In Step 5, await and deal with termination of the DMA. Once the DMA channel is enabled, it processes the first command packet as specified by the DCPP register. The DMA transfer registers are programmed by information in the command packets and the DMA transfer steps along each command packet in sequence (see Figure 2-16). The DMA will terminate when it:

- processes a command packet with the NULL bit set indicating the last packet of the list,
- is stopped with the STOP_REQ bit in the DGCS register,
- is halted with the HALT_REQ bit in the DGCS register, or
- encounters an error on either the PCI bus or VMEbus.

Each of these conditions will cause the ACT bit to clear, and a corresponding status bit to be set in the DGCS register. If enabled in step 1, an interrupt will also be generated. Once the software has set the GO bit, the software can monitor for DMA completion by either waiting for generation of an interrupt, by polling the status bits in the DGCS register, or by polling the PROCESSED bits of the command packets. It is recommended that a background timer also be initiated to time-out the transfer. This will ensure that the DMA has not been hung up by a busy VMEbus, or other such system issues.

Linked-list operation can be halted by setting the HALT_REQ bit in the DGCS register (Table B-108). When the HALT_REQ bit is set, the DMA terminates when all transfers defined by the current command packet is complete. It then loads the next command packet into its registers. The HALT bit in the DGCS register is asserted, and the ACT bit in the DGCS register is cleared. The PROCESSED bit in the linked-list is set to “1” approximately 1 ms after the HALT bit is set: therefore after a DMA halt the user should wait at least 1 ms before checking the status of the PROCESSED bit.

The DMA can be restarted by clearing the HALT status bit and setting the GO bit if desired during the same register write. If the DMA is restarted, the ACT bit is set by the Universe II and execution continues as if no HALT had occurred: for example, the Universe II processes the current command packet (see Figure 2-17 on page 2-85).

In contrast to a halt, the DMA can also be immediately terminated through the STOP_REQ bit. This will stop all DMA operations on the source bus immediately, and set the STOP bit in the same register when the last piece of queued data in the DMA FIFO has been written to the destination bus.

Once stopped, the DVA, DLA and DTBC registers contain values indicating the next addresses to read/write and the number of bytes remaining in the transfer. Clearing the STOP bit and setting the GO bit will cause the DMA to start up again from where it left off, including continuing with subsequent command packets in the list.

If the DMA is being stopped to insert a high priority DMA transfer, the remaining portion of the DMA transfer may be stored as a new command packet inserted at the top of the linked list. A new command packet with the attributes of the high priority transfer is then placed before that one in the list. Now the linked list is set up with the high priority packet first, followed be the remainder of the interrupted packet, followed in turn by the rest of the linked list. Finally, the DTBC register is cleared and the DCPP programmed with a pointer to the top of the list where the high priority command packet has been placed. When the GO bit is set (after clearing the STOP status bit in the DGCS register), the DMA will perform the transfers in the order set in the linked list. For more details on updating the linked list see “Linked List Updating” below.

DMA transfers continue until the DMA encounters a command packet with the NULL bit set to “1”, indicating that the last packet has been reached. At this point, the DMA stops, the DONE bit is set, and the ACT flag is cleared. As it completes the transfers indicated by each command packet, the DMA sets the PROCESSED bit in that command packet before reading in the next command packet and processing its contents.

Linked List Updating

The Universe II provides a mechanism which allows the linked list to be updated with additional linked list entries without halting or stopping the DMA. This takes place through the use of a semaphore in the device: the UPDATE bit in the D_LLUE register (Table B-110). This bit is meant to ensure that the DMA does not read a command packet into the DMA registers while the command packet (outside the Universe II) is being updated. This semaphore does not prevent external masters from updating the DMA registers.

Adding to a linked list begins by writing a “1” to the UPDATE bit. The DMA checks this bit before proceeding to the next command packet. If the UPDATE bit is “0”, then the DMA locks the UPDATE bit against writes and proceeds to the next command packet. If the UPDATE bit is “1”, then the DMA waits until the bit is cleared before proceeding to the next command packet.

Therefore, setting the UPDATE bit is a means of stalling the DMA at command packet boundaries while local logic updates the linked list.

In order to ensure that the DMA is not currently reading a command packet during updates, the update logic must write a “1” to the UPDATE bit and read a value back. If a “0” is read back from the UPDATE bit, then the DMA is currently reading a command packet and has locked the UPDATE bit against writes. If a “1” is read back from the UPDATE bit, then the DMA is idle or processing a transaction and command packets can be updated. If the DMA attempts to proceed to the next command packet during the update, it will encounter the set UPDATE bit and wait until the bit is cleared.

If a set of linked command packets has already been created with empty packets at the end of new transfers, adding to the end of the current linked list takes the following procedure:

1. Get UPDATE valid (write “1”, read back “1”),
2. Program attributes for new transfer in next available packet in list.
3. Change “null” pointer (on previous tail of linked list),
4. Release update (clear the UPDATE bit).

After updating the linked list, the DMA controller will be in three possible conditions:

1. It may be active and working its way through the linked list. In this case, no further steps are required.
2. The DMA may be idle (done) because it reached the final command packet. If a full set of linked command packets had already been created ahead of time, then the DCPP register would point to the most recently programmed command packet, and the DTBC register would be zero. The DMA can be started on the new packet by simply clearing the DONE bit and setting the GO bit in the DGCS register. If a set of command packets had not been created ahead of time, the DCPP register may not be programmed to any valid packet, and will need programming to the newly programmed packet.
3. The DMA has encountered an error. In this circumstance, see DMA Error Handling on page 141 for how to handle DMA errors.

Operation may be considerably simplified by ensuring that sufficient command packets have been created during system initialization, probably in a circular queue. In this fashion, when a new entry is added to the list, it is simply a matter of programming the next available entry in the list with the new transfer attributes and changing the previously last packet's NULL bit to zero. The DCPP register will be guaranteed to point to a valid command packet, so upon updating the list, both cases 1 and 2 above can be covered by clearing the DONE bit and setting the GO bit. This will have no effect for case 1 since the DMA is still active, and will restart the DMA for case 2.

If an error has been encountered by the DMA (case 3), setting the GO bit and clearing the DONE bit will not be sufficient to restart the DMA—the error bits in the DGCS register will also have to be cleared before operation can continue.

FIFO Operation and Bus Ownership

The DMA uses a 256-byte FIFO. (The DMA FIFO is 64 bits wide). This supports high performance DMA transfers. In general, the DMA reads data from the source, and stores it as transactions in the FIFO. On the destination side, the DMA requests ownership of the master and once granted begins transfers. Transfers stop on the source side when the FIFO fills, and on the destination side when the FIFO empties.

PCI-to-VMEbus Transfers

PCI-to-VMEbus transfers involve the Universe II reading from the PCI bus and writing to the VMEbus.

The PCI bus is requested for the current read once 128 bytes are available in the DMAFIFO. The DMA Channel fills the DMAFIFO using PCI read transactions with each transaction broken at address boundaries determined by the programmed PCI aligned burst size (PABS field in the MAST_CTL register, Table B-160). This ensures that the DMA makes optimal use of the PCI bus by always generating bursts of 32, 64 or 128 bytes with zero wait states.

The DMA packs read data into the DMAFIFO to the full 64-bit width of the FIFO, independent of the width of the PCI bus, or the data width of the ensuing VMEbus transaction. The PCI read transactions continue until either the DMA has completed the full programmed transfer, or there is insufficient room available in the DMAFIFO for a full transaction. The available space required for another burst read transaction is again 128 bytes. Since the VMEbus is typically much slower than the PCI bus, the DMAFIFO may fill frequently during PCI-to-VMEbus transfers, though the depth of the FIFO helps to minimize this. When the DMAFIFO fills, the PCI bus is free for other transactions (for example, between other devices on the bus or possibly for use by the Universe II’s VMEbus Slave Channel). The DMA only resumes read transactions on the PCI bus when the DMAFIFO has space for another aligned burst size transaction.

Caution

The DMA may prefetch extra read data from the external PCI target. This implies that the DMA should only be used with memory on the PCI bus which has no adverse side-effects when prefetched. The Universe II will prefetch up to the aligned address boundary defined in the PABS field of the MASC_CTL register. On the VMEbus, the actual programmed number of bytes in the DTBC register will be written. Prefetching can be avoided by programming the DMA for transfers that terminate at the PABS boundary. If further data is required beyond the boundary, but before the next boundary, the DTBC register may be programmed to eight byte transfers. The DMA will fetch the full eight bytes, and nothing more. Programming the DTBC to less than eight bytes will still result in eight bytes fetched from PCI.

The DMA requests ownership of the Universe II's VMEbus Master Interface once 64 bytes of data have been queued in the DMAFIFO (see “VMEbus Requester” on page 2-10 on how the VMEbus Master Interface is shared between the DMA, the PCI Target Channel, and the Interrupt Channel). The Universe II maintains ownership of the Master Interface until:

- the DMAFIFO is empty,
- the DMA block is complete,
- the DMA is stopped,
- a linked list is halted,
- the DMA encounters an error, or
- the DMA VMEbus tenure limit (VON in the DGCS register).

The DMA can be programmed to limit its VMEbus tenure to fixed block sizes using the VON field in the DGCS register (Table B-108). With VON enabled, the DMA will relinquish ownership of the Master Interface at defined address boundaries. See DMA VMEbus Ownership on page 2-79.

To further control the DMA's VMEbus ownership, the VOFF timer in the DGCS register can be used to program the DMA to remain off the VMEbus for a specified period when VMEbus tenure is relinquished. See DMA VMEbus Ownership on page 2-79.

The DMA Channel unpacks the 64-bit data queued in the DMAFIFO to whatever the programmed transfer width is on the VMEbus (e.g. D16, D32, or D64). The VMEbus Master Interface delivers the data in the DMAFIFO according to the VMEbus cycle type programmed into the DCTL register (Table B-98, see “DMA Controller” on page 2-75). The DMA provides data to the VMEbus until:

- the DMAFIFO empties, or
- the DMA VMEbus Tenure Byte Count (VON in the DMA_GCSR register, Table B-108) expires.

If the DMAFIFO empties, transfers on the VMEbus stop and, if the cycle being generated is a block transfer, then the block is terminated (AS* negated) and VMEbus ownership is relinquished by the DMA. The DMA does not re-request VMEbus ownership until another eight entries are queued in the DMAFIFO, or the DMA Channel has completed the current Transfer Block on the PCI bus (see VMEbus Release on page 2-11).

PCI bus transactions are the full width of the PCI data bus with appropriate byte lanes enabled. The maximum VMEbus data width is programmable to 8, 16, 32, or 64 bits. Byte transfers can be only of type DO8 (EO). Because the PCI bus has a more flexible byte lane enabling scheme than the VMEbus, the Universe II may be required to generate a variety of VMEbus transaction types to handle the byte resolution of the starting and ending addresses (see Data Transfer on page 2-40).

VMEbus-to-PCI Transfers

VMEbus-to-PCI transfers involve the Universe II reading from the VMEbus and writing to the PCI bus.

With DMA transfers in this direction, the DMA Channel begins to queue data in the DMAFIFO as soon as there is room for 64 bytes in the DMAFIFO. When this watermark is reached, the DMA will request the VMEbus (through the VMEbus Master Interface) and begin reading data from the VMEbus. The Universe II maintains VMEbus ownership until:

- the DMAFIFO is full,
- the DMA block is complete,
- the DMA is stopped,
- a linked list is halted,
- the DMA encounters an error, or
- the VMEbus tenure limit is reached (VON in the DGCS register).

The DMA can be programmed to limit its VMEbus tenure to fixed block sizes using the VON field in the DGCS register (Table B-108). With VON enabled, the DMA will relinquish ownership of the Master Interface at defined address boundaries. See “DMA VMEbus Ownership” on page 2-79.

To further control the DMA’s VMEbus ownership, the VOFF timer in the DGCS register can be used to program the DMA to remain off the VMEbus for a specified period when VMEbus tenure is relinquished. See “DMA VMEbus Ownership” on page 2-79.

Entries in the DMAFIFO are delivered to the PCI bus as PCI write transactions as soon as there are 128 bytes available in the DMAFIFO. If the PCI bus responds too slowly, the DMAFIFO runs the risk of filling before write transactions can begin at the PCI Master Interface. Once the DMAFIFO reaches a “nearly full” state (corresponding to three entries remaining) the DMA requests that the VMEbus Master Interface complete its pending operations and stop. The pending read operations typically fill the DMAFIFO. Once the pending VMEbus reads are completed (or the VON timer expires), the DMA relinquishes VMEbus ownership and only re-requests the VMEbus Master Interface once 64 bytes again become available in the DMAFIFO. If the bus was released due to encountering a VON boundary, the bus is not re-requested until the VOFF timer expires.

PCI bus transactions are the full width of the PCI data bus with appropriate byte lanes enabled. The maximum VMEbus data width is programmable to 8, 16, 32, or 64 bits. Byte transfers can be only of type DO8 (EO). Because the PCI bus has a more flexible byte lane enabling scheme than the VMEbus, the Universe II may be required to generate a variety of VMEbus transaction types to handle the byte resolution of the starting and ending addresses (see “Universe II as PCI Target” on page 2-39).

DMA Interrupts

The Interrupt Channel in the Universe II handles a single interrupt sourced from the DMA Channel which it routes to either the VMEbus or PCI bus via the DMA bits in the LINT_EN and VINT_EN registers. There are six internal DMA sources of interrupts and these are all routed to this single interrupt. Each of these six sources may be individually enabled, and are listed in Table 2-18 below. Setting the enable bit enables the corresponding interrupt source.

Table 2-18: DMA Interrupt Sources and Enable Bits

Interrupt Source	Enable Bit
Stop Request	INT_STOP
Halt Request	INT_HALT
DMA Completion	INT_DONE
PCI Target-Abort or Master-Abort	INT_LERR
VMEbus Error	INT_VERR
Protocol Error	INT_M_ERR

Once an enabled DMA interrupt has occurred, regardless of whether the LINT_EN or VINT_EN enable bits have been set, the corresponding DMA bit in the LINT_STAT (Table B-114) and VINT_STAT (Table B-122) registers are set. Each one must be cleared independently. Clearing one does not clear the other. See “Interrupt Handling” on page 2-67.

Interactions with Other Channels

This section describes the impact that the PCI Bus Target Channel and the VMEbus Slave Channel may have on the DMA Channel.

The Universe II does not apply PCI 2.1 transaction ordering requirements to the DMA Controller. That is, reads and writes through the DMA Controller can occur independently of the other channels.

ADOH cycles and RMW cycles through the VMEbus Slave Channel do impact on the DMA Channel. Once an external VMEbus master locks the PCI bus, the DMA Controller will not perform transfers on the PCI bus until the Universe II is unlocked (see “VMEbus Lock Commands (ADOH Cycles)” on page 2-21). When an external VMEbus Master begins a RMW cycle, at some point a read cycle will appear on the PCI bus. During the time between when the read cycle occurs on the PCI bus and when the associated write cycle occurs on the PCI bus, no DMA transfers will occur on the PCI bus (see “VMEbus Read-Modify-Write Cycles (RMW Cycles)” on page 2-22).

If the PCI Target Channel locks the VMEbus using VOWN, no DMA transfers will take place on the VMEbus (see “Using the VOWN bit” on page 2-47).

DMA Error Handling

This section describes how the Universe II responds to errors involving the DMA, and how the user can recover from them. As described below, the software source of a DMA error is a protocol, and the hardware source of a DMA error is a VMEbus error, or PCI bus Target-Abort or Master-Abort.

DMA Software Response to Error

While the DMA is operating normally, the ACT bit in the DGCS register will be set (Table B-108). Once the DMA has terminated, it will clear this bit, and set one of six status bits in the same register.   The DONE bit will be set if the DMA completed all its programmed operations normally. If the user interrupted the DMA, either the STOP or HALT bits will be set. If an error has occurred, one of the remaining three bits, LERR, VERR, or P_ERR, will be set. All six forms of DMA terminations can be optionally set to generate a DMA interrupt by setting the appropriate enable bit in the DGCS register (see “DMA Interrupts” on page 2-92).

- LERR is set if the DMA encounters an error on the PCI bus: either a Master-Abort or Target-Abort. Bits in the PCI_CSR register will indicate which of these conditions caused the error.
- VERR is set if the DMA encounters a bus error on the VMEbus. This will be exclusively through a detected assertion of BERR* during a DMA cycle.
- P_ERR is set if the GO bit in the DGCS register is set to start the DMA, and the DMA has been improperly programmed either because the BM bit in the PCI_CSR disables PCI bus mastership, or the source and destination start addresses are not aligned (see “Source and Destination Addresses” on page 2-76).

Whether the error occurs on the destination or source bus, the DMA_CTL register contains the attributes relevant to the particular DMA transaction. The DTBC register provides the number of bytes remaining to transfer on the PCI side. The DTBC register contains valid values after an error. The DLA and DVA registers should not be used for error recovery.

DMA Hardware Response to Error

This section describes how transfers proceed following a bus error, and how interrupts can be generated following DMA error conditions.

When the error condition (VMEbus Error, Target-Abort, or Master-Abort) occurs on the source bus while the DMA is reading from the source bus, the DMA stops reading from the source bus. Any data previously queued within the DMAFIFO is written to the destination bus. Once the DMAFIFO empties, the error status bit is set and the DMA generates an interrupt (if enabled by INT_LERR or INT_VERR in the DGCS register—see “DMA Interrupts” on page 2-92).

When the error condition (VMEbus Error, Target-Abort, or Master-Abort) occurs on the destination bus while the DMA is writing data to the destination bus, the DMA stops writing to the destination bus, and it also stops reading from the source bus. The error bit in the DGCS register is set and an interrupt asserted (if enabled).

Interrupt Generation During Bus Errors

To generate an interrupt from a DMA error, there are two bits in the DGCS register (and one bit each in the VINT_EN and LINT_EN registers). In the DGCS register the INT_LERR bit enables the DMA to generate an interrupt to the Interrupt Channel after encountering an error on the PCI bus. The INT_VERR enables the DMA to generate an interrupt to the Interrupt Channel upon encountering an error on the VMEbus. Upon reaching the Interrupt Channel, all DMA interrupts can be routed to either the PCI bus or VMEbus by setting the appropriate bit in the enable registers. All DMA sources of interrupts (Done, Stopped, Halted, VMEbus Error, and PCI Error) constitute a single interrupt into the Interrupt Channel.

Resuming DMA Transfers

When a DMA error occurs (on the source or destination bus), the user should read the status bits and determine the source of the error. If it is possible to resume the transfer, the transfer should be resumed at the address that was in place up to 256 bytes from the current byte count. The original addresses (for example, DLA and DVA) are required in order to resume the transfer at the appropriate location. However, the values in the DLA and the DVA registers should not be used to reprogram the DMA, because they are not valid once the DMA begins. In direct mode, it is the user’s responsibility to record the original state of the DVA and DLA registers for error recovery. In Linked-List mode, the user can refer to the current Command Packet stored on the PCI bus (whose location is specified by the DCPP register) for the location of the DVA and DLA information.

The DTBC register contains the number of bytes remaining to transfer on the source side. The Universe II does not store a count of bytes to transfer on the destination side. If the error occurred on the source side, then the location of the error is simply the latest source address plus the byte count. If the error occurred on the destination side, then one cannot infer specifically where the error occurred, because the byte count only refers to the number of data queued from the source, not what has been written to the destination. In this case, the error will have occurred up to 256 bytes before: the original address plus the byte count.

Given this background, the following procedure may be implemented to recover from errors.

1. Read the value contained in the DTBC register.
2. Read the record of the DVA and DLA that is stored on the PCI bus or elsewhere (not the value stored in the Universe II registers of the same name, see above).
3. If the difference between the value contained in the DTBC register and the original value is less than 256 bytes (the FIFO depth of the Universe II), reprogram all the DMA registers with their original values.
4. If the difference between the value contained in the DTBC register and the original value is greater than 256 bytes (the FIFO depth of the Universe II), add 256 (the FIFO depth of the Universe II) to the value contained in the DTBC register.

5. Add the difference between the original value in the DTBC and the new value in the DTBC register to the original value in the DLA register.
6. Add the difference between the original value in the DTBC and the new value in the DTBC register to the original value in the DVA register.
7. Clear the status flags.
8. Restart the DMA (see DMA Initiation on page 2-78).

Registers

The VMEbus Interface has two primary groups of registers:

- System Registers below
- Universe II Registers on page 2-97

The System Registers (which include the PCI-to-VME Command Register, VME BERR Address, and VME BERR Address Modifier Register) are located in memory. The user will need to access these registers to enable access to the VMEbus and make use of the various auxiliary functions.

Note

In order to enable access to the VMEbus both bits 10 and 11 must be set to high in the System COMM Register located at $D800E in memory.

The Universe II registers are located internally in the Universe II interface chip. These registers are initialized by the PCI BIOS during boot time. The registers can be accessed by PCI BIOS calls or VMIC control function library software. The user will need to access these registers primarily to obtain the base addresses for the various user programmable registers. A complete description of the PCI configuration space registers is included in Appendix A in Figure A-1 and in Figure A-2.

Table 2-19 on page 144 shows the base address mapping for each of the register sets. These base addresses are mapped by the PCI BIOS at boot time.

Table 2-19: Interface Base Address Map

Register Group	Base Address Name	Size (Bytes)	Type	Memory Space Location
PCI-to-VME System Registers	-	10 Bytes	Memory	$D800E
Universe II Registers (UCSR)	UNIV_BASE	4K	Memory	$10 (Universe II)

The Universe II chip maps the 4K register set in both I/O and memory space each with 4K byte resolution. The registers may thus be accessed using either mode. However, in order to maintain compatibility with existing software, memory space accesses should be used. The address of the registers are contained in the Universe II's configuration space base address registers 0 and 1 (for example, config space offset 0x10 and 0x14).

System Registers

The system registers consist of four registers located at Memory Address $D800E. These registers provide additional control/status not contained within the Universe II chip, such as master/slave endian conversion and non-slot 1 bus time-out timer control. A complete description of all System Registers is included in Appendix A, Table A-1, the System Register Map.

Universe II Registers

Most of the VMEbus Interface's programmable registers are located within the Universe II chip. The Universe II chip contains a 4K register set located at base address UNIV_BASE. The VMEbus Interface contains a configurable jumper which allows the Universe II based registers to be located in memory space or I/O space.

A list of Universe II registers is included in Appendix B.

Universe II Registers

This section is organized as follows:

- Overview of Universe II Registers below,
- Register Access from the PCI Bus on page 2-99,
- Register Access from the VMEbus on page 2-101,
- Mailbox Registers on page 2-105, and
- Semaphores on page 2-105.

Overview of Universe II Registers

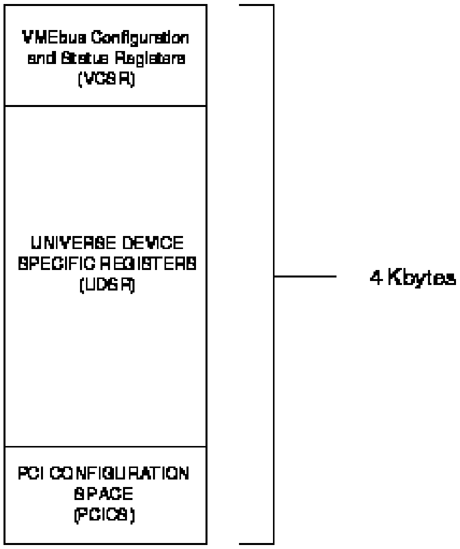
The Universe II Control and Status Registers (UCSR) occupy 4 Kbytes of internal memory. This 4 Kbytes is logically divided into three groups (see Figure 2-18 below):

- PCI Configuration Space (PCICS),
- Universe II Device Specific Registers (UDSR), and
- VMEbus Control and Status Registers (VCSR).

The Universe II registers are little-endian.

The access mechanisms for the UCSR are different depending upon whether the register space is accessed from the PCI bus or VMEbus. Register access from the PCI bus and VMEbus is discussed below.

Figure 2-18: Universe II Control and Status Register Space


VMEbus Configuration and Status Registers (VCSR)
UNIVERSE DEVICE SPECIFIC REGISTERS (UDSR)
PCI CONFIGURATION SPACE (PCICS)
4 Kbytes

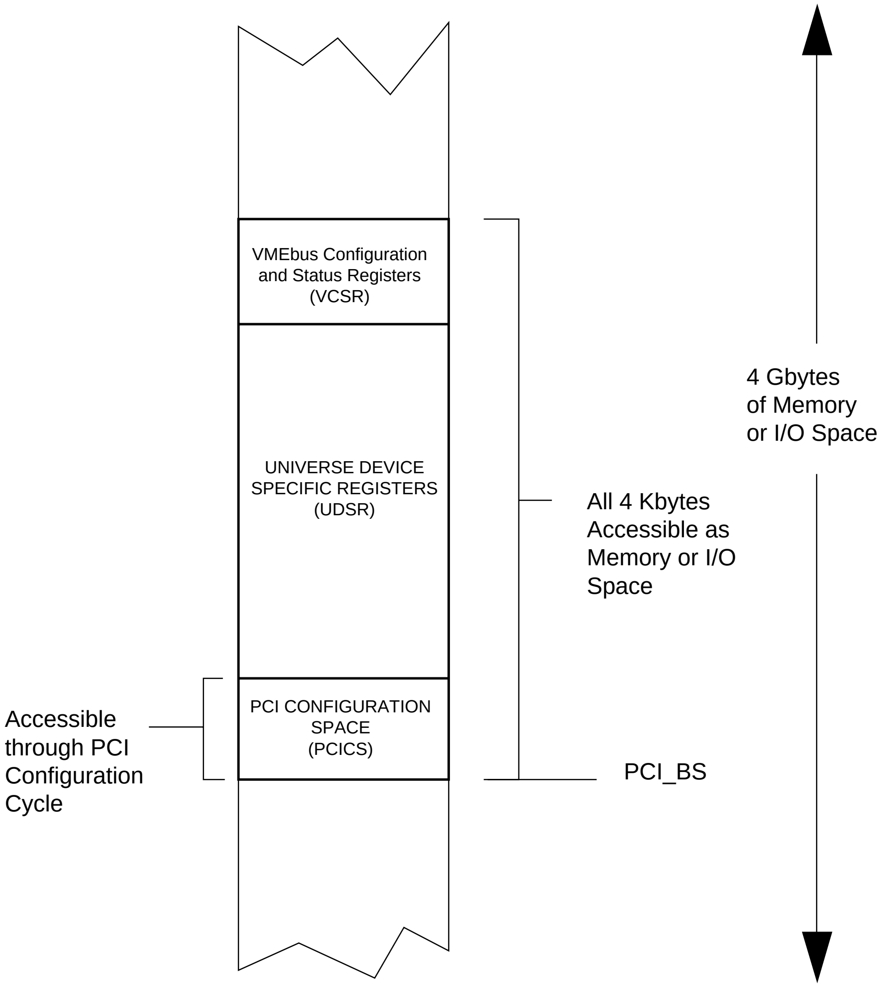
Register Access from the PCI Bus

There are two mechanisms to access the UCSR space from the PCI bus: through Configuration space or through PCI Memory or I/O space (Table B-10).

PCI Configuration Access

When the UCSR space is accessed as Configuration space, it means that the access is externally decoded and the Universe II is notified via IDSEL (much like a standard chip select signal). Since the register location is encoded by a 6-bit register number (a value used to index a 32-bit chunk of Configuration space), only the lower 256 bytes of the UCSR can be accessed as Configuration space (this corresponds to the PCICS in the UCSR space, see Figure 2-20). Thus, only the PCI configuration registers are accessible through PCI Configuration cycles.

Figure 2-19: PCI Bus Access to UCSR as Memory or I/O Space


VMEbus Configuration and Status Registers (VCSR)
UNIVERSE DEVICE SPECIFIC REGISTERS (UDSR)
PCI CONFIGURATION SPACE (PCICS)
4 Gbytes of Memory or I/O Space
All 4 Kbytes Accessible as Memory or I/O Space
Accessible through PCI Configuration Cycle
PCI_BS

Memory or I/O Access

Exactly two 4-Kbyte ranges of addresses in PCI Memory space and/or PCI I/O space can be dedicated to the Universe II registers. The Universe II has two programmable registers (PCI_BS0 and PCI_BS1) that each specify the base address and address space for PCI access to the Universe II's registers. The PCI_BSx registers can be programmed through PCI Configuration space or through a VMEbus access, to make the Universe II registers available anywhere in the 32-bit Memory space and in I/O space (as offsets of the BS[31:12] field in PCIBSx).

The Universe II chip maps the 4K register set in both I/O and memory space each with 4K byte resolution. The registers may thus be accessed using either mode. However, in order to maintain compatibility with existing software, memory space accesses should be used. The address of the registers are contained in the Universe II's configuration space base address registers 0 and 1 (for example, config space offset 0x10 and 0x14).

The SPACE bit of the PCI_BSx registers specifies whether the address lies in Memory space or I/O space. The SPACE bit of these two registers are read-only. There is a power-up option that determines the value of the SPACE bit of the PCI_BSx registers. At power-up the SPACE bit of the PCI_BS1 register is the negation of the SPACE bit of the PCI_BS0 register.

- When the VA[1] pin is sampled low at power-up, the PCI_BS0 register's SPACE bit is set to "1", which signifies I/O space, and the PCI_BS1 register's SPACE bit is set to "0", which signifies Memory space.
- When VA[1] is sampled high at power-up, the PCI_BS0 register's SPACE register's bit is set to "0", which signifies Memory space, and the PCI_BS1 register's SPACE bit is set to "1", which signifies I/O space.

Universe II registers are not prefetchable. The Universe II does not accept burst writes to its registers.

Eliciting Conditions of Target-Retry

Attempts to access UCSR space from the PCI bus will be retried by the Universe II under the following conditions:

- While UCSR space is being accessed by a VMEbus master, PCI masters will be retried.
- If a VMEbus master is performing a RMW access to the UCSRs then PCI attempts to access the USCR space will result in a Target-Retry until AS* is negated.
- If the Universe II registers are accessed through an ADOH cycle from the VMEbus, any PCI attempt to access the UCSRs will be retried until BBSY* is negated.

Locking the Register Block from the PCI bus

The Universe II registers can be locked by a PCI master by using a PCI locked transaction. When an external PCI master locks the register block of the Universe II, an access to the register block from the VMEbus will not terminate with the assertion of DTACK* until the register block is unlocked. Hence a prolonged lock of the register block by a PCI resource may cause the VMEbus to timeout with a BERR*.

Register Access from the VMEbus

There are two mechanisms to access the UCSR space from the VMEbus. One method uses a VMEbus Register Access Image (VRAI) which allows the user to put the UCSR in an A16, A24 or A32 address space. The VRAI approach is useful in systems not implementing CR/CSR space as defined in the VME64 specification. The other way to access the UCSR is as CR/CSR space, where each slot in the VMEbus system is assigned 512 Kbytes of CR/CSR space.

Each method is discussed below.

VMEbus Register Access Image (VRAI)

The VMEbus register access image is defined by the following register fields:

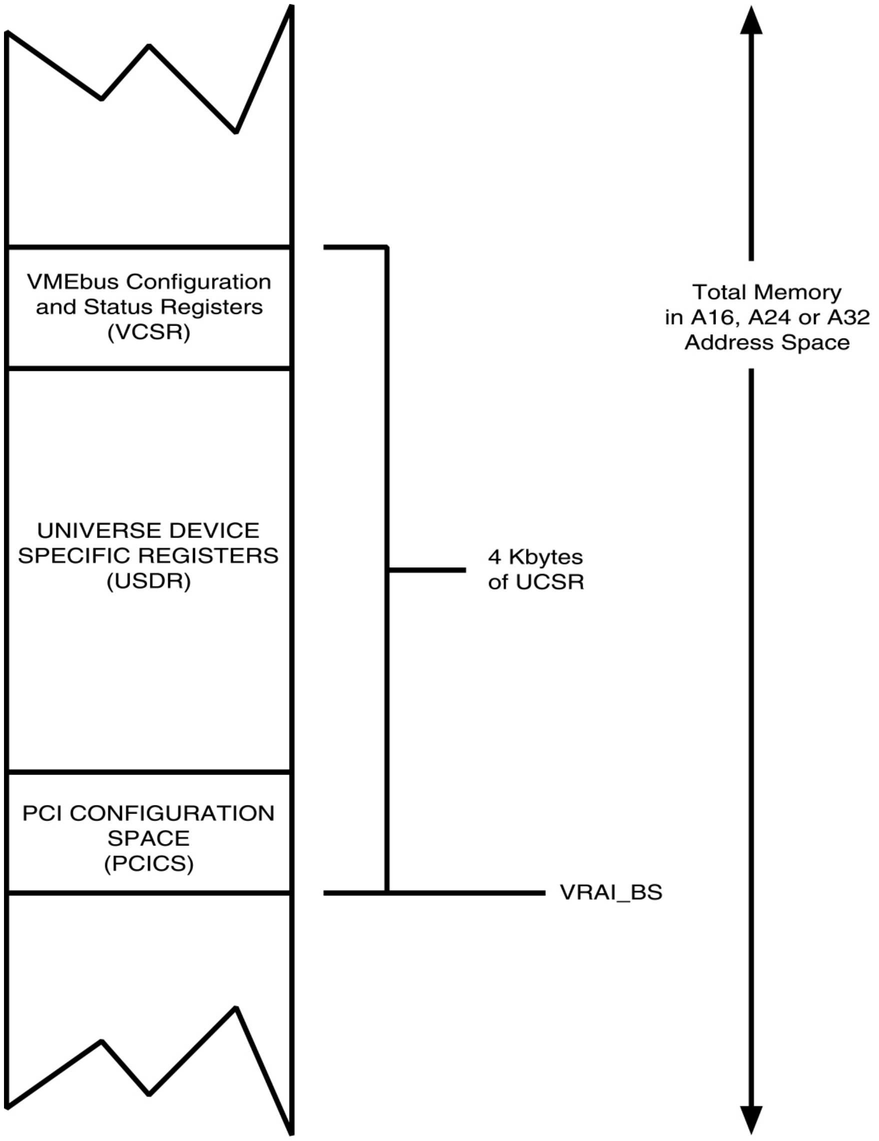
Table 2-20: Programming the VMEbus Register Access Image

Field	Register Bits	Description
address space	VAS in Table B-204	one of A16, A24, A32
base address	BS[31:12] in Table B-206	lowest address in the 4Kbyte slave image
slave image enable	EN in Table B-204	enables VMEbus register access image
mode	SUPER in Table B-204	Supervisor and/or Non-Privileged
type	PGM in Table B-204	Program and/or Data

The VMEbus Register Access Image occupies 4 Kbytes in A16, A24 or A32 space (depending upon the programming of the address space described in Table 2-20 above, see Figure 2-20). All registers are accessed as address offsets from the VRAI base address programmed in the VRAI_BS register (Table B-207). The image can be enabled or disabled using the EN bit in the VRAI_CTL register (Table B-204).

Note that the VRAI base address can be configured as a power-up option (see “Power-up Option Descriptions” on page 2-114).

Figure 2-20: UCSR Access from the VMEbus Register Access Image


VMEbus Configuration and Status Registers (VCSR)
UNIVERSE DEVICE SPECIFIC REGISTERS (USDR)
PCI CONFIGURATION SPACE (PCICS)
4 Kbytes of UCSR
VRAI_BS
Total Memory in A16, A24 or A32 Address Space

CR/CSR Accesses

The PCI-to-VMEbus interface does not support CR/CSR space.

The VME64 specification assigns a total of 16 Mbytes of CR/CSR space for the entire VMEbus system. The CR/CSR image is enabled with the EN bit in the VCSR_CTL register (Table B-208). This 16 Mbytes is broken up into 512 Kbytes per slot for a total of 32 slots. The first 512 Kbyte block is reserved for use by the Auto-ID mechanism. The UCSR space occupies the upper 4 Kbytes of the 512 Kbytes available for its slot position (see Figure 2-21 below). The base address of the CR/CSR space allocated to the Universe II's slot is programmed in the VCSR_BS register (Table B-252). For CSRs not supported in the Universe II and for CR accesses, the LAS field in the VCSR_CTL register specifies the PCI bus command that is generated when the cycle is mapped to the PCI bus. There is also a translation offset added to the 24-bit VMEbus address to produce a 32-bit PCI bus address (programmed in the VCSR_TO register, Table B-210).

Note that the registers in the UCSR space are located as address offsets from VCSR_BS. These offsets are different from those used in the VRAI mechanisms, where the first register in the UCSR has address offset of zero (see Table B-1). When accessing the UCSR in CR/CSR space, the first register will have an address offset of 508 Kbytes (512 Kbytes minus 4 Kbytes). A simple approach for determining the register offset when accessing the UCSR in CR/CSR space is to add 508 Kbytes (0x7F000) to the address offsets given in Table B-1.

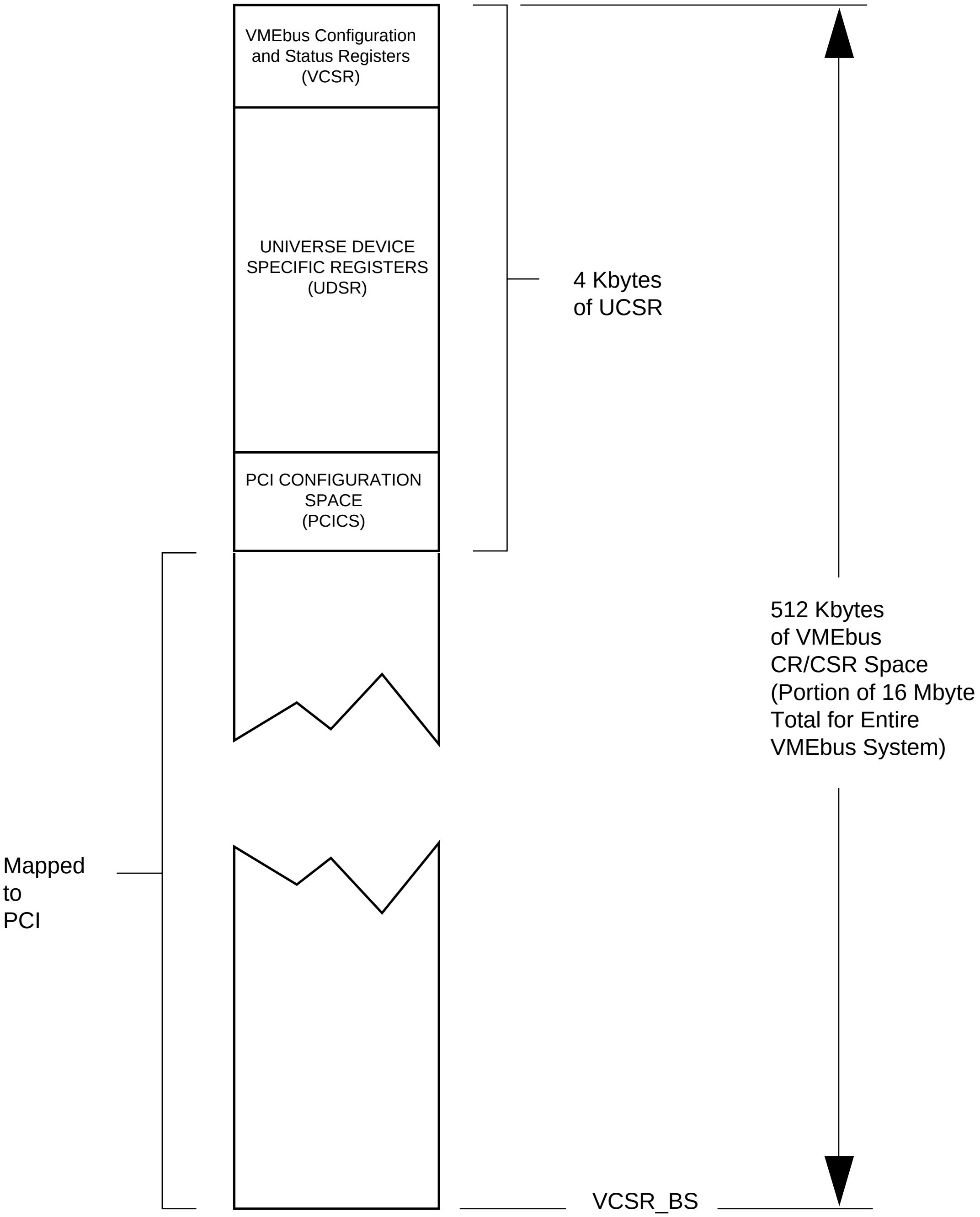
RMW and ADOH Register Access Cycles

The Universe II supports RMW and ADOH accesses to its registers.

A read-modify-write (RMW) cycle allows a VMEbus master to read from a VMEbus slave and then write to the same resource without relinquishing VMEbus tenure between the two operations. The Universe II accepts RMW cycles to any of its registers. This prevents an external PCI Master from accessing the registers of the Universe II until VMEbus AS* is asserted. This is useful if a single RMW access to the ADHO is required.

If a sequence of accesses to the Universe registers must be performed without intervening PCI access to UCSR is required, then the VMEbus master should lock the Universe II through the use of ADOH. This prevents an external PCI Master from accessing the registers of the Universe II until VMEbus BBSY* is negated. It also prevents other VMEbus masters from accessing the Universe II registers.

Figure 2-21: UCSR Access in VMEbus CR/CSR Space



Mailbox Registers

The Universe II has four 32-bit mailbox registers which provide an additional communication path between the VMEbus and the PCI bus (see Table B-148 to Table B-154). The mailboxes support read and write accesses from either bus, and may be enabled to generate interrupts on either bus when they are written to. The mailboxes are accessible from the same address spaces and in the same manner as the other Universe II registers, as described above.

Mailbox registers are useful for the communication of concise command, status, and parameter data. The specific uses of mailboxes depend on the application. For example, they can be used when a master on one bus needs to pass information (a message) on the other bus, without knowing where the information should be stored in the other bus's address space. Or they can be used to store the address of a longer message written by the processor on one bus to the address space on the other bus, through the Universe II. They can also be used to initiate larger transfers through the FIFO, in a user-defined manner.

Often users will enable and map mailbox interrupts, so that when the processor writes to a mailbox from one bus, the Universe II will interrupt the opposite bus. The interrupt service routine on the opposite bus would then cause a read from this same mailbox.

Reading a mailbox cannot automatically trigger an interrupt. However, a similar effect can be achieved by reading the mailbox and then triggering an interrupt through hardware or software. Or one may use a "polling" approach, where one designates a bit in a mailbox register to indicate whether one has read from the mailbox.

For details on how the mailbox interrupts are enabled and mapped, see "Interrupt Handling" on page 2-67 and Mailbox Register Access Interrupts on page 2-73.

Applications will sometimes designate two mailboxes on one interface as being read/write from the PCI bus, and read-only from the VMEbus, and the two other mailboxes as read/write from the VMEbus and read-only from the PCI bus. This eliminates the need to implement locking. The Universe II provides semaphores which can be also be used to synchronize access to the mailboxes. Semaphores are described in the next section.

Semaphores

The Universe II has two general-purpose semaphore registers each containing four semaphores. The registers are SEMA0 (Table B-156) and SEMA1 (Table B-158). To gain ownership of a semaphore, a process writes a logic one to the semaphore bit and a unique pattern to the associated tag field. If a subsequent read of the tag field returns the same pattern, the process can consider itself the owner of the semaphore. A process writes a value of 0 to the semaphore to release it.

When a semaphore bit is a value of 1, the associated tag field cannot be updated. Only when a semaphore is a value of 0 can the associated tag field be updated.

These semaphores allow the user to share resources in the system. While the Universe II provides the semaphores, it is up to the user to determine access to which part of the system will be controlled by semaphores, and to design the system to enforce these rules.

An example of a use of the semaphore involves gating access to the Special Cycle Generator (page 2-44). It may be necessary to ensure that while one process uses the Special Cycle Generator on an address, no other process accesses this address. Before performing a Special Cycle, a process would be required to obtain the semaphore. This process would hold the semaphore until the Special Cycle completes. A separate process that intends to modify the same address would need to obtain the semaphore before proceeding (it need not verify the state of the SCYC[1:0] bit). This mechanism requires that processes know which addresses might be accessed through the Special Cycle Generator.

Utility Functions

This section discusses miscellaneous utility functions that are provided by the Universe II, including:

Resets on page 2-107,

Power-Up Options on page 2-112,

Hardware Initialization (Normal Operating Mode) on page 2-118,

Test Modes on page 2-118, and

Clocks on page 2-119.

Resets

This section is divided in three sections. The first section lists the pins and registers that are involved in the reset circuitry. The second section presents and explains the reset circuitry diagram. The third section provides important suggestions and warnings about configuring the Universe II reset circuitry.

Overview of Reset Support

The Universe II provides a number of pins and registers for reset support. Pin support is summarized in Table 2-21.

Table 2-21: Hardware Reset Mechanism

Interface and Direction	Pin Name	Long Name	Effects[a]
VMEbus Input	VRSYSRST#	VMEbus Reset Input	Asserts LRST# on the local bus, resets the Universe II, and reconfigures power-up options.
VMEbus Output	VXSYSRST	VMEbus System Reset	Universe II output for SYSRST* (resets the VMEbus)
PCI Input	PWRRST#	Power-up Reset	Resets the Universe II and reconfigures power-up options.
	RST#	PCI Reset Input	Resets the Universe II from the PCI bus.
	VME_RESET#	VMEbus Reset Initiator	Causes Universe II to assert VXSYSRST
PCI Output	LRST#	PCI Bus Reset Output	Resets PCI resources
JTAG Input	TRST#	JTAG Test Reset	Provides asynchronous initialization of the TAP controller in the Universe II.

[a] *A more detailed account of the effects of reset signals is provided in Reset Implementation Cautions on page 2-111.*

The Universe II is only reset through hardware. Software can only make the Universe II assert its reset outputs. In order to reset the Universe II through software, the Universe II reset outputs must be connected to the Universe II reset inputs. For example, the SW_LRST bit in the MISC_CTL register, which asserts the LRST# output, will not reset the Universe II itself unless LRST# is looped back to RST#. As described in Reset Implementation Cautions on page 2-111, there are potential loopback configurations resulting in permanent reset.

Table 2-22: Software Reset Mechanism

Register and Table	Name	Type	Function
MISC_CTL Table B-162	SW_LRST	W	Software PCI Reset 0=No effect, 1=Initiate LRST# A read always returns 0.
	SW_SYSRST	W	Software VMEbus SYSRESET 0=No effect, 1=Initiate SYSRST* A read always returns 0.
VCSR_SET Table B-250	RESET	R/W	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=assert LRST#
	SYSFAIL	R/W	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes:0=no effect, 1=assert VXSYSFAIL
VCSR_CLR Table B-248	RESET	R/W	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=negate LRST#
	SYSFAIL	R/W	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes:0=no effect, 1=negate VXSYSFAIL

More detailed information about the effects of various reset events is provided in the next section.

Universe II Reset Circuitry

Table 2-23 below shows how to reset various aspects of the Universe II. For example, it shows that in order to reset the clock services (SYSCLK, CLK64 enables, and PLL divider), PWRRST# should be asserted. If the table is read from left to right, it indicates the effects of various reset sources. Notice that PWRRST# resets all aspects of Universe II listed in column 1 of Table 2-23. Table 2-23 also indicates the reset effects that are extended in time. For example, VXSYSRST# remains asserted for 256 ms after all initiators are removed—this satisfies VMEbus rule 5.2 (minimum of 200 ms SYSRST*). The external 64 MHz clock controls this assertion time. LRST# is asserted for 5 ms or more from all sources except VRSYSRST#. The same information is presented in pictorial format in Figure 2-22.

Table 2-23: Functions Affected by Reset Initiators

Effect of Reset [a,b]	Reset Source
Clock Services SYSCLK CLK64 enables PLL Divider	PWRRST#
VMEbus Services VMEbus Arbiter VMEbus Timer VCSR Registers	PWRRST#, or VRSYSRST#
General Services Most registers Internal state machines	PWRRST#, RST# or VRSYSRST#
Power-Up and Reset State Machine Power-up the device Reset Registers	PWRRST#, or VRSYSRST#
VMEbus Reset Output VXSYSRST# (asserted for more than 200 ms)	PWRRST#, or VME_RESET#, or SW_SYSRST bit in MISC_CTL register
PCI Bus Reset Output LRST# (asserted for at least 5 ms)	PWRRST#, or SW_LRST bit in MISC_CTL register, or RESET bit in VCSR_SET register[c]

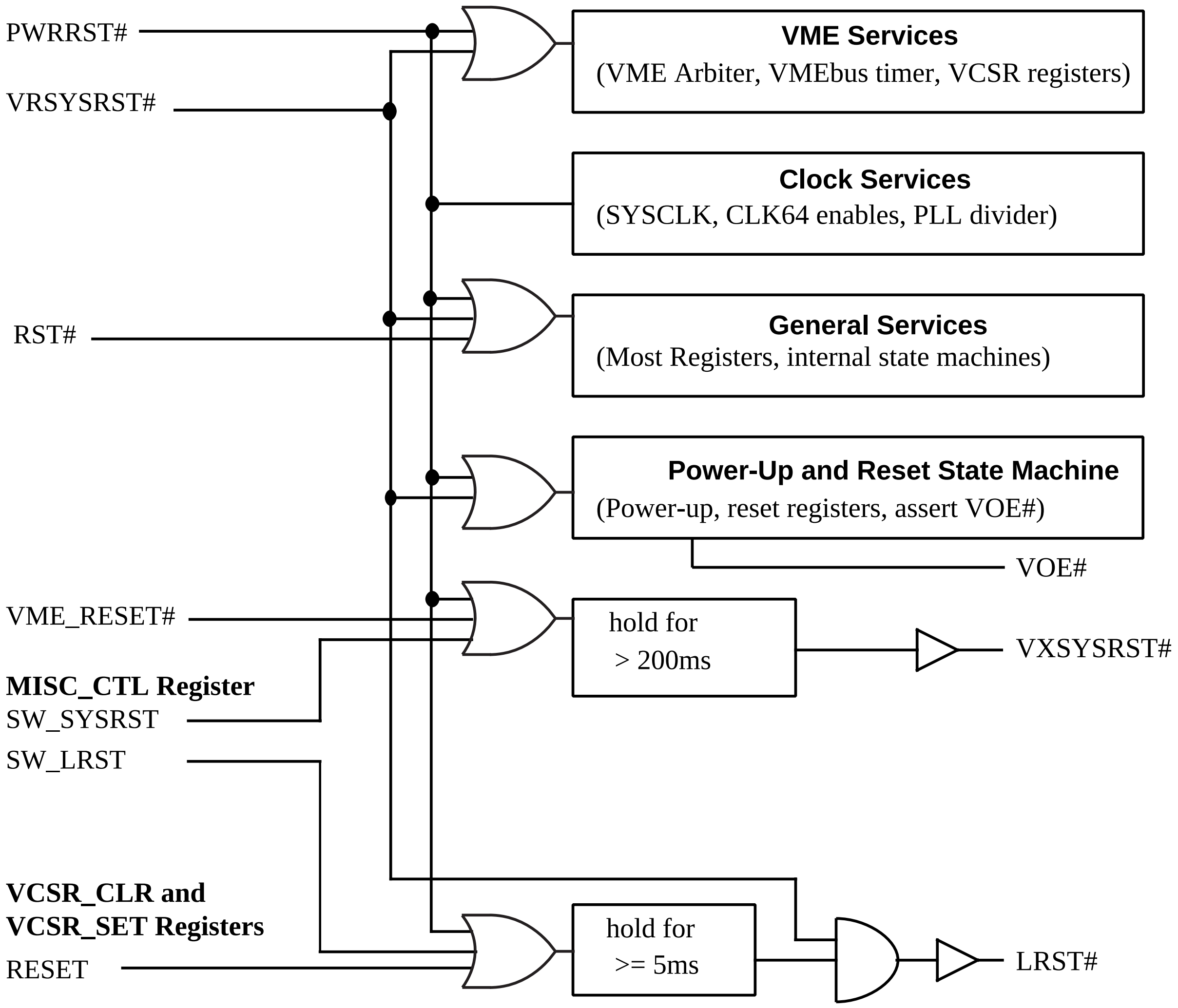
a. On PWRRST#, options are loaded from pins. On SYSRST and RST#, options are loaded from values that were latched at the previous PWRRST#.

b. Refer to Appendix A to find the effects of various reset events.

c. LRST# may be cleared by writing 1 to the RESET bit in the CSR_CLR register.

The interface includes configurable jumpers for enabling/disabling and driving/receiving VMEbus SYSRESET*

Figure 2-22: Reset Circuitry


PWRRST#
VRSYSRST#
RST#
VME_RESET#
MISC_CTL Register
SW_SYSRST
SW_LRST
VCSR_CLR and
VCSR_SET Registers
RESET
VME Services
(VME Arbiter, VMEbus timer, VCSR registers)
Clock Services
(SYSCLK, CLK64 enables, PLL divider)
General Services
(Most Registers, internal state machines)
Power-Up and Reset State Machine
(Power-up, reset registers, assert VOE#)
VOE#
hold for
> 200ms
VXSYSRST#
hold for
>= 5ms
LRST#

Reset Implementation Cautions

To prevent the Universe II from resetting the PCI bus, the LRST# output may be left unconnected. Otherwise, LRST# should be grouped with other PCI reset generators to assert the RST# signal such that:

RST# = LRST# & reset_source1 & reset_source2 &...

If the Universe II is the only initiator of PCI reset, LRST# may be directly connected to RST#.

Assertion of VME_RESET# causes the Universe II to assert VXSYSRST#.

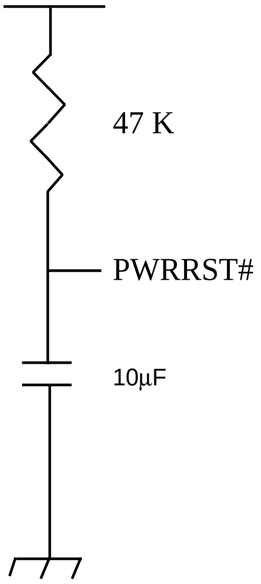
Caution

Since VME_RESET# causes assertion of SYSRST*, and since SYSRST* causes assertion of LRST#, tying both VME_RESET# and LRST# to RST# will put the Universe II into permanent reset. If VME_RESET# is to be driven by PCI reset logic, ensure that the logic is designed to break this feedback path.

The PWRRST# input keeps the Universe II in reset until the power supply has reached a stable level (see Table 2-23 on page 157). It should be held asserted for over 100 milliseconds after power is stable. Typically this can be achieved through a resistor/capacitor combination (see Figure 2-23); however, a more reliable solution using under voltage sensing circuits (e.g. MC34064) is common.

The Universe II supports the VMEbus CSR Bit Clear and Bit Set registers (Table B-248 and Table B-250). The VCSR_SET registers allows the user to assert LRST# or SYSFAIL by writing to the RESET or SYSFAIL bits, respectively. LRST# or SYSFAIL remains asserted until the corresponding bit is cleared in the VCSR_CLR register. The FAIL bit in each of these registers is a status bit and is set by the software to indicate board failure.

Figure 2-23: Resistor-Capacitor Circuit Ensuring Power-Up Reset Duration



Power-Up Options

The Universe II may be automatically configured at power-up to operate in different functional modes. These power-up options allow the Universe II to be set in a particular mode independent of any local intelligence. The Universe II power-up options are listed in Table 2-24 on page 161 and described below.

The majority of the Universe II power-up options (listed below) are loaded from the VMEbus address and data lines after any PWRRST#. There are two power-up options that are not initiated by PWRRST#. The first of these is PCI bus width (a power-up option required by the PCI bus specification), and this is loaded on any RST# event from the REQ64# pin. The second special power-up option is VMEbus SYSCON enabling, required by the VMEbus specification. The SYSCON option is loaded during a SYSRST* event from the BG3IN* signal.

All power-up options are latched from the state of a particular pin or group of pins on the rising edge of PWRST#. Each of these pins except REQ64# has a weak internal pull-down to put the Universe II into a default configuration. (REQ64# has an internal pull-up). If a non-default configuration is required, a pull-up of approximately 10k W (or active drive) is required on the signal. See PCI Bus Width on page 2-116 and the VMEbus Specification concerning Auto-Syscon Detect for the exceptions to the rule described in this paragraph.

The Universe II may be restored to the state it was in immediately following the previous power-up without re-asserting PWRRST#: after SYSRST* or RST# (with PWRRST# negated), the values that were originally latched at the rising edge of PWRRST# will be reloaded into the Universe II (except for PCI bus width and VMEbus SYSCON enabling, which are loaded from their pins).

Table 2-24 on page 161 lists the power-up options of the Universe II, the pins which determine the options, and the register settings that are set by this option. Each option is described in more detail in Power-up Option Descriptions on page 2-114.

Table 2-24: Power-Up Options[a]

Option	Register	Field	Default	Pins
VMEbus Register Access Slave Image	VRAI_CTL	EN	disabled	VA[31]
		VAS	A16	VA[30:29]
	VRAI_BS	BS	0x00	VA[28:21]
VMEbus CR/CSR slave image	VCSR_CTL	LAS[0][b]	memory	VA[20]
	VCSR_TO	TO	0x00	VA[19:15]
Auto-ID	MISC_STAT	DY4AUTO	disabled	VD[30]
	MISC_CTL	V64AUTO	disabled	VD[29]
	VINT_EN	SW_INT	0	
	VINT_STAT	SW_INT	0	
	VINT_MAP1	SW_INT	000	
BI-Mode®	MISC_CTL	BI	disabled	VD[28]
Auto-Syscon Detect	MISC_CTL	SYSCON	enabled	VBGIN[3]*
SYSFAIL* Assertion	VCSR_SET	SYSFAIL	asserted	VD[27]
	VCSR_CLR	SYSFAIL		
PCI Target Image	LSI0_CTL	EN	disabled	VA[13]
		LAS[0]	memory	VA[12]
		VAS	A16	VA[11:10]
	LSI0_BS	BS	0x0	VA[9:6]
	LSI0_BD	BD	0x0	VA[5:2]
PCI Register Access	PCI_BS0, PCI_BS1	SPACE	See Table B-10 on page 227 and Table B-12 on page 228	VA[1]
PCI Bus Size[c]	MISC_STAT	LCLSIZE	32-bit	REQ64#
PCI CSR Image Space	PCI_CSR	BM	disabled	VA[14]

a.All power-up options are latched only at the rising-edge of PWRRST#. They are loaded when PWRRST#, SYSRST, and RST# are negated.*

b.The LAS field will enable the PCI_CSR register's MS or IOS field if the EN FIELD of the LSIO_CTL register is set.

c.The PCI Bus Size is loaded on any RST# event, as per the PCI 2.1 Specification.

Power-up Option Descriptions

This section describes each of the groups of power-up options that were listed in Table 2-24.

VMEbus Register Access Image

The Universe II has several VMEbus slave images, each of which may provide a different mapping of VMEbus cycles to PCI cycles. All VMEbus slave images are configurable through a set of VMEbus slave registers: VSIxCTL, VSIx_BS, VSIx_BA, and VSIx_IO. No VMEbus-to-PCI transaction is possible until these registers are programmed.

The VMEbus Register Access Image (VRAI) power-up option permits access from the VMEbus to the Universe II internal registers at power-up. The power-up option allows programming of the VMEbus register slave image address space and the upper five bits of its base address; all other bits will be zero (see Table 2-25 below). Once access is provided to the registers, then all other Universe II features (such as further VMEbus slave images) can be configured from the VMEbus.

Table 2-25: VRAI Base Address Power-up Options

VRAI_CTL: VAS	BS [31:24]	BS [23:16]	BS [15:12]
A16	0	0	Power-up Option VA [28:25]
A24	0	Power-up Option VA [28:21]	0
A32	Power-up Option VA [28:21]	0	0

Table 2-25 on page 162 above shows how the upper bits in the VRAI base address are programmed for A16, A24, and A32 VMEbus register access images.

VMEbus CR/CSR Slave Image

CR/CSR space is an address space introduced in the VME64 specification. The CR/CSR space on any VMEbus device is 512 Kbytes in size: the upper region of the 512 Kbytes dedicated to register space, and the lower region is dedicated to configuration ROM. The Universe II maps its internal registers to the upper region of the CR/CSR space, and passes all other accesses through to the PCI bus (see “Universe II Registers” on page 2-97).

The VMEbus CR/CSR Slave Image power-up option maps CR/CSR accesses to the PCI bus. CR/CSR space can be mapped to memory or I/O space with a 5-bit offset. This allows mapping to any 128Mbyte page on the PCI bus. As part of this implementation, ensure that the PCI Master Interface is enabled through the MAST_EN bit power-up option (see below) or configured through a register access before accessing configuration ROM.

Auto-ID

There are two Auto-ID mechanisms provided by the Universe II. One is the VME64 specified version which relies upon use of the CR/CSR space for configuration of the VMEbus system, and a Tundra proprietary system which uses the IACK daisy chain for identifying cards in a system. Either of these mechanisms can be enabled at power-up (see “Automatic Slot Identification” on page 2-26).

Because VME64 Auto-ID relies upon SYSFAIL to operate correctly, this power-up option overrides the SYSFAIL power-up option described below.

BI-Mode

BI-Mode (Bus Isolation Mode) is a mechanism for logically isolating the Universe II from the VMEbus for diagnostic, maintenance and failure recovery purposes. BI-Mode may be enabled as a power-up option (see “BI-Mode” on page 2-30). When the Universe II has been powered-up in BI-Mode, then any subsequent SYSRST* or RST# restores the Universe II to BI-Mode.

Auto-Syscon Detect

The VMEbus SYSCON enabling, required by the VMEbus specification, is a special power-up option in that it does not return to its after-power-up state following RST# or SYSRST#. The SYSCON option is loaded during a SYSRST* event from the VBG3IN* signal.

SYSFAIL* Assertion

This power-up option causes the Universe II to assert SYSFAIL* immediately upon entry into reset. The SYSFAIL* pin is released through a register access. Note that this power-up option is over-ridden if VME64 Auto-ID has been enabled. This option would be used when extensive on-board diagnostics need to be performed before release of SYSFAIL*. After completion of diagnostics, SYSFAIL* may be released through software or through initiation of the VME64 Auto-ID sequence if that mechanism is to be used (see “Auto Slot ID: VME64 Specified” on page 2-26).

PCI Target Image

The PCI Target Image power-up option provides for default enabling of a PCI target image (automatically mapping PCI cycles to the VMEbus). The default target image can be mapped with base and bounds at 256Mb resolution in Memory or I/O space, and map PCI transactions to different VMEbus address spaces. Beyond the settings provided for in this power-up option, the target image will possess its other default conditions: the translation offset will be zero, posted writes will be disabled, and only 32-bit (maximum) non-block VMEbus cycles in the non-privileged data space will be generated. This option would typically be used to access permits the use of Boot ROM on another card in the VMEbus system.

PCI Register Access

A power-up option determines if the registers are mapped into Memory or I/O space.

PCI Bus Width

The PCI Interface can be used as a 32-bit bus or 64-bit bus. The PCI bus width is determined during a PCI reset (see Section 4.3.2 of the PCI Specification, Rev. 2.1). The Universe II is configured as 32-bit PCI if REQ64# is high on RST#; it is configured as 64-bit if REQ64# is low. The Universe II has an internal pull-up on REQ64#, so the Universe II defaults to 32-bit PCI. On a 32-bit PCI bus, the Universe II drives all its 64-bit extension bi-direct signals at all times; these signals include: C/BE[7:4]#, AD[63:32], REQ64#, PAR64 and ACK64# to unknown values. If used as a 32-bit interface, the 64-bit pins, AD[63:32], C/BE[7:4], PAR64 and ACK64# may be left unterminated.

The VMEbus Interface supports only a 32-bit PCI bus.

PCI CSR Image Space

There is a power-up option (using the VA[1] pin) that determines the value of the SPACE bit of the PCI_BSx registers. At power-up the SPACE bit of the PCI_BS1 register is the negation of the SPACE bit of the PCI_BS0 register.

- When the VA[1] pin is sampled low at power-up, the PCI_BS0 register’s SPACE bit is set to “1”, which signifies I/O space, and the PCI_BS1 register’s SPACE bit is set to “0”, which signifies Memory space.
- When VA[1] is sampled high at power-up, the PCI_BS0 register’s SPACE register’s bit is set to “0”, which signifies Memory space, and the PCI_BS1 register’s SPACE bit is set to “1”, which signifies I/O space.

Once set, this mapping persists until the next power-up sequence.

See “Memory or I/O Access” on page 2-100, Table B-10 and Table B-12.

Power-Up Option Implementation

This section describes pull-up requirements and timing relevant to the power-up options.

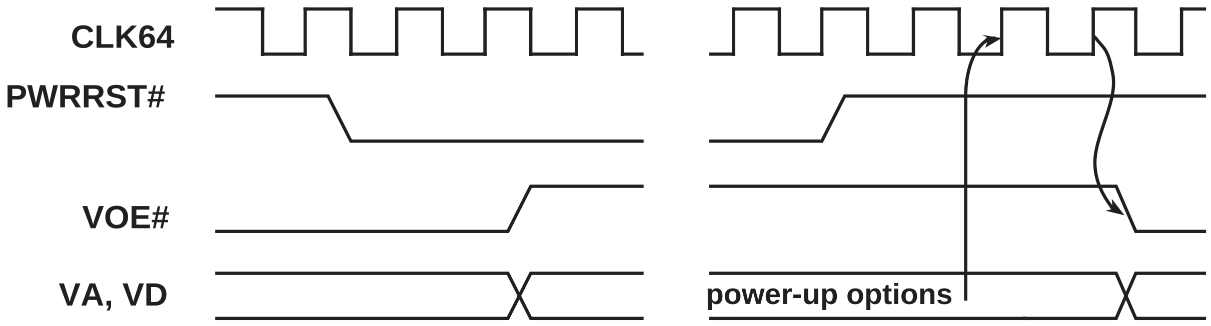
The pull-ups for the general power-up options (if other than default values are required) must be placed on the VA[31:1] and VD[31:27] lines. During reset, the Universe II will negate VOE#, putting these signals into a high-impedance state. While VOE# is negated the pull-ups (or internal pull-downs) will bring the option pins (on A[31:1] and D[31:27]) to their appropriate state.

Caution

The internal pull-downs are very weak. The leakage current on many transceivers may be sufficient to override these pull-downs. To ensure proper operation designers should ensure power-up option pins will go to the correct state.

Within two CLK64 periods after PWRRST# is negated, the Universe II latches the levels on the option pins, and then negates VOE# one clock later. This enables the VMEbus transceivers inwards.

Figure 2-24: Power-up Options Timing



The power-up options are subsequently loaded into their respective registers several PCI clock periods after PWRRST#, SYSRST* and RST# have all been negated.

Note

Because of the power-up configuration, the VMEbus buffers are not enabled until several CLK64 periods after release of SYSRST* (approximately 45 ns). Allowing for worst case backplane skew of 25 ns, the Universe II will not be prepared to receive a slave access until 70 ns after release of SYSRST*.

Hardware Initialization (Normal Operating Mode)

The Universe II has I/O capabilities that are specific to manufacturing test functions. These pins are not required in a non-manufacturing test setting. Table 2-26 below shows how these pins should be terminated.

Table 2-26: Manufacturing Pin Requirements for Normal Operating Mode

Pin Name	Pin Value
TMODE[2]	V_{SS} (or pulled-down if board tests will occasionally be performed, see Auxiliary Test Modes below)
TMODE[1]	
TMODE[0]	
PLL_TESTSEL	V_{SS}
ENID	V_{SS}
PLL_TESTOUT	N/C
VCOCTL	V_{SS}

Test Modes

The Universe II provides two types of test modes: auxiliary modes (NAND tree simulation and High Impedance) and JTAG (IEEE 1149.1).

Auxiliary Test Modes

Two auxiliary test modes are supported: NAND tree and high impedance. The Universe II has three test mode input pins (TMODE[2:0]). For normal operations these inputs should be tied to ground (or pulled to ground through resistors). Table 2-27 below indicates the 3 operating modes of the Universe II. At reset the TMODE[2:0] inputs are latched by the Universe II to determine the mode of operation. The Universe II remains in this mode until the TMODE[2:0] inputs have changed and a reset event has occurred. PLL_TESTSEL must be high for any test mode.

Table 2-27: Test Mode Operation

Operation Mode	TMODE[2:0]	PLL_TESTSEL
Normal Mode	000	0
Accelerate	001	0
Reserved	010	1
Reserved	011	1
NAND Tree Simulation	100	1
Reserved	101	1
High Impedance	110	0/1
Reserved	111	1

For NAND Tree Simulation, the values of the TMODE pins are latched during the active part of PWRRST#. These pins can change state during the NAND Tree tests. The timers are always accelerated in this mode. All outputs are tristated in this mode, except for the VXSYSFAIL output pin.

For High Impedance mode, the values of the TMODE pins are also latched during the active part of PWRRST#. All outputs are tristated in this mode, except for the VXSYSFAIL output pin.

JTAG support

The Universe II includes dedicated user-accessible test logic that is fully compatible with the IEEE 1149.1 Standard Test Access Port (TAP) and Boundary Scan Architecture. This standard was developed by the Test Technology Technical Committee of IEEE Computer Society and the Joint Test Action Group (JTAG). The Universe II’s JTAG support includes:

- A five-pin JTAG interface (TCK, TDI, TDO, TMS, and TRST#),
- a JTAG TAP controller,
- a three-bit instruction register,
- a boundary scan register,
- a bypass register,
- and an IDCODE register.

The following required public instructions are supported: BYPASS (3'b111), SAMPLE(3'b100), and EXTEST(3'b000). The optional public instruction IDCODE(3'b011) selects the IDCODE register which returns 32'b01e201d. The following external pins are not part of the boundary scan register: LCLK, PLL_TESTOUT, PLL_TESTSEL, TMODE[3:0], and VCOCTL.A BSDL file is available upon request from Tundra Semiconductor Corporation.

Clocks

CLK64 is a 64 MHz clock that is required by the Universe II in order to synchronize internal Universe II state machines and to produce the VMEbus system clock (VSYSCLK) when the Universe II is system controller (SYSCON). This clock is specified to have a minimum 60-40 duty cycle with a maximum rise time of 5 ns. Using a different frequency is not recommended as it will alter various internal timers and change some VME timing.

Chapter 3

Auxiliary Functions

The material in Chapter 3 is exclusively GE Fanuc.

Auxiliary Bus Timeout Timer

The GE Fanuc PCI-to-VMEbus interface contains a programmable bus timeout timer which functions only when the interface is the system controller. The PCI-to-VMEbus interface also has an auxiliary bus timeout timer which can be enabled to function when the board is not the system controller. This timer is enabled by setting the BTO bit (bit 3) in the System COMM register located at $D800E in memory. The timer has four programmable timeout periods which are set by the BTOV(1:0) bits (bit 5, 4 respectively) in the System COMM register. The four settings are shown in Table 3-1.

Table 3-1: Auxiliary Bus Timeout Timer Settings

BTOV1	**BTOV0**	**Timeout Period (PCI CLK = 25,30,33 MHz)**			
0	0	21	18	16	ms
0	1	85	70	64	ms
1	0	341	282	256	ms
1	1	1.33	1.09	1.00	ms

As shown above, the timeout period is a function of the PCI clock frequency. Care must be taken when enabling the auxiliary timer when the PCI-to-VMEbus interface is the system controller to ensure that only one timer is enabled.

Auxiliary BERR Interrupt

The Universe II chip is capable of capturing the VME address and address modifiers when a BERR occurs on the VME bus for decoupled transactions only. The SBC contains auxiliary circuitry, external and independent of the Universe II chip, capable of capturing the VME address and address modifiers of any transaction which results in a BERR and of which the SBC is master. This circuitry is capable of generating an interrupt if enabled.

The auxiliary BERR circuitry works as follows: The logic is enabled by setting Auxiliary BERR Logic Enable bit (ABLE, bit 2) in the System COMM register located at $D800E in memory. Once enabled the circuitry will, upon receiving a VME BERR from a VME transaction of which the SBC is master, capture and hold the VME address and VME address modifiers which caused the BERR. The VME address will be a long word located in the VME BERR Address register (VBAR) at $D8010 in memory and the VME address modifiers will be the bits 5 - 0 of a byte located in the VME BERR Address Modifier Register at $D8014 in memory.

When a BERR occurs the BERR Status Read/Clear bit (BERRST, bit 7) of the System COMM register will be set. The offending VME Address and Address Modifiers will be held until the BERR Status Read/Clear bit is cleared. The BERR Status Read/Clear bit is cleared by writing a one to this bit. This is most easily done by reading the System COMM register and writing it back into the register. If the BERR Status Read/Clear is set it will clear, if it is not set nothing will change.

By setting the BERR Interrupt enable (BERRI, bit 6) in the System COMM register, an interrupt will be generated on PCI int A when the auxiliary BERR circuitry is enabled as explained above and a BERR occurs from a VME transaction of which the SBC is master. The interrupt is cleared by clearing the BERR Status Read/Clear (BERRST, bit 7) of the System COMM register as explained above.

Chapter 4 Endian Conversion

The material in Chapter 4 is exclusively GE Fanuc.

VMEbus Byte Lanes

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

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

It is important to note the major byte-ordering differences between the Motorola and the Intel based microprocessor. In addition, communication between Motorola-compatible 680X0 VMEbus modules and the PCI-to-VMEbus interface requires special attention to avoid byte-ordering conflicts.

Table 4-1: VMEbus Byte Assignment to the Data Lines

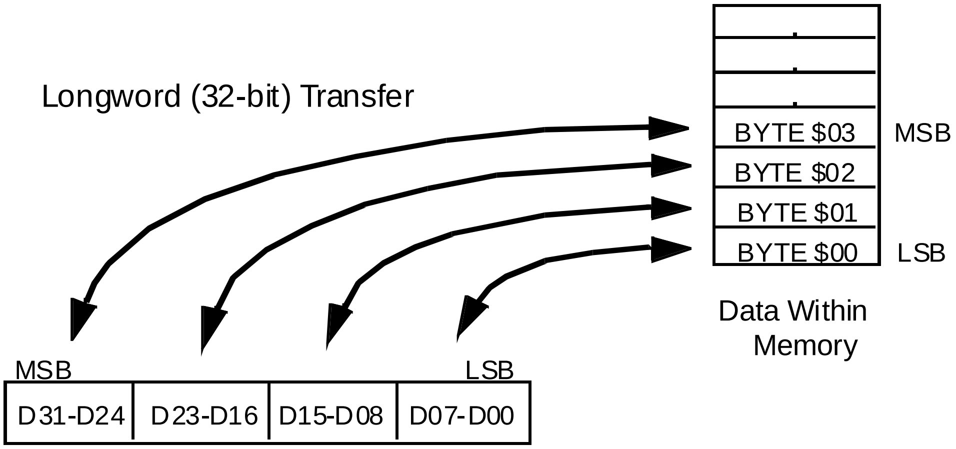
DTB CYCLE TYPE	D31-D24	D23-D16	D15-D08 EVEN ADDRESS	D07-D00 ODD ADDRESS
D08(EO) EVEN OR ODD				
Single Odd Byte(3)				Byte(3)
Single Even Byte(2)			Byte(2)	
Single Odd Byte(1)				Byte(1)
Single Even Byte(0)			Byte(0)	
D08(O) ODD ONLY				
Single Odd Byte(3)				Byte(3)
Single Odd Byte(1)				Byte(1)
D16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

Byte Ordering: Big Endian / Little Endian

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

The GE Fanuc PCI-to-VMEbus interface uses an Intel based or equivalent microprocessor, which uses little-endian byte ordering. Byte arrangement and the byte relationship between data in the processor and transferred data in memory are shown in Figure 4-1.

Figure 4-1: Byte Relationships Using the Little-Endian Pentium Microprocessor

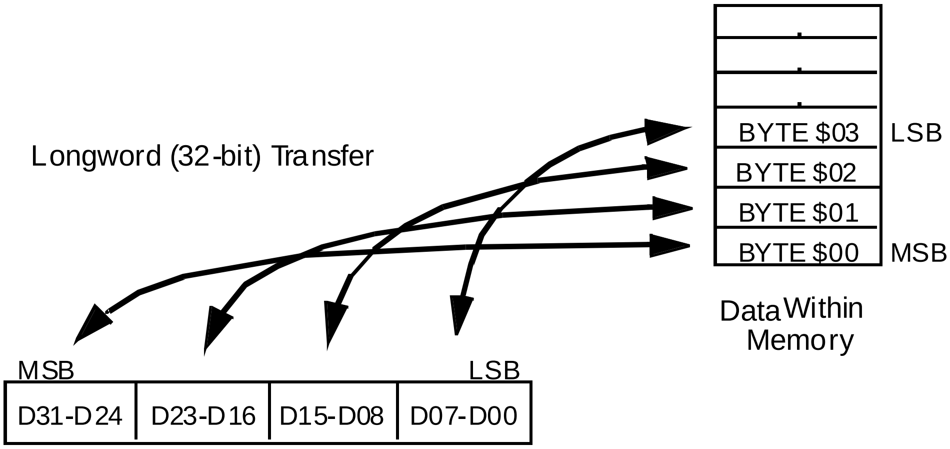


Note that in little-endian processors like the Pentium, the processor's least significant byte is stored in the lowest byte address after a multiple-byte write (such as the longword transfer illustrated), while the processor's most significant byte is stored in the highest byte address after such transfers.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

Contrast the behavior of the little-endian Pentium in Figure 4-1 with the same longword transfer using a big-endian processor like the Motorola 68040 in Figure 4-2 on page 4-3.

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

Figure 4-2: Byte Relationships Using the Big-Endian 68040 Microprocessor



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

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

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

Endian Conversion Hardware

The Universe II chip performs Address Invariant translation between the PCI and VMEbus interfaces. Address Invariant mapping or “Non-endian conversion” mode maintains the byte ordering between the two interfaces (i.e. data originating in little-endian mode on the CPU/PCI side will remain in little-endian mode on the VMEbus side of the interface). However, the GE Fanuc PCI-to-VMEbus interface has external endian conversion logic which allows the application to perform independent master/slave hardware endian conversion. The enables for the circuitry (MEC,SEC) are located in the System COMM register, bits 0,1.

Unaligned Transfers with Endian Conversion Enabled

The GE Fanuc PCI-to-VMEbus interface was designed to be consistent with the endian conversion approach that was used on previous GE Fanuc products. That approach is: defeat endian conversion for unaligned transfers to/from VMEbus - two cases of 3-byte, one unaligned 2-byte case.

The defeat on unaligned transfers (UAT) philosophy was adopted given that data sharing between CPUs with different endianess only makes sense for longword aligned and word aligned data. However, the Universe II chip breaks up 3-byte transfers on the PCI bus into two VMEbus transfers: a byte access and a word access. Our endian conversion hardware “sees” the word access (which is word aligned) and performs a byte swap. This causes a word within the 3-byte entity to be swapped. To avoid this inadvertent swapping, the user should disable master endian conversion when performing any unaligned transfers to the VMEbus. This only affects transfers originating from the microprocessor and destined for the VMEbus.

PCI Bus Data Combining: Byte Swap

The GE Fanuc PCI-to-VMEbus interface performs endian conversion (byte swapping) based on size. If a longword access is performed, it swaps the two words and the two bytes within those words. If a word access is performed, it swaps the two bytes. If byte access is performed, no swapping occurs.

Unfortunately, successive writes from the microprocessor to VME through the host bridge are subject to combining: four bytes may combine to form one longword, two words may combine to form a longword, two bytes may combine to form a word. Since the endian conversion logic swaps on size, the data combining causes data to be swapped when it shouldn't have been, or longword swaps to occur when word swaps should have occurred.

This phenomena only occurs for microprocessor to VME write accesses: reads are not affected, nor are VME slave accesses or DMA BLTs.

The data combining can be defeated by programming the Universe II VME master channel (PCI slave channel) to be longword, word or byte wide depending on the size of accesses being generated from the microprocessor. Data combining may still occur on the PCI bus, but the Universe II chip will only generate the programmed size transfers on VMEbus. Thus, the endian conversion logic will "see" the data in the size that is consistent with swap protocol. Again, reads, VME slave accesses in either direction, or DMA BLTs are not affected.

Please refer to Chapter 6 Universe II Errata And Notes for additional information.

Chapter 5 PCI/VMEbus Deadlock

The material in Chapter 5 is exclusively GE Fanuc.

Scenario Overview

There is a deadlock scenario which is inherent to systems containing a PCI/VME interface. The PCI specification allows for and defines devices called bridges. A bridge could be a host bridge which interfaces a host CPU to the system PCI bus. Another type of bridge is the PCI-to-PCI bridge which bridges two different PCI buses.

In order to optimize system performance, PCI specification allows bridges to provide buffering. Almost all host bridges and PCI-PCI bridges contain write-posting buffers. These buffers allow writes from one side of the bridge to be acknowledged before the data is actually written to the other side of the bridge.

Without write posting, the CPU or PCI initiator would have to wait for the device that is receiving the data to acknowledge the data transfer before it could proceed to the next bus transaction. In addition to allowing for write posting, the PCI bridge specification imposes transaction ordering rules on the bridge design to ensure data consistency in the system. These ordering rules are imposed on host bridges and on PCI-to-PCI bridges. One ordering rule is that reads cannot traverse across a bridge until all write-posted data has been flushed. In a PCI/VME system, this ordering rule coupled with the lack of retry on the VMEbus create the potential for system deadlocks.

An Example

To understand the need for ordering rules and the potential for deadlock, examine the following scenario.

Suppose the CPU writes a buffer to a VMEbus SRAM board, and then sets a flag in the DRAM. Meanwhile, a VME master, such as a 68000 CPU, is polling the flag in DRAM, to determine when the SRAM buffer is ready to process. The ordering rule guarantees that the VMEbus master will not “see” the status flag set in DRAM until all the VMEbus write data has been flushed from the host bridge. However, a deadlock condition will occur if the PCI/VME interface has been granted the PCI bus to perform a read from DRAM while write posting data destined for the VMEbus is contained in the host bridge. In this case, the write posted data cannot flush (complete on VME) since the VMEbus is being held by the 68 K CPU, which cannot complete its read because the host bridge will not allow the read to occur, thus a DEADLOCK occurs.

The above deadlock scenario is inherent in PCI systems that interface to busses that do not have retry capability. Since the VMEbus does not currently provide for bus retry, the resulting deadlock condition will result in a VMEbus error (BERR) condition.

Possible Solutions

The simplest solution is to never simultaneously enable the VMEbus master and slave interfaces; however, in most systems this is not practical.

If simultaneous VMEbus master/slave interfacing is required (i.e. shared memory applications), the user can use the Universe VMEbus ownership bit in the MAST_CTL register to guarantee exclusive access to the VMEbus. The user would employ the following protocol when using the VME ownership bit:

1. Set the VME ownership bit in the MAST_CTL register.
2. Poll the VOWN_ACK bit which is asserted when the VMEbus has been acquired.
3. Once the VOWN_ACK bit is asserted, perform VMEbus master write operations.
4. Clear the VME ownership bit.

Note that the assertion of the VOWN_ACK bit can be programmed to generate a PCI interrupt.

Chapter 6 Universe II Errata and Notes

The material in Chapter 6 is exclusively GE Fanuc.

Introduction

The Universe II interface chip is a second-generation integrated circuit that fixes errata associated with its predecessor and provides new product features. This chapter describes all of the errata associated with the Universe II and the work-around and/or fixes provided with the VMEbus interface. Table 6-1 below shows a matrix of Universe II errata and solutions. Also included in this chapter is a description of the new features available with the Universe II chip and information regarding compatibility between Universe I and Universe II.

Table 6-1: Universe II Errata and Solutions

Errata #	Description	Hardware Workaround	Software Workaround	Not Fixed
1	PCI Base Registers		BIOS Does Not Map At Address 0000	
2	SYSCLK Generation During SYSRESET	Hardware Designed To Generate SYSCLK during SYSRESET		
3	Invalid Release Of LOCK# During Exclusive Access		Use Semaphores To Gain Exclusive Access To Resources	
4	DY4 Auto-ID Incompatibility			DY4 Auto-ID Not Used And Not Supported
5	Simultaneous, Single-Level Interrupts	Hardware Designed To Prevent This Condition		

Universe II Errata

The following are brief descriptions of all of the device errata associated with the Universe II interface chip and the solution or work-around used in the VMEbus interface. For complete errata descriptions, please see the Tundra resource listed in the Overview chapter.

1. PCI BASE REGISTERS

   Problem: PCI base registers accept “zero” as valid decodable address which violates PCI 2.1 spec.

   Solution: Software must be aware that Universe II register space is not disabled but instead moved to a desired location. The BIOS in GE Fanuc products set the base registers to a non-zero value. Care must be taken by the user not to set these registers to zero.

   ### Note

   The Universe II decodes for both memory and I/O space. Therefore, software must appropriately position the Universe II registers. For backward software compatibility, only access the Universe II registers through memory space accesses.

2. SYSCLK GENERATION DURING SYSTEM RESET

   Problem: The Universe II stops the generation of SYSCLK during system reset, when the Universe II is the SYSCON and BGIN3 is asserted.

   Solution: GE Fanuc products contain hardware to generate SYSCLK continuously when operating as a system controller.

   ### Note

   Very few boards in production today rely upon SYSCLK during reset for proper operation.

3. INVALID RELEASE OF PCI LOCK# DURING EXCLUSIVE ACCESSES

   Problems: When an external VME master generates a VMEbus LOCK/ADOH to the Universe II, the cycle is processed on the PCI bus as a read cycle with the LOCK# signal asserted. If the target responds with a Master Completion Termination, then the Universe II will improperly relinquish LOCK# even though VME BBSY* is still asserted.

   Solution: The Universe II's semaphores can be used to ensure exclusive accesses. Refer to Chapter 2 of this manual.

4. DY4 AUTO-ID INCOMPATIBILITY

   Problem: The propagation delay through the Universe II from IACKIN* to IACKOUT* will be five clocks instead of four when using the DY4 Auto-ID mechanism. This may cause failure of existing DY4 Auto-ID algorithms to correctly identify related Universe II board positions.

   Solution: The GE Fanuc supplied software does not make use of the DY4 Auto-ID method of detection. Software written to perform the DY4 Auto-ID detection must account for the extra clock cycle.

5. SIMULTANEOUS, SINGLE LEVEL INTERRUPTS

   Problem: The Universe II incorrectly qualifies the assertion of VSLAVE_DIR (which enables DTACK* onto the backplane) on IACK* assertion rather than a combination of IACK* and IACKIN*. The assertion of VSLAVE_DIR will assert DTACK* high onto the VME backplane, where it will remain high until the IACK cycle is complete. If a number of Universe II boards simultaneously assert the same level IRQ (as in the case of VME64 Auto ID) there is a chance that the card attempting to properly respond to the IACK cycle with DTACK* low will never be seen because all of the other Universe II cards will be driving DTACK* high.

   Solution: GE Fanuc products contain hardware to correct this errata.

Universe II Design Notes

Table 6-2 below shows a matrix of Universe II design notes and solutions.

Table 6-2: Universe II Design Note Matrix

Design Note #	Description	Hardware Workaround	Software Workaround	Not Fixed
1	DMA Operation During PCI Reads		Careful Programming Of DMA Transfer Size	
2	Potential For DTACK*/AS* Deadlock		Enable Coupled Window Timer When In Coupled Mode	Not Fixed For IACK Cycles

The following are brief descriptions of all of the design notes associated with the Universe II interface chip. For complete design note descriptions, please see the Tundra resource listed in the Overview chapter.

1. DMA OPERATION DURING PCI READS

   Problem: Universe II prefetches on DMA reads from the PCI bus up to the aligned address boundary defined in the PABS field of the MAST_CTL register. Therefore, Universe II may read beyond the programmed transfer length. This extra data is not transferred to the VMEbus.

   Solution: Prefetching can be avoided by programming the DMA for transfers that terminate at the PABS boundary. If further data is required beyond the boundary, but before the next boundary, the DTBC register may be programmed to eight byte transfers. Note however that programming the DTBC to less than eight bytes will still result in eight bytes fetched from PCI. In all cases, the correct amount of data is transferred to the VMEbus.

2. POTENTIAL FOR DTACK*/AS* DEADLOCK WITH SOME SLAVE CARDS

   Problem: When performing IACK cycles and coupled 8 or 16 bit cycles as a VME master, the Universe II will wait for the VME slave to release DTACK* before it removes AS*. If the slave card links release of DTACK* to the release of AS*, a deadlock condition will result where both master and slave wait for each other to end the cycle. Slave cards should not assume any relationship between DTACK* and AS* as the ANSI VME64 specification does not state any relationship.

   Solution: If a slave card is designed such that the deadlock condition might occur, deadlock may be avoided during coupled cycles by enabling the Coupled Window Timer in the LMISC register. The condition cannot be disabled for IACK cycles.

3. NOISE ON VME DATA STROBES

   Problem: The Universe II may, under some circumstances, be sensitive to noise on the rising edge of the VME data strobes particularly in large systems with heavily loaded backplanes.

   Solution: None. This design note results from a single case reported to Tundra where the Universe II operated on a heavily loaded backplane. This is not expected to be a problem for most users.

4. 16-BIT PCI BUS BURST TRANSFERS SPLIT INTO 8-BIT BYTES

   Problem: GE Fanuc has observed that when performing a PCI bus 16-bit burst to a Universe II PCI slave image with write posting enabled may, in some cases, result in transfers on the VMEbus as 8-bit bytes. This causes incorrect conversion by the endian conversion hardware.

   Solution: None. If PCI bus data is to be transferred as words (16-bits) and endian conversion is required, do not enable write posting in the Universe II.

Other Compatibility Issues

During testing of Universe II components, GE Fanuc found the following issues concerning compatibility of software between Universe I and Universe II.

1. UNALIGNED TRANSFERS

   Problem: During unaligned transfers, the Universe II arranges the bytes of the word or long word different from the Universe I.

   Solution: Software must keep track of which device is being used and swap the bytes appropriately. The device in use is indicated by the revision ID (RID) field of the PCI_CLASS register. A value of 0 indicates the presence of a Universe I chip. A value of 1 indicates the presence of a Universe II chip.

2. AM CODE ERROR LOG

   Problem: The Universe I set the multiple error bit in the VMEbus AM code error log (V_AMERR) for any amount of errors, even single errors. The Universe II fixes this problem. However, software that expects this bit to be set for single errors will not operate correctly with a Universe II.

3. USER DEFINED AM

   Problem: The Universe I allows any address modifier to be programmed into the User Defined AM Codes Register (USER_AM) regardless of whether it is actually a VMEbus specification defined “User Defined AM” ranging from 0x10 to 0x1F. The Universe II only allows the User Defined AM Codes Register (USER_AM) to be programmed to values from 0x10 to 0x1F.

Universe II Changes

This section provides information about the operational changes made in the Universe II as they relate to compatibility between the Universe I and Universe II.

Register Reset Values

The Universe I interface chip contained many register bits that had undefined values after reset. The Universe II’s register bits are always defined after reset, as indicated in Appendix B. This should present no compatibility problem for software running on systems using Universe I or Universe II.

Coupled Request Timer

The Universe II no longer uses the coupled request timer (CRT) value in the LMISC register. The coupled request phase will expire if no coupled transfer occurs within 215 PCI clock cycles on the Universe II. Since the CRT can still be read and written, this should not present a compatibility problem with existing software.

MFUNCT Field In PCI_MISC0

The Universe I set the MFUNCT field in the PCI_MISC0 field to 1. This required that the Universe I examine the state of the AD[10:8] lines during Type 0 configuration accesses. The MFUNCT field in the Universe II is now set to 0. This should not present a compatibility problem with existing software.

Config Type 1 Accesses

Config Type 1 cycle mapping to the VMEbus is not available in the Universe II. Accordingly, the upper "Type 1 Configuration Space" bit is removed from the LAS fields in the Universe II's PCI Slave Image Control registers (LSIxx_CTL). The remaining LAS bit allows for the selection of either Memory or I/O space accesses.

PCI Base Address Registers

The Universe I did not decode the base address properly and thus took 64K of I/O or memory space. The Universe II corrects this problem and also decodes for both I/O and memory space simultaneously. However, the space allocated for each is programmable with 4K bytes resolution through the use of the PCI_BS and PCI_BS1 registers.

The PCI_BS[15:12] bits of the PCI_BS register are no longer hardwired to logic zero. However, these bits power-up in a logic low state such that existing Universe software should not be affected.

DGCS VON[3]

Bit 23 of the DMA General Control/Status Register was not used in the Universe I. The Universe II now reserves this bit.

IACKIN* Monitoring

When configured as SYSCON, the Universe II will monitor IACK* rather than IACKIN*. This permits it to operate as the SYSCON in places other than slot 1. The slot with SYSCON in it becomes a virtual slot 1.

Rescinding DTACK

The Universe II ignores the state of the RESCIND bit in the MISC_CTL register and always rescinds DTACK. This bit may still be read and written such that existing software should not be affected.

Reset Operation

The Universe I only loads power-up options during power-up reset. The power-up options are cleared on the Universe I during PCI reset (RST#) or VME reset (SYSRESET*). The Universe II latches the power-up values during PWRRST# such that subsequent SYSRESET* or RST# activity will restore power-up values. This should present no compatibility problems.

Universe II Additions

The Universe II interface chip has many additional features and enhancements over its predecessor, the Universe I. The following section provides information about the features and enhancements as they relate to compatibility between the two chips and is divided into two parts: General Feature Additions/Enhancements, and Performance Enhancements.

General Feature Additions/Enhancements

Mailboxes

The Universe II now includes four, 32bit mailbox registers. Writing control and status data to these registers cause an interrupt to occur on either or both PCI and VMEbus buses. Interrupt generation is controlled by the Universe II interrupt enable registers and two new interrupt map registers. These mailbox registers are available for use by software on the VMEbus interface. However, software that makes use of these mailboxes will not operate properly on the Universe I chip. It is therefore recommended that software use the PLX9060ES mailboxes for backward software compatibility.

Location Monitor

The Universe II contains a location monitor that allows for broadcast events across the VME backplane. Each Universe II that shares the same enabled location monitor image will respond to a read or write within that range by generating one of four internal Universe “interrupts”. Each interrupt can be individually enabled and mapped to specific PCI interrupts. The action of the location monitor is controlled by the Location Monitor Control register, the Location Monitor Base Address Register, and the Local Interrupt Map 2 Register. The Universe I does not support the location monitor and software written to use the location monitor will not function on a Universe I.

Additional Slave Images

The Universe II provides four additional slave images for the PCI bus and the VMEbus. Three of these images have 64K byte resolutions and one has a 4K byte resolution. For backward compatibility, software should not make use of the additional slave images.

VME Software Interrupts

The Universe II has expanded VME software interrupts such that each of the seven IRQ levels can be generated directly by writing to the VME Interrupt Enable Register. The status of the interrupts can be verified by reading the VME Interrupt Status Register. Software that utilizes the new VME interrupt features will not function on a Universe I chip.

Semaphores

The Universe II has eight semaphores, accessible by two new registers. Each register has a status bit and a 7 bit tag field for each of the semaphores. The semaphores are used to ensure exclusive access to system resources on either the VMEbus or the PCI bus. Software that utilize semaphores will not function on a Universe I chip.

New SCYC_CTL LAS Field

The Universe II now supports both memory and I/O accesses to the special cycle generator image. A new LAS bit has been added to the SCYC_CTL register to support this new feature. Software written to utilize this feature will not function on the Universe I chip.

Performance Enhancements

Early Release Of BBSY*

The Universe II will execute an early release of BBSY* when possible to optimize the use of the VMEbus. This enhancement is transparent to software and should present no compatibility problems.

VOFF/VON

The Universe II adds three new settings, 2, 4, and 8µ seconds, to the VOFF timer found in the DGCS to allow for “fine-tuning” of VMEbus usage. Software that makes use of these new settings will not function on a Universe I.

Aligned Burst Size

The Universe II provides a new setting of 128 bytes to the PCI aligned burst size (PABS) field of the MAST_CTL register. Software that makes use of the new burst size will not function on a Universe I.

PCI Bus Parking

The Universe II will only assert REQ# if it is not already the PCI bus master, possibly saving one clock cycle delay before starting a PCI bus transaction. This enhancement is transparent to software and should present no compatibility problems.

Universe I/Universe II Detection

Software may be written to support the enhanced features of the Universe II chip while remaining compatible with Universe I devices. This can be accomplished by detecting the type of device being used and enabling the Universe II enhanced features only when a Universe II is detected. The device in use may be detected by reading the revision ID (RID) field of the PCI_CLASS register. A value of 0 indicates the presence of a Universe I chip. A value of 1 indicates the presence of a Universe II chip. By detecting the presence of a Universe II chip, software may automatically enable Universe II features that improve the performance of VMEbus transactions.

Chapter 7 Description of Signals

Introduction

The following detailed description of the Universe II signals is organized according to these functional groups:

Table 7-1: VMEbus Signals

CLK64	Input
Reference Clock – this 64MHz clock is used to generate fixed timing parameters. It requires a 50-50 duty cycle (±20%) with a 5ns maximum rise time. CLK64 is required to synchronize the internal state machines of the VME side of the Universe II.	
VA [31:1]	**Bidirectional**
VMEbus Address Lines 31 to 01 – during MBLT transfers, VA 31-01 serve as data bits D63-D33. VA03-01 are used to indicate interrupt level on the VMEbus.	
VA_DIR	**Output**
VMEbus Address Transceiver Direction Control – the Universe II controls the direction of the address (VA31-01, VLWORD#) transceivers as required for master, slave and bus isolation modes. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.	
VAM [5:0]	**Bidirectional**
VMEbus Address Modifier Codes – these codes indicate the address space being accessed (A16, A24, A32), the privilege level (user, supervisor), the cycle type (standard, BLT, MBLT) and the data type (program, data).	
VAM_DIR	**Output**
VMEbus AM Code Direction Control – controls the direction of the AM code transceivers as required for master, slave and bus isolation modes. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.	
VAS#	**Bidirectional**
VMEbus Address Strobe – the falling edge of VAS# indicates a valid address on the bus. By continuing to assert VAS#, ownership of the bus is maintained during a RMW cycle.	

Table 7-1: VMEbus Signals (continued)

<table>
<tr><th>VAS_DIR</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Address Strobe Direction Control</b> – controls the direction of the address strobe transceiver as required for master, slave and bus isolation modes. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.</td></tr>
<tr><th>VBCLR#</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Bus Clear</b> – requests that the current owner release the bus.
Asserted by the Universe II when configured as SYSCON and the arbiter detects a higher level pending request.</td></tr>
<tr><th>VBGI# [3:0]</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Bus Grant Inputs</b> – The VME arbiter awards use of the data transfer bus by driving these bus grant lines low. The signal propagates down the bus grant daisy chain and is either:
accepted by a requester if it requesting at the appropriate level, or
passed on as a VBGO [3:0]# to the next board in the bus grant daisy chain.</td></tr>
<tr><th>VBGO# [3:0]</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Bus Grant Outputs</b> – Only one output is asserted at any time, according to the level at which the VMEbus is being granted.</td></tr>
<tr><th>VD [31:0]</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus Data Lines 31 through 00</b></td></tr>
<tr><th>VD_DIR</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Data Transceiver Direction Control</b> – the Universe II controls the direction of the data (VD [31:0]) transceivers as required for master, slave and bus isolation modes. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.</td></tr>
<tr><th>VDS# [1:0]</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus Data Strobes</b> – the level of these signals are used to indicate active byte lanes:
During write cycles, the falling edge indicates valid data on the bus.
During read cycles, assertion indicates a request to a slave to provide data.</td></tr>
<tr><th>VDS_DIR</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Data Strobe Direction Control</b> – controls the direction of the data strobe transceivers as required for master, slave and bus isolation modes. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.</td></tr>
<tr><th>VDTACK#</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus Data Transfer Acknowledge</b> – VDTACK# driven low indicates that the addressed slave has responded to the transfer. The Universe II always rescinds DTACK*. It is tristated once the initiating master negates AS*.</td></tr>
<tr><th>VIACK#</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus Interrupt Acknowledge</b> – indicates that the cycle just beginning is an interrupt acknowledge cycle.</td></tr>
<tr><th>VIACKI#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Interrupt Acknowledge In</b> – Input for IACK daisy chain driver. If interrupt acknowledge is at same level as interrupt currently generated by the Universe II, then the cycle is accepted. If interrupt acknowledge is not at same level as current interrupt or Universe II is not generating an interrupt, then the Universe II propagates VIACKO#.</td></tr>
</table>

Table 7-1: VMEbus Signals (continued)

<table>
<tr><th>VIACKO#</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Interrupt Acknowledge Out</b>– generated by the Universe II if it receives VIACKI# and is not currently generating an interrupt at the level being acknowledged.</td></tr>
<tr><th>VLWORD#</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus Longword Data Transfer Size Indicator</b> – this signal is used in conjunction with the two data strobes VDS [1:0]# and VA 01 to indicate the number of bytes (1 – 4) in the current transfer. During MBLT transfers VLWORD# serves as data bit D32.</td></tr>
<tr><th>VOE#</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Transceiver Output Enable</b> – used to control transceivers to isolate the Universe II from the VMEbus during a reset or BI-mode. On power-up, VOE# is high (to disable the buffers).</td></tr>
<tr><th>VRACFAIL#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus ACFAIL Input signal</b> – warns the VMEbus system of imminent power failure. This gives the modules in the system time to shut down in an orderly fashion before powerdown. ACFAIL is mapped to a PCI interrupt.</td></tr>
<tr><th>VRBBSY#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Receive Bus Busy</b> – allows the Universe II to monitor whether the VMEbus is owned by another VMEbus master</td></tr>
<tr><th>VRBERR#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Receive Bus Error</b> – a low level signal indicates that the addressed slave has not responded, or is signalling an error.</td></tr>
<tr><th>VRBR# [3:0]</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Receive Bus Request Lines</b> – if the Universe II is the Syscon, the Arbiter logic monitors these signals and generates the appropriate Bus Grant signals. Also monitored by requester in ROR mode.</td></tr>
<tr><th>VRIRQ# [7:1]</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Receive Interrupts 7 through 1</b> – these interrupts can be mapped to any of the Universe II’s PCI interrupt outputs. VRIRQ7-1# are individually maskable, but cannot be read.</td></tr>
<tr><th>VRSYSFAIL#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Receive SYSFAIL</b> – asserted by a VMEbus system to indicate some system failure. VRSYSFAIL# is mapped to a PCI interrupt.</td></tr>
<tr><th>VRSYSRST#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Receive System Reset</b> – causes assertion of LRST# on the local bus and resets the Universe II.</td></tr>
<tr><th>VSLAVE_DIR</th><th>Output</th></tr>
<tr><td colspan="2"><b>VMEbus Slave Direction Control</b> – transceiver control that allow the Universe II to drive DTACK* on the VMEbus. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.</td></tr>
<tr><th>VSYSCLK</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus System Clock</b> – generated by the Universe II when it is the Syscon and monitored during DY4 Auto ID sequence</td></tr>
<tr><th>VSCON_DIR</th><th>Output</th></tr>
<tr><td colspan="2"><b>Syscon Direction Control</b> – transceiver control that allows the Universe II to drive VBCLR# and SYSCLK. When the Universe II is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe II, this signal is driven low.</td></tr>
<tr><th>VWRITE#</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>VMEbus Write signal</b> – indicates the direction of data transfer.</td></tr>
</table>

Table 7-1: VMEbus Signals (continued)

VXBBSY	Output
VMEbus Transmit Bus Busy Signal – generated by the Universe II when it is VMEbus master	
VXBERR	**Output**
VMEbus Transmit Bus Error Signal – generated by the Universe II when PCI target generates target abort on coupled PCI access from VMEbus.	
VXBR [3:0]	**Output**
VMEbus Transmit Bus Request – the Universe II requests the VMEbus when it needs to become VMEbus master.	
VXIRQ [7:1]	**Output**
VMEbus Transmit Interrupts – the VMEbus interrupt outputs are individually maskable.	
VXSYSFAIL	**Output**
VMEbus System Failure – asserted by the Universe II during reset and plays a role in VME64 Auto ID.	
VXSYSRST	**Output**
VMEbus System Reset – the Universe II output for SYSRST*.	

Table 7-2: PCI Bus Signals

ACK64#	Bidirectional
Acknowledge 64-bit Transfer – when driven by the PCI slave (target), it indicates slave can perform a 64-bit transfer.	
AD [31:0]	**Bidirectional**
PCI Address/Data Bus – address and data are multiplexed over these pins providing a 32-bit address/data bus.	
AD [63:32]	**Bidirectional**
PCI Address/Data Bus – address and data are multiplexed over these pins providing 64-bit address and data capability.	
C/BE# [7:0]	**Bidirectional**
PCI Bus Command and Byte Enable Lines – command and byte enable information is multiplexed over all eight C/BE lines. C/BE [7:4]# are only used in a 64-bit PCI bus	
DEVSEL#	**Bidirectional**
PCI Device Select – is driven by the Universe II when it is accessed as PCI slave.	
ENID	**Input**
Enable IDD Tests – required for ASIC manufacturing test, tie to ground for normal operation.	
FRAME#	**Bidirectional**
Cycle Frame – is driven by the Universe II when it is PCI initiator, and is monitored by the Universe II when it is PCI target	
GNT#	**Input**
PCI Grant – indicates to the Universe II that it has been granted ownership of the PCI bus.	
IDSEL	**Input**
PCI Initialization Device Select – is used as a chip select during configuration read and write transactions	

Table 7-2: PCI Bus Signals (continued)

LINT# [7:0]	**Bidirectional (Open Drain)**
PCI Interrupt Inputs – these PCI interrupt inputs can be mapped to any PCI bus or VMEbus interrupt output.	
IRDY#	**Bidirectional**
Initiator Ready – is used by the Universe II as PCI master to indicate that is ready to complete a current data phase.	
LCLK	**Input**
PCI Clock – provides timing for all transactions on the PCI bus. PCI signals are sampled on the rising edge of CLK, and all timing parameters are defined relative to this signal. The PCI clock frequency of the Universe II must be between 25 and 33MHz. Lower frequencies will result in invalid VME timing.	
LOCK#	**Bidirectional**
Lock – used by the Universe II to indicate an exclusive operation with a PCI device. While the Universe II drives LOCK#, other PCI masters are excluded from accessing that particular PCI device. Likewise, when the Universe II samples LOCK#, it may be excluded from a particular PCI device.	
LRST#	**Output**
PCI Reset Output – used to reset PCI resources.	
PAR	**Bidirectional**
Parity – parity is even across AD [31:0] and C/BE [3:0] (the number of 1s summed across these lines and PAR equal an even number).	
PAR64	**Bidirectional**
Parity Upper DWORD – parity is even across AD [63:32] and C/BE [7:4] (the number of 1s summed across these lines and PAR equal an even number).	
PERR#	**Bidirectional**
Parity Error – reports parity errors during all transactions. The Universe II drives PERR# high within two clocks of receiving a parity error on incoming data, and holds PERR# for at least one clock for each errored data phase.	
PLL_TESTOUT	**Output**
Manufacturing Test Output – No connect	
PLL_TESTSEL	**Input**
Manufacturing Test Select – tie to ground for normal operation	
PWRRST#	**Input**
Power-up Reset – all Universe II circuitry is reset by this input.	
REQ#	**Output**
Bus Request – used by the Universe II to indicate that it requires the use of the PCI bus.	
REQ64#	**Bidirectional**
64-Bit Bus Request– used to request a 64-bit PCI transaction. If the target does not respond with ACK64#, 32-bit operation is assumed.	
RST#	**Input**
PCI Reset Input – resets the Universe II from the PCI bus.	
SERR#	**Bidirectional**
System Error – reports address parity errors or any other system error.	

Table 7-2: PCI Bus Signals (continued)

<table>
<tr><th>STOP#</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>Stop</b> – used by the Universe II as PCI slave when it wishes to signal the PCI master to stop the current transaction. As PCI master, the Universe II will terminate the transaction if it receives STOP# from the PCI slave.</td></tr>
<tr><th>TCK</th><th>Input</th></tr>
<tr><td colspan="2"><b>JTAG Test Clock Input</b> – used to clock the Universe II TAP controller. Tie to any logic level if JTAG is not used in the system.</td></tr>
<tr><th>TDI</th><th>Input</th></tr>
<tr><td colspan="2"><b>JTAG Test Data Input</b> – used to serially shift test data and test instructions into the Universe II. Tie to any logic level if JTAG is not used in the system.</td></tr>
<tr><th>TDO</th><th>Output</th></tr>
<tr><td colspan="2"><b>JTAG Test Data Output</b> – used to serially shift test data and test instructions out of the Universe II</td></tr>
<tr><th>TMODE [2:0]</th><th>Input</th></tr>
<tr><td colspan="2"><b>Test Mode Enable</b> – used for chip testing, tie to ground for normal operation.</td></tr>
<tr><th>TMS</th><th>Input</th></tr>
<tr><td colspan="2"><b>JTAG Test Mode Select</b> – controls the state of the Test Access Port (TAP) controller in the Universe II. Tie to any logic level if JTAG is not used in the system.</td></tr>
<tr><th>TRDY#</th><th>Bidirectional</th></tr>
<tr><td colspan="2"><b>Target Ready</b> – used by the Universe II as PCI slave to indicate that it is ready to complete the current data phase. During a read with Universe II as PCI master, the slave asserts TRDY# to indicate to the Universe II that valid data is present on the data bus.</td></tr>
<tr><th>TRST#</th><th>Input</th></tr>
<tr><td colspan="2"><b>JTAG Test Reset</b> – provides asynchronous initialization of the TAP controller in the Universe II. Tie to ground if JTAG is not used in the system.</td></tr>
<tr><th>VCOCTL</th><th>Input</th></tr>
<tr><td colspan="2">Manufacturing testing, tie to ground for normal operation</td></tr>
<tr><th>VME_RESET#</th><th>Input</th></tr>
<tr><td colspan="2"><b>VMEbus Reset Input</b> — generates a VME bus system reset.</td></tr>
</table>

Chapter 8 Signals and DC Characteristics

Terminology

The I/O type abbreviations used in the pin list in Table 8-2 on page 8-3 are defined below. A numbered suffix indicates the current rating of the output (in mA).

Analog	Analog input signal
I	Input only
I/O	Input and output
O	Output only
OD	Open drain output
PD	Pulled-down internally
PU	Pulled-up internally
TP	Totem pole output
TTL	Input with TTL thresholds
TTL SCH	Schmitt trigger input with TTL thresholds
3S	Tri–state output

DC Characteristics and Pin Assignments

Table 8-1: DC Electrical Characteristics (VDD = 5 V ± 10%)

Symbol	Parameter	Signal Type	Test Conditions	Tested at 0°C to 70°C	
				Min	Max
VIH	Min. high–level input	CTTL	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	2.2 V	V_{DD} + 0.3V
		CMOS	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	0.7V_{DD}	V_{DD} + 0.3V
VIL	Max. low–level input	CTTL	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	–0.3 V	0.8V
		CMOS	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	–0.3 V	0.3V_{DD}
VT+	Positive going Schmitt trigger voltage	CTTL/SCH	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	–	2.4 V
		CMOS/SCH	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	–	0.7V_{DD}
VT–	Negative going Schmitt trigger voltage	CTTL/SCH	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	0.8 V	–
		CMOS/SCH	VOUT = 0.1V or V_{DD} – 0.1V; [IOUT] = 20 µA	0.25V_{DD}	–
VHysteresis	Schmitt trigger hysteresis voltage	CTTL/SCH	VT+ to VT–	0.05V_{DD}	–
		CMOS/SCH	VT+ to VT–	0.12V_{DD}	–
I_{IN}	Maximum input leakage current	CMOS and CTTL	With no pull–up resistor (V_{IN} = V_{SS} or V_{DD})	–5.0 µA	5.0 µA
I_{OZ}	Maximum output leakage current	3S	(V_{OUT} = V_{SS} or V_{DD})	–10.0 µA	10.0 µA
		OD	(V_{OUT} = V_{DD})	–10.0 µA	10.0 µA

Table 8-2: Pin List and DC Characteristics for Universe II Signals

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description[1]
ack64#	W11	W8	I/O	TTL	3S	6	–2	PCI Acknowledge 64 Bit Transfer
AD [63:0]	Table 8-3		I/O	TTL	3S	6	–2	PCI Address/Data Pins
C/BE# [0]	Y14	T11	I/O	TTL	3S	6	–2	PCI Command and Byte Enables
C/BE# [1]	V14	Y11						
C/BE# [2]	T14	U11						
C/BE# [3]	W13	W10						
C/BE# [4]	AE15	Y12						
C/BE# [5]	AD14	V11						
C/BE# [6]	T12	V10						
C/BE# [7]	AD12	Y9						
clk64	C23	D16	I	TTL	–	–	–	VME Clock 64 MHz—60-40 duty, 5 ns rise time
devsel#	AC7	V6	I/O	–	3S	6	–2	PCI Device Select
enid	AE21	Y16	I	CMOS	–	–	–	Enable IDD Tests
frame#	W17	T12	I/O	TTL	3S	6	–2	PCI Cycle Frame
gnt#	AE17	Y14	I	TTL	–	–	–	PCI Grant
idsel	AB16	Y15	I	TTL	–	–	–	PCI Initialization Device Select
lint# [0]	K20	H15	I/O	TTL	OD	12	–12	PCI Interrupt
lint# [1]	AA5	U3	I/O	TTL	OD	4	–4	
lint# [2]	L9	J3						
lint# [3]	V6	R4						
lint# [4]	M4	J4						
lint# [5]	L3	J6						
lint# [6]	M8	J5						
lint# [7]	L1	K4						
irdy#	AC15	V12	I/O	TTL	3S	6	–2	PCI Initiator Ready
lclk	AA3	W3	I	TTL	–	–	–	PCI Clock Signal
lock#	AA23	T18	I/O	TTL	3S	6	–2	PCI Lock
lrst#	R1	M1	O	–	3S	6	–2	PCI Reset Output
par	P8	L1	I/O	TTL	3S	6	–2	PCI parity
par64	AE5	W4	I/O	TTL	3S	6	–2	PCI Parity Upper DWORD
perr#	AB4	W5	I/O	TTL	3S	6	–2	PCI Parity Error
pll_testout	AB2	V1	For factory testing					
pll_testsel	AC1	T2	For factory testing					
pwrrst#	T4	R1	I	TTL/Schm	–	–	–	Power–up Reset
req#	K22	F20	O	–	3S	6	–2	PCI Request

Table 8-2: Pin List and DC Characteristics for Universe II Signals (continued)

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description[1]
req64#	AD18	T13	I/O	TTL	3S	6	–2	PCI Request 64 Bit Transfer
rst#	AA19	W17	I	TTL	–	–	–	PCI Reset
serr#	AA7	R7	O	TTL	OD	12	–12	PCI System Error
stop#	AB18	W15	I/O	TTL	3S	6	–2	PCI Stop
tck	H12	A10	I	TTL	–	–	–	JTAG Test Clock Input
tdi	A13	F10	I	TTL (PU)				JTAG Test Data Input
tdo	C13	E11	O	–	3S	–	–	JTAG Test Data OUTput
tmode [0]	AA13	W11	I	TTL	–	–	–	Test Mode Enable
tmode [1]	AA21	V17						
tmode [2]	W23	R18						
tms	C11	C9	I	TTL (PU)				JTAG Test Mode Select
trdy#	AD8	W7	I/O	TTL	3S	6	–2	PCI Target Ready
trst#	E13	B10	I	TTL (PU)				JTAG Test Reset
VA [31:1]	Table 8-4		I/O	TTL (PD)	3S	3	-3	VMEbus Address Pins
vam [0]	E11	D8	I/O	TTL	3S	3	–3	VMEbus Address Modifier Signals
vam [1]	D10	A6						
vam [2]	G9	E9						
vam [3]	B10	B8						
vam [4]	H10	D9						
vam [5]	A9	A7						
vam_dir	B8	E8	O	–	3S	6	–6	VMEbus AM Signal Direction Control
vas#	B14	A12	I/O	TTL/Schm (PU)	3S	3	-3	VMEbus Address Strobe
vas_dir	K12	D10	O	–	3S	6	–6	VMEbus AS Direction Control
va_dir	G13	B11	O	–	3S	12	–12	VMEbus Address Direction Control
vbclr#	N3	K5	O	–	3S	3	–3	VMEbus BCLR* Signal
vbgi# [0]	N21	K19	I	TT	–	–	–	VMEbus Bus Grant In
vbgi# [1]	M16	K17						
vbgi# [2]	N25	K15						
vbgi# [3]	N23	L16		TT (PD)				

Table 8-2: Pin List and DC Characteristics for Universe II Signals (continued)

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description[1]
vbgo# [0]	M20	K16	O	–	3S	12	–12	VMEbus Bus Grant Out
vbgo# [1]	L25	J20						
vbgo# [2]	M18	K20						
vbgo# [3]	M24	K18						
vcoctl	AE3	T5	I	–	–	–	–	Factory testing
VD [31:0]	Table 8-4		I/O	TTL	3S	3	–3	VMEbus Data Pins
vd_dir	F10	F8	O	–	3S	12	–12	VMEbus Data Direction Control
vds# [0]	F12	E10	I/O	TTL (PU)	3S	3	-3	VMEbus Data Strobes
vds# [1]	A11	A9						
vds_dir	J11	F9	O	–	3S	6	–6	VMEbus Data Strobe Direction Control
vdtack#	G15	B13	I/O	TTL/Schm (PU)	3S	3	-3	VMEbus DTACK* Signal
viack#	E7	E6	I/O	TTL	3S	3	–3	VMEbus IACK* Signal
viacki#	AE23	W16	I	TTL	–	–	–	VMEbus IACKIN* Signal
viacko#	L21	H17	O	–	3S	12	–12	VMEbus IACKOUT* Signal
vlword#	K14	C11	I/O	TTL (PD)	3S	3	-3	VMEbus LWORD* Signal
VME_RESET#	V22	T19	I	TTL				VMEbus Reset Input
voe#	B12	C10	O	–	3S	24	–24	VMEbus Transceiver Output Enable
vracfail#	P18	M16	I	TTL/Schm	–	–	–	VMEbus ACFAIL* Signal
vrbbsy#	M6	J2	I	TTL/Schm	–	–	–	VMEbus Received BBSY* Signal
vrberr#	A7	B7	I	TTL/Schm	–	–	–	VMEbus Receive Bus Error
vrbr# [0]	W5	U2	I	TTL/Schm	–	–	–	VMEbus Receive Bus Request
vrbr# [1]	U1	P1						
vrbr# [2]	R3	M3						
vrbr# [3]	L7	H2						
vrirq# [1]	H22	F19	I	TTL/Schm	–	–	–	VMEbus Receive Interrupts
vrirq# [2]	H20	F17						
vrirq# [3]	E25	E20						
vrirq# [4]	J21	G18						
vrirq# [5]	V16	U12						

Table 8-2: Pin List and DC Characteristics for Universe II Signals (continued)

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description[1]
vrirq# [6]	P20	M19						
vrirq# [7]	R17	M18						
vrsysfail#	AC13	T10						
vrsysrst#	C21	C16	I	TTL	–	–	–	VMEbus Receive SYSFAIL Signal
			I	TTL/Schm	–	–	–	VMEbus Receive SYSRESET* Signal
vscon_dir	M2	J1	O	–	3S	6	–6	SYSCON signals direction control
vslave_dir	C15	F12	O	–	3S	6	–6	DTACK/BERR direction control
vsysclk	N7	K2	I/O	TTL	3S	3	–3	VMEbus SYSCLK Signal
vwrite#	D8	B6	I/O	TTL	3S	3	–3	VMEbus Write
vxbbsy	P2	L3	O	–	3S	3	–3	VMEbus Transmit BBSY* Signal
vxberr	D12	B9	O	–	3S	3	–3	VMEbus Transmit Bus Error (BERR*)
vxbr [0]	G25	G19	O	–	3S	3	–3	VMEbus Transmit Bus Request
vxbr [1]	H24	H16						
vxbr [2]	P24	M20						
vxbr [3]	G23	C19						
vxirq [1]	J19	J16	O	–	3S	3	–3	VMEbus Transmit Interrupts
vxirq [2]	K24	H19						
vxirq [3]	K18	J17						
vxirq [4]	J25	G20						
vxirq [5]	L23	J18						
vxirq [6]	M22	J19						
vxirq [7]	R25	L17						
vxsysfail	M10	K3	O	–	3S	3	–3	VMEbus Transmit SYSFAIL Signal
vxsysrst	A23	E16	O	–	3S	3	–3	VMEbus Transmit SYSRESET* Signal

Note 1: All PCI pins meet PCI's AC current specifications.

Table 8-3: PCI Bus Address/Data Pins

Signal	PBGA	CBGA		Signal	PBGA	CBGA
ad [0]	P16	L18		ad [32]	L17	J15
ad [1]	P22	M17		ad [33]	N19	L19
ad [2]	R19	N19		ad [34]	R23	M15
ad [3]	T18	U20		ad [35]	U19	N18
ad [4]	T22	N15		ad [36]	U23	R20
ad [5]	T20	T20		ad [37]	W25	P16
ad [6]	AA25	U19		ad [38]	U21	P17
ad [7]	AB24	R17		ad [39]	V20	R19
ad [8]	AB22	R16		ad [40]	Y22	U18
ad [9]	AE25	T17		ad [41]	W21	P15
ad [10]	AC21	V19		ad [42]	AD22	Y18
ad [11]	AB20	V15		ad [43]	Y20	T15
ad [12]	AC19	W18		ad [44]	AD20	T14
ad [13]	AA17	V14		ad [45]	Y18	U15
ad [14]	AA15	U13		ad [46]	AE19	W14
ad [15]	U15	R12		ad [47]	AD16	W13
ad [16]	AE11	U10		ad [48]	V12	T9
ad [17]	AB12	U9		ad [49]	Y12	W9
ad [18]	W9	V8		ad [50]	AC11	R9
ad [19]	AD10	U8		ad [51]	V10	Y4
ad [20]	AE7	T7		ad [52]	AB10	R8
ad [21]	Y8	W6		ad [53]	AA9	U7
ad [22]	AD4	U6		ad [54]	AB8	T6
ad [23]	Y6	R5		ad [55]	AB6	V4
ad [24]	Y2	P5		ad [56]	Y4	R3
ad [25]	V4	R2		ad [57]	W3	V2
ad [26]	U5	P3		ad [58]	AA1	T1
ad [27]	W1	P2		ad [59]	V2	N5
ad [28]	U7	M5		ad [60]	R5	N4
ad [29]	T8	M4		ad [61]	T2	N2
ad [30]	P6	L5		ad [62]	R9	M6
ad [31]	P10	L4		ad [63]	N5	L2

Table 8-4: VMEbus Address Pins[a]

Signal	PBGA	CBGA		Signal	PBGA	CBGA
va [1]	H14	E12		va [17]	D18	B16
va [2]	A15	D11		va [18]	C19	C15
va [3]	F14	B12		va [19]	B20	D17
va [4]	J15	C12		va [20]	B22	D15
va [5]	D14	D12		va [21]	D20	C17
va [6]	G17	C13		va [22]	F20	E15
va [7]	H16	A17		va [23]	E19	F14
va [8]	B16	D13		va [24]	A25	C20
va [9]	C17	A15		va [25]	E23	B18
va [10]	D16	F13		va [26]	C25	E19
va [11]	A19	E14		va [27]	G21	F16
va [12]	B18	B14		va [28]	E21	D18
va [13]	F16	A16		va [29]	F22	F18
va [14]	E17	D14		va [30]	D24	D19
va [15]	A21	B17		va [31]	F24	G16
va [16]	F18	B15				

[a] *All VA pins have an internal pull-down.*

Table 8-5: VMEbus Data Pins[a]

Signal	PBGA	CBGA		Signal	PBGA	CBGA
vd [0]	J7	H3		vd [16]	F6	E2
vd [1]	K8	D1		vd [17]	G5	G6
vd [2]	K2	H4		vd [18]	B2	E5
vd [3]	J3	F1		vd [19]	C1	E3
vd [4]	K4	H6		vd [20]	E3	E4
vd [5]	G1	G5		vd [21]	C3	A3
vd [6]	H2	G2		vd [22]	A1	C2
vd [7]	K6	E1		vd [23]	A3	B5
vd [8]	J5	G4		vd [24]	C5	C4
vd [9]	E1	D2		vd [25]	D6	C6
vd [10]	H6	F2		vd [26]	B4	B4
vd [11]	H4	F5		vd [27]	B6	E7
vd [12]	G3	F3		vd [28]	C7	B3
vd [13]	F2	D4		vd [29]	F8	D6
vd [14]	D2	F4		vd [30]	A5	A5
vd [15]	F4	D3		vd [31]	E9	C7

[a] *VD[30:27] have internal pull-downs.*

Table 8-6: Pin Assignments for Power and Ground

V_{SS} Pins					V_{DD} Pins					
PBGA		**CBGA**			**PBGA**			**CBGA**		
AB14	N15	A14	R10		A17	h8	W15	A4	G17	U1
AC9	N17	A18	R11		AA11	H18	W19	A8	H1	U14
AC25	P4	B2	R13		AC3	J1	Y24	A11	H5	U16
AD6	P12	B19	R14		*AC5	J9		a13	H18	U17
*AE1	P14	C1	T16		AC17	J17		c3	H20	V3
AE13	R11	F7	U4		AC23	J23		C5	K1	*V5
J13	R13	F11	*U5		AD2	L5		C8	L20	V7
K10	R15	G1	V9		AD24	L19		C14	N1	V13
K16	T6	G15	V20		AE9	R7		C18	N3	V16
L11	T16	K6	W2		B24	R21		D5	N16	V18
L13	T24	L6	W12		C9	T10		D7	N20	Y8
L15	U11	L15	W19		D4	U3		D20	P4	Y10
M12	U13	M2	Y3		D22	U9		E13	P18	Y13
M14	V24	N6	Y5		E5	U17		E17	R6	Y17
N1	Y10	N17	Y6		E15	U25		E18	R15	
N9	Y16	P6	Y7		G7	V8		F6	T3	
N11		P19			G11	V18		F15	T4	
N13		P20			g19	W7		G3	T8	

**AVDD and AVSS are power pins specifically used for powering the analog circuitry in the Universe II. Extra care should be taken to avoid noise and ground shifting on these pins through the use of decoupling capacitors or isolated ground and power planes.*

Table 8-7: Pinout for 313-pin Plastic BGA Package

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	T	U	V	W	Y	AA	AB	AC	AD	AE
1	VD[22]		VD[19]		VD[9]		VD[5]		VDD		INT#[7]		VSS		LRST#		VRBR#[1]		AD[27]		AD[58]		PLL_ TESTSEL		**AVSS**
2		VD[18]		VD[14]		VD[13]		VD[6]		VD[2]		VSCON_ DIR		VXBBSY		AD[61]		AD[59]		AD[24]		PLL_ TESTOUT		VDD	
3	VD[23]		VD[21]		VD[20]		VD[12]		VD[3]		INT#[5]		VBCLR#		VRBR#[2]		VDD		AD[57]		LCLK		VDD		VCOCTL
4		VD[26]		VDD		VD[15]		VD[11]		VD[4]		INT#[4]		VSS		PWRRST#		AD[25]		AD[56]		PERR#		AD[22]	
5	VD[30]		VD[24]		VDD		VD[17]		VD[8]		VDD		AD[63]		AD[60]		AD[26]		VRBR#[0]		INT#[1]		**AVDD**		PAR64
6		VD[27]		VD[25]		VD[16]		VD[10]		VD[7]		VRBBSY#		AD[30]		VSS		INT#[3]		AD[23]		AD[55]		VSS	
7	VRBERR#		VD[28]		VIACK#		**VDD**		VD[0]		VRBR#[3]		VSYSCLK		VDD		AD[28]		**VDD**		SERR#		DEVSEL#		AD[20]
8		VAM_DIR		VWRITE#		VD[29]		**VDD**		VD[1]		INT#[6]		PAR		AD[29]		**VDD**		AD[21]		AD[54]		TRDY#	
9	VAM[5]		VDD		VD[31]		VAM[2]		**VDD**		INT#[2]		**VSS**		AD[62]		**VDD**		AD[18]		AD[53]		VSS		VDD
10		VAM[3]		VAM[1]		VD_DIR		VAM[4]		**VSS**		VXSYSFAIL		AD[31]		**VDD**		AD[51]		VSS		AD[52]		AD[19]	
11	VDS#[1]		TMS		VAM[0]		VDD		VDS_DIR		**VSS**		**VSS**		**VSS**		VSS		ACK64#		VDD		AD[50]		AD[16]
12		VOE#		VXBERR		VDS#[0]		TCK		VAS_DIR		**VSS**		**VSS**		CBE[6]		AD[48]		AD[49]		AD[17]		CBE[7]	
13	TDI		TDO		TRST#		VA_DIR		**VSS**		**VSS**		**VSS**		**VSS**		**VSS**		CBE[3]		TMODE[0]		VRSYS- FAIL#		VSS
14		VAS#		VA[5]		VA[3]		VA[1]		VLWORD#		**VSS**		**VSS**		CBE[2]		CBE[1]		CBE[0]		VSS		CBE[5]	
15	VA[2]		VSLAVE_DIR		VDD		VDTACK#		VA[4]		**VSS**		**VSS**		**VSS**		AD[15]		VDD		AD[14]		IRDY#		CBE[4]
16		VA[8]		VA[10]		VA[13]		VA[7]		**VSS**		VBGI#[1]		AD[0]		**VSS**		VRIRQ#[5]		VSS		IDSEL		AD[47]	
17	VDD		VA[9]		VA[14]		VA[6]		**VDD**		AD[32]		**VSS**		VRIRQ#[7]		**VDD**		FRAME#		AD[13]		VDD		GNT#
18		VA[12]		VA[17]		VA[16]		**VDD**		VXIRQ[3]		VBGO#[2]		VRACFAIL#		AD[3]		**VDD**		AD[45]		STOP#		REQ64#	
19	VA[11]		VA[18]		VA[23]		**VDD**		VXIRQ[1]		VDD		AD[33]		AD[2]		AD[35]		**VDD**		RST#		AD[12]		AD[46]
20		VA[19]		VA[21]		VA[22]		VRIRQ#[2]		INT#[0]		VBGO#[0]		VRIRQ#[6]		AD[5]		AD[39]		AD[43]		AD[11]		AD[44]	
21	VA[15]		VRSYSRST#		VA[28]		VA[27]		VRIRQ#[4]		VIACKO#		VBGI#[0]		VDD		AD[38]		AD[41]		TMODE[1]		AD[10]		ENID
22		VA[20]		VDD		VA[29]		VRIRQ#[1]		REQ#		VXIRQ[6]		AD[1]		AD[4]		VME_RST#		AD[40]		AD[8]		AD[42]	
23	VXSYSRST		CLK64		VA[25]		VXBR[3]		VDD		VXIRQ[5]		VBGI#[3]		AD[34]		AD[36]		TMODE[2]		LOCK#		VDD		VIACKI#
24		VDD		VA[30]		VA[31]		VXBR[1]		VXIRQ[2]		VBGO#[3]		VXBR[2]		VSS		VSS		VDD		AD[7]		VDD	
25	VA[24]		VA[26]		VRIRQ#[3]		VXBR[0]		VXIRQ[4]		VBGO#[1]		VBGI#[2]		VXIRQ[7]		VDD		AD[37]		AD[6]		VSS		AD[9]

Table 8-8: Pinout for 324–pin Ceramic BGA Package

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	T	U	V	W	Y
1			VSS	VD[1]	VD[7]	VD[3]	VSS	VDD	VSCON_DIR	VDD	PAR	LRST#	VDD	VRBR#[1]	PWRRST#	AD[58]	VDD	PLL_TESTOUT		
2		VSS	VD[22]	VD[9]	VD[16]	VD[10]	VD[6]	VRBR#[3]	VRBBSY#	VSYSCLK	AD[63]	VSS	AD[61]	AD[27]	AD[25]	PLL_TESTSEL	VRBR#[0]	AD[57]	VSS	
3	VD[21]	VD[28]	VDD	VD[15]	VD[19]	VD[12]	VDD	VD[0]	INT#[2]	VXSYSFAIL	VXBBSY	VRBR#[2]	VDD	AD[26]	AD[56]	VDD	INT#[1]	VDD	LCLK	VSS
4	VDD	VD[26]	VD[24]	VD[13]	VD[20]	VD[14]	VD[8]	VD[2]	INT#[4]	INT#[7]	AD[31]	AD[29]	AD[60]	VDD	INT#[3]	VDD	VSS	AD[55]	PAR64	AD[51]
5	VD[30]	VD[23]	VDD	VDD	VD[18]	VD[11]	VD[5]	VDD	INT#[6]	VBCLR#	AD[30]	AD[28]	AD[59]	AD[24]	AD[23]	VCOCTL	AVSS	AVDD	PERR#	VSS
6	VAM[1]	VWRITE#	VD[25]	VD[29]	VIACK#	VDD	VD[17]	VD[4]	INT#[5]	VSS	VSS	AD[62]	VSS	VSS	VDD	AD[54]	AD[22]	DEVSEL#	AD[21]	VSS
7	VAM[5]	VRBERR#	VD[31]	VDD	VD[27]	VSS									SERR#	AD[20]	AD[53]	VDD	TRDY#	VSS
8	VDD	VAM[3]	VDD	VAM[0]	VAM_DIR	VD_DIR									AD[52]	VDD	AD[19]	AD[18]	ACK64#	VDD
9	VDS#[1]	VXBERR	TMS	VAM[4]	VAM[2]	VDS_DIR									AD[50]	AD[48]	AD[17]	VSS	AD[49]	CBE[7]
10	TCK	TRST#	VOE#	VAS_DIR	VDS#[0]	TDI									VSS	VRSYSFAIL#	AD[16]	CBE[6]	CBE[3]	VDD
11	VDD	VA_DIR	VLWORD#	VA[2]	TDO	VSS									VSS	CBE[0]	CBE[2]	CBE[5]	TMODE[0]	CBE[1]
12	VAS#	VA[3]	VA[4]	VA[5]	VA[1]	VSLAVE_DIR									AD[15]	FRAME#	VRIRQ#[5]	IRDY#	VSS	CBE[4]
13	VDD	VDTACK#	VA[6]	VA[8]	VDD	VA[10]									VSS	REQ64#	AD[14]	VDD	AD[47]	VDD
14	VSS	VA[12]	VDD	VA[14]	VA[11]	VA[23]									VSS	AD[44]	VDD	AD[13]	AD[46]	GNT#
15	VA[9]	VA[16]	VA[18]	VA[20]	VA[22]	VDD	VSS	INT#[0]	AD[32]	VBGI#[2]	VSS	AD[34]	AD[4]	AD[41]	VDD	AD[43]	AD[45]	AD[11]	STOP#	IDSEL
16	VA[13]	VA[17]	VRSYSRST#	CLK64	VXSYSRST	VA[27]	VA[31]	VXBR[1]	VXIRQ[1]	VBGO#[0]	VBGI#[3]	VRACFAIL#	VDD	AD[37]	AD[8]	VSS	VDD	VDD	VIACKI#	ENID
17	VA[7]	VA[15]	VA[21]	VA[19]	VDD	VRIRQ#[2]	VDD	VIACKO#	VXIRQ[3]	VBGI#[1]	VXIRQ[7]	AD[1]	VSS	AD[38]	AD[7]	AD[9]	VDD	TMODE[1]	RST#	VDD
18	VSS	VA[25]	VDD	VA[28]	VDD	VA[29]	VRIRQ#[4]	VDD	VXIRQ[5]	VBGO#[3]	AD[0]	VRIRQ#[7]	AD[35]	VDD	TMODE[2]	LOCK#	AD[40]	VDD	AD[12]	AD[42]
19		VSS	VXBR[3]	VA[30]	VA[26]	VRIRQ#[1]	VXBR[0]	VXIRQ[2]	VXIRQ[6]	VBGI#[0]	AD[33]	VRIRQ#[6]	AD[2]	VSS	AD[39]	VME_RESET#	AD[6]	AD[10]	VSS	
20			VA[24]	VDD	VRIRQ#[3]	REQ#	VXIRQ[4]	VDD	VBGO#[1]	VBGO#[2]	VDD	VXBR[2]	VDD	VSS	AD[36]	AD[5]	AD[3]	VSS		

Table 8-7 and Table 8-8 map pin numbers to signal names. These tables should not be read as figures. For layout purposes, please see Appendix D Typical Applications.

Appendix A System Registers

Introduction

The System Registers provide control/status not contained within the Universe II chip such as master/slave endian conversion and non-slot 1 bus time-out timer. Table A-1 defines the System Register Map.

The Universe II Control and Status Registers (USCRs) facilitate host system configuration and allow the user to control Universe II operational characteristics. The UCSRs are divided into four groups:

* the DMA Channel

* PCI Configuration Space (PCICS)

* VMEbus Control and Status Registers (VCSR)

* Universe II Device Specific Status Registers (UDSR)

The Universe II registers are little-endian. The register map for the UCSR's is included in Appendix B.

Table A-1: System Register Map

Register Name	**Mnemonic**	**Access**	**Mem Address**													
Command	COMM	B,W: R/W	0xD800E	**D15**	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**
				X	X	X	X	VME_EN	BYPASS		WTDSYS	BERRST	BERRI	BTOV1	BTOV0	BT
VME BERR Address Modifieer Regisgers	VBAR	B,W,L: R/WC	0xD8010	**D31**	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**	**D23**	**D22**	**D21**	**D20**	**D1**
				A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A1
				D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**
				A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3
VME BERR Address Modifier Rebisters	VBAMR	W,B: R/W	0xD8014	**D15**	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**
				X	X	X	X	X	X	X	X	X	X	AM5	AM4	AM
Board ID Register	ID	B,W: R	0xD8016	**D15**	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**
				0	1	1	1	0	1	0	1	1	0	0	1	0

The Table A-1 System Register Map Bit Definitions are defined as follows:

COMM (16-bit Read/Write register):

- MEC: Master Endian Conversion enable bit (1=enabled, 0=disabled)
- SEC: Slave Endian Conversion enable bit (1=enabled, 0=disabled)
- ABLE: Auxiliary BERR logic enable bit (1=enabled, 0=disabled
- BTO: Auxiliary Bus Time-out Timer enable bit (1=enabled, 0=disabled)
- BTOV(1:0): Auxiliary Bus Time-out Timer value: (00 = 16 ms, 01 = 64 ms,
- 10 = 256 ms, 11 = 1 ms)
- BERRI: BERR Interrupt enable bit (1=enabled, 0=disabled)
- BERRST: BERR Status Read/Clear
- WTDSYS: Watchdog Time-out to VME Sysfail (1=enabled, 0=disabled)
- BYPASS: VME Endian Conversion bypass Enable (1=enabled, 0=disabled)
- VME_EN: VMEbus enable (1=enabled, 0=disabled)

VBAR (32-bit Read register):

- A(31:1): Address of BERR cycle

VBAMR (16 bit Read register):

- AM(5:0): AM code of BERR cycle

BTR (16-bit Read-only register):

- BCD value representing the board number 7698

Appendix B Universe II Registers

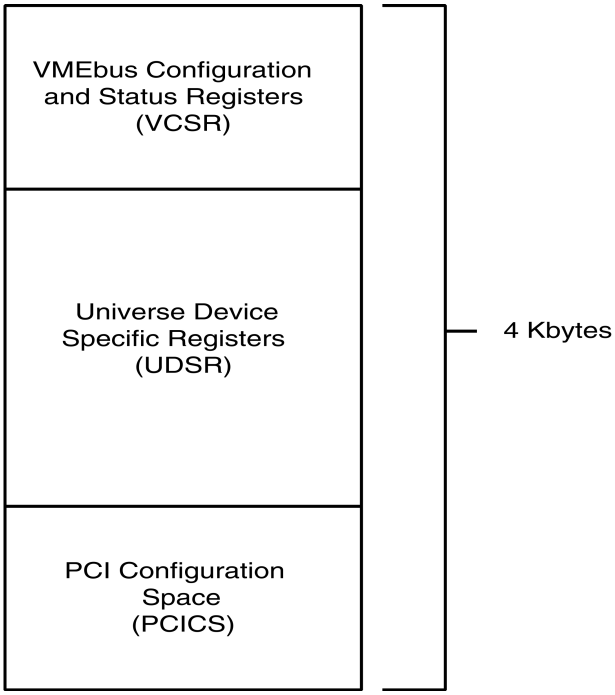
Introduction

The Universe II Control and Status Registers facilitate host system configuration and allow the user to control Universe II operational characteristics. The registers are divided into three groups:

- PCI Configuration Space,
- VMEbus Configuration and Status Registers, and
- Universe II Device Specific Status Registers.

The Universe II registers have little-endian byte-ordering. Figure B-1 below summarizes the supported register access mechanisms.

Figure B-1: UCSR Access Mechanisms


VMEbus Configuration and Status Registers (VCSR)
Universe Device Specific Registers (UDSR)
PCI Configuration Space (PCICS)
4 Kbytes

The bit combinations listed as “Reserved” must not be programmed. All bits listed as “Reserved” must read back a value of zero.

Table B-1 below lists the Universe II registers by address offset. The tables following the register map (Table B-2 to Table B-252) provide detailed descriptions of each register.

Table B-1: Universe II Register Map

Offset	Register	Name
000	PCI Configuration Space ID Register	PCI_ID
004	PCI Configuration Space Control and Status Register	PCI_CSR
008	PCI Configuration Class Register	PCI_CLASS
00C	PCI Configuration Miscellaneous 0 Register	PCI_MISC0
010	PCI Configuration Base Address Register	PCI_BS0
014	PCI Configuration Base Address 1 Register	PCI_BS1
018-024	PCI Unimplemented	
028	PCI Reserved	
02C	PCI Reserved	
030	PCI Unimplemented	
034	PCI Reserved	
038	PCI Reserved	
03C	PCI Configuration Miscellaneous 1 Register	PCI_MISC1
040-0FF	PCI Unimplemented	
100	PCI Target Image 0 Control Register	LSI0_CTL
104	PCI Target Image 0 Base Address Register	LSI0_BS
108	PCI Target Image 0 Bound Address Register	LSI0_BD
10C	PCI Target Image 0 Translation Offset Register	LSI0_TO
110	Reserved	
114	PCI Target Image 1 Control Register	LSI1_CTL
118	PCI Target Image 1 Base Address Register	LSI1_BS
11C	PCI Target Image 1 Bound Address Register	LSI1_BD
120	PCI Target Image 1 Translation Offset Register	LSI1_TO
124	Reserved	
128	PCI Target Image 2 Control Register	LSI2_CTL
12C	PCI Target Image 2 Base Address Register	LSI2_BS
130	PCI Target Image 2 Bound Address Register	LSI2_BD
134	PCI Target Image 2 Translation Offset Register	LSI2_TO
138	Reserved	
13C	PCI Target Image 3 Control Register	LSI3_CTL
140	PCI Target Image 3 Base Address Register	LSI3_BS
144	PCI Target Image 3 Bound Address Register	LSI3_BD
148	PCI Target Image 3 Translation Offset Register	LSI3_TO
14C-16C	Reserved	
170	Special Cycle Control Register	SCYC_CTL
174	Special Cycle PCI Bus Address Register	SCYC_ADDR
178	Special Cycle Swap/Compare Enable Register	SCYC_EN

Table B-1: Universe II Register Map (continued)

Offset	Register	Name
17C	Special Cycle Compare Data Register	SCYC_CMP
180	Special Cycle Swap Data Register	SCYC_SWP
184	PCI Miscellaneous Register	LMISC
188	Special PCI Target Image Register	SLSI
18C	PCI Command Error Log Register	L_CMDERR
190	PCI Address Error Log Register	LAERR
194-19C	Reserved	
1A0	PCI Target Image 4 Control Register	LSI4_CTL
1A4	PCI Target Image 4 Base Address Register	LSI4_BS
1A8	PCI Target Image 4 Bound Address Register	LSI4_BD
1AC	PCI Target Image 4 Translation Offset Register	LSI4_TO
1B0	Reserved	
1B4	PCI Target Image 5 Control Register	LSI5_CTL
1B8	PCI Target Image 5 Base Address Register	LSI5_BS
1BC	PCI Target Image 5 Bound Address Register	LSI5_BD
1C0	PCI Slave Image 5 Translation Offset Register	LSI5_TO
1C4	Reserved	
1C8	PCI Target Image 6 Control Register	LSI6_CTL
1CC	PCI Target Image 6 Base Address Register	LSI6_BS
1D0	PCI Target Image 6 Bound Address Register	LSI6_BD
1D4	PCI Target Image 6 Translation Offset Register	LSI6_TO
1D8	Reserved	
1DC	PCI Target Image 7 Control Register	LSI7_CTL
1E0	PCI Target Image 7 Base Address Register	LSI7_BS
1E4	PCI Target Image 7 Bound Address Register	LSI7_BD
1E8	PCI Target Image 7 Translation Offset Register	LSI7_TO
1EC-1FC	Reserved	
200	DMA Transfer Control Register	DCTL
204	DMA Transfer Byte Count Register	DTBC
208	DMA PCI Bus Address Register	DLA
20C	Reserved	
210	DMA VMEbus Address Register	DVA
214	Reserved	
218	DMA Command Packet Pointer Register	DCPP
21C	Reserved	
220	DMA General Control and Status Register	DGCS
224	DMA Linked List Update Enable Register	D_LLUE
228-2FC	Reserved	
300	PCI Interrupt Enable Register	LINT_EN
304	PCI Interrupt Status Register	LINT_STAT
308	PCI Interrupt Map 0 Register	LINT_MAP0
30C	PCI Interrupt Map 1 Register	LINT_MAP1
310	VMEbus Interrupt Enable Register	VINT_EN

Table B-1: Universe II Register Map (continued)

Offset	Register	Name
314	VMEbus Interrupt Status Register	VINT_STAT
318	VMEbus Interrupt Map 0 Register	VINT_MAP0
31C	VMEbus Interrupt Map 1 Register	VINT_MAP1
320	Interrupt Status/ID Out Register	STATID
324	VIRQ1 STATUS/ID Register	V1_STATID
328	VIRQ2 STATUS/ID Register	V2_STATID
32C	VIRQ3 STATUS/ID Register	V3_STATID
330	VIRQ4 STATUS/ID Register	V4_STATID
334	VIRQ5 STATUS/ID Register	V5_STATID
338	VIRQ6 STATUS/ID Register	V6_STATID
33C	VIRQ7 STATUS/ID Register	V7_STATID
340	PCI Interrupt Map 2 Register	LINT_MAP2
344	VME Interrupt Map 1 Register	VINT_MAP2
348	Mailbox 0 Register	MBOX0
34C	Mailbox 1 Register	MBOX1
350	Mailbox 2 Register	MBOX2
354	Mailbox 3 Register	MBOX3
358	Semaphore 0 Register	SEMA0
35C	Semaphore 1 Register	SEMA1
360-3FC	Reserved	
400	Master Control Register	MAST_CTL
404	Miscellaneous Control Register	MISC_CTL
408	Miscellaneous Status Register	MISC_STAT
40C	User AM Codes Register	USER_AM
410-EFC	Reserved	
F00	VMEbus Slave Image 0 Control Register	VSI0_CTL
F04	VMEbus Slave Image 0 Base Address Register	VSI0_BS
F08	VMEbus Slave Image 0 Bound Address Register	VSI0_BD
F0C	VMEbus Slave Image 0 Translation Offset Register	VSI0_TO
F10	Reserved	
F14	VMEbus Slave Image 1 Control Register	VSI1_CTL
F18	VMEbus Slave Image 1 Base Address Register	VSI1_BS
F1C	VMEbus Slave Image 1 Bound Address Register	VSI1_BD
F20	VMEbus Slave Image 1 Translation Offset Register	VSI1_TO
F24	Reserved	
F28	VMEbus Slave Image 2 Control Register	VSI2_CTL
F2C	VMEbus Slave Image 2 Base Address Register	VSI2_BS
F30	VMEbus Slave Image 2 Bound Address Register	VSI2_BD
F34	VMEbus Slave Image 2 Translation Offset Register	VSI2_TO
F38	Reserved	
F3C	VMEbus Slave Image 3 Control Register	VSI3_CTL
F40	VMEbus Slave Image 3 Base Address Register	VSI3_BS
F44	VMEbus Slave Image 3 Bound Address Register	VSI3_BD

Table B-1: Universe II Register Map (continued)

Offset	Register	Name
F48	VMEbus Slave Image 3 Translation Offset Register	VSI3_TO
F4C-F60	Reserved	
F64	Location Monitor Control Register	LM_CTL
F68	Location Monitor Base Address Register	LM_BS
F6C	Reserved	
F70	VMEbus Register Access Image Control Register	VRAI_CTL
F74	VMEbus Register Access Image Base Address Register	VRAI_BS
F78	Reserved	
F7C	Reserved	
F80	VMEbus CSR Control Register	VCSR_CTL
F84	VMEbus CSR Translation Offset Register	VCSR_TO
F88	VMEbus AM Code Error Log Register	V_AMERR
F8C	VMEbus Address Error Log Register	VAERR
F90	VMEbus Slave Image 4 Control Register	VSI4_CTL
F94	VMEbus Slave Image 4 Base Address Register	VSI4_BS
F98	VMEbus Slave Image 4 Bound Address Register	VSI4_BD
F9C	VMEbus Slave Image 4 Translation Offset Register	VSI4_TO
FA0	Reserved	
FA4	VMEbus Slave Image 5 Control Register	VSI5_CTL
FA8	VMEbus Slave Image 5 Base Address Register	VSI5_BS
FAC	VMEbus Slave Image 5 Bound Address Register	VSI5_BD
FB0	VMEbus Slave Image 5 Translation Offset Register	VSI5_TO
FB4	Reserved	
FB8	VMEbus Slave Image 6 Control Register	VSI6_CTL
FBC	VMEbus Slave Image 6 Base Address Register	VSI6_BS
FC0	VMEbus Slave Image 6 Bound Address Register	VSI6_BD
FC4	VMEbus Slave Image 6 Translation Offset Register	VSI6_TO
FC8	Reserved	
FCC	VMEbus Slave Image 7 Control Register	VSI7_CTL
FD0	VMEbus Slave Image 7 Base Address Register	VSI7_BS
FD4	VMEbus Slave Image 7 Bound Address Register	VSI7_BD
FD8	VMEbus Slave Image 7 Translation Offset Register	VSI7_TO
FDC-FEC	Reserved	
FF0	VME CR/CSR Reserved	
FF4	VMEbus CSR Bit Clear Register	VCSR_CLR
FF8	VMEbus CSR Bit Set Register	VCSR_SET
FFC	VMEbus CSR Base Address Register	VCSR_BS

Table B-2: PCI Configuration Space ID Register (PCI_ID)

Register Name: PCI_ID	Offset:000
Bits	**Function**
31-24	DID
23-16	DID
15-08	VID
07-00	VID

Table B-3: PCI_ID Description

Name	Type	Reset By	Reset State	Function
DID[15:0]	R	All	0	Device ID - Tundra allocated device identifier
VID[15:0]	R	All	10E3	Vendor ID - PCI SIG allocated vendor identifier

Table B-4: PCI Configuration Space Control and Status Register (PCI_CSR)

Register Name: PCI_CSR								Offset:004
Bits	**Function**							
31-24	D_PE	S_SERR	R_MA	R_TA	S_TA	DEVSEL		DP_D
23-16	TFBBC	PCI Reserved						
15-08	PCI Reserved						MFBBC	SERR_EN
07-00	WAIT	PERESP	VGAPS	MWI_EN	SC	BM	MS	IOS

Table B-5: PCI_CSR Description

Name	Type	Reset By	Reset State	Function
D_PE	R/Write 1 to Clear	All	0	Detected Parity Error 0=No parity error, 1=Parity error This bit is always set by the Universe II when the PCI master interface detects a data parity error or the PCI target interface detects address or data parity errors.
S_SERR	R/Write 1 to Clear	All	0	Signalled SERR# 0=SERR# not asserted, 1=SERR# asserted. The Universe II PCI target interface sets this bit when it asserts SERR# to signal an address parity error. SERR_EN must be set before SERR# can be asserted.

Table B-5: PCI_CSR Description (continued)

Name	Type	Reset By	Reset State	Function
R_MA	R/Write 1 to Clear	All	0	Received Master-Abort 0=Master did not generate Master-Abort, 1=Master generated Master-Abort The Universe II PCI master interface sets this bit when a transaction it initiated had to be terminated with a Master-Abort.
R_TA	R/Write 1 to Clear	All	0	Received Target-Abort 0=Master did not detect Target-Abort, 1=Master detected Target-Abort. The Universe II PCI master interface sets this bit when a transaction it initiated was terminated with a Target-Abort.
S_TA	R/Write 1 to Clear	All	0	Signalled Target-Abort 0=Target did not terminate transaction with Target-Abort,1=Target terminated transaction with Target-Abort.
DEVSEL	R	All	01	Device Select Timing The Universe II is a medium speed device
DP_D	R/Write 1 to Clear	All		Data Parity Detected 0=Master did not detect/generate data parity error, 1=Master detected/generated data parity error. The Universe II PCI master interface sets this bit if the Parity Error Response bit is set, it is the master of transaction in which it asserts PERR#, or the addressed target asserts PERR#.
TFBBC	R	All	0	Target Fast Back to Back Capable Universe II cannot accept Back to Back cycles from a different agent.
MFBBC	R	All	0	Master Fast Back to Back Enable 0=no fast back-to-back transactions The Universe II master never generates fast back to back transactions.
SERR_EN	R/W	All	0	SERR# Enable 0=Disable SERR# driver, 1=Enable SERR# driver. Setting this and PERESP allows the Universe II PCI target interface to report address parity errors with SERR#.
WAIT	R	All	0	Wait Cycle Control 0=No address/data stepping

Table B-5: PCI_CSR Description (continued)

Name	Type	Reset By	Reset State	Function
PERESP	R/W	All	0	Parity Error Response 0=Disable, 1=Enable Controls the Universe II response to data and address parity errors. When enabled, it allows the assertion of PERR# to report data parity errors. When this bit and SERR_EN are asserted, the Universe II can report address parity errors on SERR#. Universe II parity generation is unaffected by this bit.
VGAPS	R	All	0	VGA Palette Snoop 0=Disable The Universe II treats palette accesses like all other accesses.
MWI_EN	R	All	0	Memory Write and Invalidate Enable 0=Disable The Universe II PCI master interface never generates a Memory Write and Invalidate command.
SC	R	All	0	Special Cycles 0=Disable The Universe II PCI target interface never responds to special cycles.
BM	R/W	PWR VME	see note 1	Master Enable 0=Disable, 1=Enable For a VMEbus slave image to respond to an incoming cycle, this bit must be set. If this bit is cleared while there is data in the VMEbus Slave Posted Write FIFO, the data will be written to the PCI bus but no further data will be accepted into this FIFO until the bit is set.
MS	R/W	PWR VME	see note 1	Target Memory Enable 0=Disable, 1=Enable-
IOS	R/W	PWR VME	see note 1	Target IO Enable 0=Disable, 1=Enable

1.If the VCSR or LSI0 power-up options are enabled, these bits are not disabled after reset.

2.The Universe II only rejects PCI addresses with parity errors in the event that both the PERESP and SERR_EN bits are programmed to a value of 1.

Table B-6: PCI Configuration Class Register (PCI_CLASS)

Register Name: PCI_CLASS	Offset:008
Bits	**Function**
31-24	BASE
23-16	SUB
15-08	PROG
07-00	RID

Table B-7: PCI_CLASS Description

Name	Type	Reset By	Reset State	Function
BASE [7:0]	R	All	06	Base Class Code The Universe II is defined as a PCI bridge device
SUB [7:0]	R	All	80	Sub Class Code The Universe II sub-class is “other bridge device”
PROG [7:0]	R	All	00	Programming Interface The Universe II does not have a standardized register-level programming interface
RID [7:0]	R	All	01	Revision ID

Table B-8: PCI Configuration Miscellaneous 0 Register (PCI_MISC0)

Register Name: PCI_MISC0				**Offset:00C**
Bits	**Function**			
31-24	BISTC	SBIST	PCI Reserved	CCODE
23-16	MFUNCT	LAYOUT		
15-08	LTIMER	0	0	0
07-00	PCI Unimplemented			

Table B-9: PCI_MISC0 Description

Name	Type	Reset By	Reset State	Function
BISTC	R	All	0	The Universe II is not BIST Capable
SBIST	R	All	0	Start BIST The Universe II is not BIST capable
CCODE	R	All	0	Completion Code The Universe II is not BIST capable
MFUNCT	R	All	0	Multifunction Device 0=No, 1=Yes The Universe II is not a multi-function device.
LAYOUT	R	All	0	Configuration Space Layout
LTIMER [7:3]	R/W	All	0	Latency Timer: The latency timer has a resolution of 8 clocks

The Universe II is not a multi-function device.

Table B-10: PCI Configuration Base Address Register (PCI_BS0)

Register Name: PCI_BS0								**Offset:010**
Bits	**Function**							
31-24	BS							
23-16	BS							
15-08	BS				0	0	0	0
07-00	0	0	0	0	0	0	0	SPACE

Table B-11: PCI_BS0 Description

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	All	0	Base Address
SPACE	R	All	Power-up Option	PCI Bus Address Space 0=Memory, 1=I/O

This register specifies the 4 Kbyte aligned base address of the 4 Kbyte Universe II register space on PCI.

A power-up option determines if the registers are mapped into Memory or I/O space in relation to this base address. (See Power-Up Options on page 160). If mapped into Memory space, the user is free to locate the registers anywhere in the 32-bit address space.

- When the VA[1] pin is sampled low at power-up, the PCI_BS0 register’s SPACE bit is set to “1”, which signifies I/O space, and the PCI_BS1 register’s SPACE bit is set to “0”, which signifies memory space.
- When VA[1] is sampled high at power-up, the PCI_BS0 register’s SPACE register’s bit is set to “0”, which signifies Memory space, and the PCI_BS1 register’s SPACE bit is set to “1”, which signifies I/O space.

A write must occur to this register before the Universe II Device Specific Registers can be accessed. This write can be performed with a PCI configuration transaction or a VMEbus register access.

Table B-12: PCI Configuration Base Address 1 Register (PCI_BS1)

Register Name:								PCI_BS1Offset:014
Bits	Function							
31-24	BS							
23-16	BS							
15-08	BS				0	0	0	0
07-00	0	0	0	0	0	0	0	SPACE

Tale B-13: PCI_BS1 Description

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	All	0	Base Address
SPACE	R	All	Power-up Option	PCI Bus Address Space 0=Memory, 1=I/O

This register specifies the 4 KByte aligned base address of the 4 KByte Universe II register space in PCI.

A power-up option determines the value of the SPACE bit. This determines whether the registers are mapped into Memory or I/O space in relation to this base address. (See Power-Up Options on page 160). If mapped into Memory space, the user is free to locate the Universe registers anywhere in the 32-bit address space. If PCI_BS0 is mapped to Memory space, PCI_BS1 is mapped to I/O space; if PCI_BS0 is mapped to I/O space, then PCI_BS1 is mapped to Memory space.

- When the VA[1] pin is sampled low at power-up, the PCI_BS0 register’s SPACE bit is set to “1”, which signifies I/O space, and the PCI_BS1 register’s SPACE bit is set to “0”, which signifies memory space.
- When VA[1] is sampled high at power-up, the PCI_BS0 register’s SPACE register’s bit is set to “0”, which signifies Memory space, and the PCI_BS1 register’s SPACE bit is set to “1”, which signifies I/O space.

A write must occur to this register before the Universe II Device Specific Registers can be accessed. This write can be performed with a PCI configuration transaction or a VMEbus register access.

The SPACE bit in this register is an inversion of the SPACE field in PCI_BS0.

Table B-14: PCI Configuration Miscellaneous 1 Register (PCI_MISC1)

Register Name:	**PCI_MISC1** **Offset:03C**
Bits	**Function**
31-24	MAX_LAT [7:0}
23-16	MIN_GNT [7:0}
15-08	INT_PIN [7:0}
07-00	INT_LINE [7:0}

Table B-15: PCI_MISC1 Description

Name	Type	Reset By	Reset State	Function
MAX_LAT[7:0]	R	All	0	Maximum Latency: This device has no special latency requirements
MIN_GNT[7:0}	R	All	00000011	Minimum Grant:.250 ns units
INT_PIN[7:0]	R	All	00000001	Interrupt Pin: Universe II pin INT# [0] has a PCI compliant I/O buffer
INT_LINE[7:0]	R/W	All	0	Interrupt Line: used by some PCI systems to record interrupt routing information

The MIN_GNT parameter assumes the Universe II master is transferring an aligned burst size of 64 bytes to a 32-bit target with no wait states. This would require roughly 20 clocks (at a clock frequency of 33 MHz, this is about 600 ns). MIN_GNT is set to three, or 750 ns.

Table B-16: PCI Target Image 0 Control (LSI0_CTL)

<table>
<tr><th colspan="7">Register Name: LSI0_CTL</th><th>Offset:100</th></tr>
<tr><th>Bits</th><th colspan="7">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="5">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="3">Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>PGM</td><td>Reserved</td><td>SUPER</td><td colspan="2">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td colspan="6">Reserved</td><td>LAS</td></tr>
</table>

Table B-17: LSI0_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	Power-up Option	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	Power-up Option	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=No BLTs on VMEbus, 1=Single BLTs on VMEbus
LAS	R/W	All	Power-up Option	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-18: PCI Target Image 0 Base Address Register (LSI0_BS)

Register Name: LSI0_BS		Offset:104
Bits	**Function**	
31-24	BS	
23-16	BS	
15-08	BS	Reserved
07-00	Reserved	

Table B-19: LSI0_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:28]	R/W	All	Power-up Option	Base Address
BS[27:12]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

The base address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Table B-20: PCI Target Image 0 Bound Address Register(LSI0_BD)

Register Name: LSI0_BD		Offset:108
Bits	**Function**	
31-24	BD	
23-16	BD	
15-08	BD	Reserved
07-00	Reserved	

Table B-21: LSI0_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:28]	R/W	All	Power-up Option	Bound Address
BD[27:12]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

The bound address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64 Kbyte resolution.

Table B-22: PCI Target Image 0 Translation Offset (LSI0_TO)

Register Name: LSI0_TO								Offset:10C
Bits	**Function**							
31-24	TO							
23-16	TO							
15-08	TO					Reserved		
07-00	Reserved							

Table B-23: LSI0_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:12]	R/W	All	0	Translation Offset

The translation offset for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Address bits [31:12] generated on the VMEbus in response to an image decode are a two’s complement addition of address bits [31:12] on the PCI Bus and bits [31:12] of the image’s translation offset.

Table B-24: PCI Target Image 1 Control (LSI1_CTL)

Register Name: LSI1_CTL								Offset:114
Bits	**Function**							
31-24	EN	PWEN	Reserved					
23-16	VDW		Reserved			VAS		
15-08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07-00	Reserved							LAS

Table B-25: LSI1_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor

Table 2-25: LSI1_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-26: PCI Target Image 1 Base Address Register (LSI1_BS)

Register Name: LSI1_BS	Offset:118
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-27: LSI1_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-28: PCI Target Image 1 Bound Address Register (LSI1_BD)

Register Name: LSI1_BD	Offset:11C
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-29: LSI1_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

The bound address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Table B-30: PCI Target Image 1 Translation Offset (LSI1_TO)

Register Name: LSI1_TO	Offset:120
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-31: LSI1_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	All	0	Translation offset

Address bits [31:16] generated on the VMEbus in response to an image decode are a two’s complement addition of address bits [31:16] on the PCI Bus and bits [31:16] of the image’s translation offset.

Table B-32: PCI Target Image 2 Control (LSI2_CTL)

<table>
<tr><th colspan="5">Register Name: LSI2_CTL</th><th colspan="4">Offset:128</th></tr>
<tr><th>Bits</th><th colspan="8">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="6">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="3">Reserved</td><td colspan="3">VAS</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>PGM</td><td>Reserved</td><td>SUPER</td><td colspan="3">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td colspan="7">Reserved</td><td>LAS</td></tr>
</table>

Table B-33: LSI2_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTson VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-34: PCI Target Image 2 Base Address Register (LSI2_BS)

Register Name: LSI2_BS	Offset:12C
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-35: LSI2_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B- 36: PCI Target Image 2 Bound Address Register (LSI2_BD)

Register Name: LSI2_BD	**Offset:130**
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-37: LSI2_BD Description

Name	**Type**	**Reset By**	**Reset State**	**Function**
BD[31:16]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

Table B-38: PCI Target Image 2 Translation Offset

Register Name: LSI2_TO	**Offset:134**
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-39: LSI2_TO Description

Name	**Type**	**Reset By**	**Reset State**	**Function**
TO[31:16]	R/W	All	0	Translation offset

Address bits [31:16] generated on the VMEbus in response to an image decode are a two's complement addition of address bits [31:16] on the PCI Bus and bits [31:16] of the image's translation offset.

Table B-40: PCI Target Image 3 Control (LSI3_CTL)

Register Name: LSI3_CTL								**Offset:13C**
Bits	**Function**							
31-24	EN	PWEN	Reserved					
23-16	VDW		Reserved			VAS		
15-08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07-00	Reserved							LAS

Table B-41: LSI3_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-42: PCI Target Image 3 Base Address Register (LSI3_BS)

Register Name: LSI3_BS		**Offset:140**
Bits	**Function**	
31-24	BS	
23-16	BS	
15-08	Reserved	
07-00	Reserved	

Table B-43: LSI3_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-44: PCI Target Image 3 Bound Address Register (LSI3_BD)

Register Name: LSI3_BD	Offset:144
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-45: TableLSI3_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

Table B-46: PCI Target Image 3 Translation Offset (LSI3_TO)

Register Name: LSI3_TO	Offset:148
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-47: LSI3_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	All	0	Translation Offset

Address bits [31:16] generated on the VMEbus in response to an image decode are a two’s complement addition of address bits [31:16] on the PCI Bus and bits [31:16] of the image’s translation offset.

Table B-48: Special Cycle Control Register (SCYC_CTL)

Register Name: SCYC_CTL			Offset: 170
Bits	**Function**		
31-24	Reserved		
23-16	Reserved		
15-08	Reserved		
07-00	Reserved	LAS	SCYC

Table B-49: SCYC_CTL Description

Name	Type	Reset By	Reset State	Function
LAS	R/W	All	0	PCI Bus Address Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space
SCYC	R/W	All	0	Special Cycle 00=Disable, 01=RMW, 10=ADOH, 11=Reserved

The special cycle generator will generate an ADOH or RMW cycle for the 32-bit PCI Bus address which matches the programmed address in SCYC_ADDR, in the address space specified in the LAS field of the SCYC_CTL register. A Read-Modify-Write command is initiated by a read to the specified address. Address-Only cycles are initiated by either read or write cycles.

Table B-50: Special Cycle PCI Bus Address Register (SCYC_ADDR)

Register Name: SCYC_ADDR		Offset: 174
Bits	**Function**	
31-24	ADDR	
23-16	ADDR	
15-08	ADDR	
07-00	ADDR	Reserved

Table B-51: SCYC_ADDR Description

Name	Type	Reset By	Reset State	Function
ADDR[31:2]	R/W	All	0	Address

This register designates the special cycle address. This address must appear on the PCI Bus during the address phase of a transfer for the Special Cycle Generator to perform its function. Whenever the addresses match, the Universe II does not respond with ACK64#

Table B-52: Special Cycle Swap/Compare Enable Register (SCYC_EN)

Register Name: SCYC_EN	Offset: 178
Bits	**Function**
31-24	EN
23-16	EN
15-08	EN
07-00	EN

Table B-53: SCYC_EN Description

Name	Type	Reset By	Reset State	Function
EN[31:0]	R/W	All	0	Bit Enable 0=Disable, 1=Enable

The bits enabled in this register determine the bits that will be involved in the compare and swap operations for VME RMW cycles.

Table B-54: Special Cycle Compare Data Register

Register Name: SCYC_CMP	Offset: 17C
Bits	**Function**
31-24	CMP
23-16	CMP
15-08	CMP
07-00	CMP

Table B-55: SCYC_CMP Description

Name	Type	Reset By	Reset State	Function
CMP[31:0]	R/W	All	0	The data returned from the VMEbus is compared with the contents of this register.

The data returned from the read portion of a VMEbus RMW is compared with the contents of this register. SCYC_EN is used to control which bits are compared.

Table B-56: Special Cycle Swap Data Register (SCYC_SWP)

Register Name: SCYC_SWP	Offset: 180
Bits	**Function**
31-24	SWP
23-16	SWP
15-08	SWP
07-00	SWP

Table B-57: SCYC_SWP Description

Name	Type	Reset By	Reset State	Function
SWP[31:0]	R/W	All	0	Swap data

If enabled bits matched with the value in the compare register, then the contents of the swap data register is written back to VME. SCYC_EN is used to control which bits are written back to VME.

Table B-58: PCI Miscellaneous Register (LMISC)

<table>
<tr><th colspan="2">Register Name: LMISC</th><th colspan="2">Offset:184</th></tr>
<tr><th>Bits</th><th colspan="3">Function</th></tr>
<tr><td>31-24</td><td>CRT[3:0]</td><td>Reserved</td><td>CWT</td></tr>
<tr><td>23-16</td><td colspan="3">Reserved</td></tr>
<tr><td>15-08</td><td colspan="3">Reserved</td></tr>
<tr><td>07-00</td><td colspan="3">Reserved</td></tr>
</table>

Table B-59: SLSI Description

Name	Type	Reset By	Reset State	Function
CRT[3:0]	R/W	All	0000	CRT This field is provided for backward compatibility with the Universe I. It has no effect on the operation of the Universe II.
CWT [2:0]	R/W	All	000	Coupled Window Timer 000=Disable - release after first coupled transaction, 001=16 PCI Clocks, 010=32 PCI Clocks, 011=64 PCI Clocks, 100=128 PCI Clocks, 101=246 PCI Clocks, 110=512 PCI Clocks, others=Reserved

The Universe II uses CWT to determine how long to hold ownership of the VMEbus after processing a coupled transaction. The timer is restarted each time the Universe II processes a coupled transaction. If this timer expires, the PCI Slave Channel releases the VMEbus.

Device behaviour is unpredictable if CWT is changed during coupled cycle activity. This register can only be set at configuration or after disabling all PCI Slave Images.

Table B-60: Special PCI Target Image (SLSI)

<table>
<tr><th colspan="4">Register Name: SLSI</th><th colspan="3">Offset:188</th></tr>
<tr><th>Bits</th><th colspan="6">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="4">Reserved</td></tr>
<tr><td>23-16</td><td colspan="3">VDW</td><td colspan="3">Reserved</td></tr>
<tr><td>15-08</td><td colspan="3">PGM</td><td colspan="3">SUPER</td></tr>
<tr><td>07-00</td><td colspan="4">BS</td><td>Reserved</td><td>LAS</td></tr>
</table>

Table B-61: SLSI Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW [3:0]	R/W	All	0	VMEbus Maximum Datawidth Each of the four bits specifies a data width for the corresponding 16 MByte region. Low order bits correspond to the lower address regions. 0=16-bit, 1=32-bit
PGM [3:0]	R/W	All	0	Program/Data AM Code Each of the four bits specifies Program/Data AM code for the corresponding 16 MByte region. Low order bits correspond to the lower address regions. 0=Data, 1=Program

Table B-61: SLSI Description (continued)

Name	Type	Reset By	Reset State	Function
SUPER [3:0]	R/W	All	0	Supervisor/User AM Code Each of the four bits specifies Supervisor/User AM code for the corresponding 16 MByte region. Low order bits correspond to the lower address regions. 0=Non-Privileged, 1=Supervisor
BS [5:0]	R/W	All	0	Base Address Specifies a 64 MByte aligned base address for this 64 MByte image.
LAS	R/W	All	0	PCI Bus Address Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

This register fully specifies an A32 capable special PCI Target Image. The base is programmable to a 64 Mbyte alignment, and the size is fixed at 64 Mbytes. Incoming address lines [31:26] (in Memory or I/O) must match this field for the Universe II to decode the access. This special PCI Target Image has lower priority than any other PCI Target Image.

The 64 Mbytes of the SLSI is partitioned into four 16 Mbyte regions, numbered 0 to 3 (0 is at the lowest address). PCI address bits [25:24] are used to select regions. The top 64 Kbyte of each region is mapped to VMEbus A16 space, and the rest of each 16 Mbyte region is mapped to A24 space.

The user can use the PGM, SUPER and VDW fields to specify the AM code and the maximum port size for each region. The PGM field is ignored for the portion of each region mapped to A16 space.

No block transfer AM codes are generated.

Table B-62: PCI Command Error Log Register (L_CMDERR)

<table>
<tr><th colspan="4">Register Name: L_CMDERR</th><th>Offset: 18C</th></tr>
<tr><th>Bits</th><th colspan="4">Function</th></tr>
<tr><td>31-24</td><td colspan="2">CMDERR</td><td>M_ERR</td><td>Reserved</td></tr>
<tr><td>23-16</td><td>L_STAT</td><td colspan="3">Reserved</td></tr>
<tr><td>15-08</td><td colspan="4">Reserved</td></tr>
<tr><td>07-00</td><td colspan="4">Reserved</td></tr>
</table>

Table B-63: L_CMDERR Description

Name	Type	Reset By	Reset State	Function
CMDERR [3:0]	R	All	0111	PCI Command Error Log
M_ERR	R	All	0	Multiple Error Occurred 0=Single error, 1=At least one error has occurred since the logs were frozen.

Table B-63: L_CMDERR Description (continued)

Name	Type	Reset By	Reset State	Function
L_STAT	R/W	All	0	PCI Error Log Status Reads: 0=logs invalid, 1=logs are valid and error logging halted Writes: 0=no effect, 1=clears L_STAT and enables error logging

The Universe II PCI Master Interface is responsible for logging errors under the following conditions:

- a posted write transaction results in a target abort
- a posted write transaction results in a master abort
- a maximum retry counter expires during retry of posted write transaction

This register logs the command information.

Table B-64: PCI Address Error Log (LAERR)

Register Name: LAERR	Offset: 190
Bits	**Function**
31-24	LAERR
23-16	LAERR
15-08	LAERR
07-00	LAERR

Table B-65: LAERR Description

Name	Type	Reset By	Reset State	Function
LAERR [31:0]	R	All	0	PCI address error log

The starting address of an errored PCI transaction is logged in this register under the following conditions:

- a posted write transaction results in a target abort,
- a posted write transaction results in a master abort, or
- a maximum retry counter expires during retry of posted write transaction.

Contents are qualified by bit L_STAT of the L_CMDERR register.

Table B-66: PCI Target Image 4 Control Register (LSI4_CTL)

<table>
<tr><th colspan="9">Register Name: LSI4_CTL Offset:1A0</th></tr>
<tr><th>Bits</th><th colspan="8">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="6">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="3">Reserved</td><td colspan="3">VAS</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>PGM</td><td>Reserved</td><td>SUPER</td><td colspan="3">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td colspan="7">Reserved</td><td>LAS</td></tr>
</table>

Table B-67: LSI4_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-68: PCI Target Image 4 Base Address Register (LSI4_BS)

Register Name: LSI4_BS		**Offset:1A4**
Bits	**Function**	
31-24	BS	
23-16	BS	
15-08	BS	Reserved
07-00	Reserved	

Table B-69: LSI4_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-70: PCI Target Image 4 Bound Address Register (LSI4_BD)

Register Name: LSI4_BD		**Offset:1A**
Bits	**Function**	
31-24	BD	
23-16	BD	
15-08	BD	Reserved
07-00	Reserved	

Table B-71: LSI4_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:12]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

The bound address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Table B-72: PCI Target Image 4 Translation Offset (LSI4_TO)

Register Name: LSI4_TO		**Offset:1B0**
Bits	**Function**	
31-24	TO	
23-16	TO	
15-08	TO	Reserved
07-00	Reserved	

Table B-73: LSI4_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:12]	R/W	All	0	Translation Offset

Address bits [31:12] generated on the VMEbus in response to an image decode are a two’s complement addition of address bits [31:12] on the PCI Bus and bits [31:12] of the image’s translation offset.

Table B-74: PCI Target Image 5 Control Register (LSI5_CTL)

<table>
<tr><th colspan="6">Register Name: LSI5_CTL</th><th colspan="2">Offset:1B4</th></tr>
<tr><th>Bits</th><th colspan="7">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="5">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="3">Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>PGM</td><td>Reserved</td><td>SUPER</td><td colspan="2">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td colspan="6">Reserved</td><td>LAS</td></tr>
</table>

Table B-75: LSI5_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-76: PCI Target Image 5 Base Address Register (LSI5_BS)

Register Name: LSI5_BS	**Offset:1B8**
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-77: LSI5_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-78: PCI Target Image 5 Bound Address Register (LSI5_BD)

Register Name: LSI5_BD	**Offset:1BC**
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-79: LSI5_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

The bound address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Table B-80: PCI Target Image 5 Translation Offset (LSI5_TO)

Register Name: LSI5_TO	**Offset:1C0**
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-81: LSI5_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	All	0	Translation offset

Address bits [31:16] generated on the VMEbus in response to an image decode are a two's complement addition of address bits [31:16] on the PCI Bus and bits [31:16] of the image's translation offset.

Table B-82: PCI Target Image 6 Control Register (LSI6_CTL)

<table>
<tr><th colspan="6">Register Name: LSI6_CTL</th><th colspan="2">Offset:1C8</th></tr>
<tr><th>Bits</th><th colspan="7">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="5">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="3">Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>PGM</td><td>Reserved</td><td>SUPER</td><td colspan="2">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td colspan="6">Reserved</td><td>LAS</td></tr>
</table>

Table B- 83: LSI6_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-84: PCI Target Image 6 Base Address Register (LSI6_BS)

Register Name: LSI6_BS	Offset:1CC
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-85: LSI1_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-86: PCI Target Image 6 Translation Offset (LS16_TO)

Register Name: LSI6_BD	Offset:1D0
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-87: LSI6_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

The bound address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Table B-88: PCI Target Image 6 Translation Offset (LSI6_TO)

Register Name: LSI6_TO	Offset:1D4
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-89: LSI6_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	All	0	Translation Offset

Address bits [31:16] generated on the VMEbus in response to an image decode are a two's complement addition of address bits [31:16] on the PCI Bus and bits [31:16] of the image's translation offset.

Table B-90: PCI Target Image 7 Control Register (LSI7_CTL)

<table>
<tr><th colspan="8">Register Name: LSI7_CTL Offset:1DC</th></tr>
<tr><th>Bits</th><th colspan="7">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td colspan="5">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="3">Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>PGM</td><td>Reserved</td><td>SUPER</td><td colspan="2">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td colspan="6">Reserved</td><td>LAS</td></tr>
</table>

Table B-91: LSI7_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	All	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	All	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	All	10	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 0=Data, 1=Program
SUPER	R/W	All	0	Supervisor/User AM Code 0=Non-Privileged, 1=Supervisor
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LAS	R/W	All	0	PCI Bus Memory Space 0=PCI Bus Memory Space, 1=PCI Bus I/O Space

In the PCI Target Image Control register, setting the VCT bit will only have effect if the VAS bits are programmed for A24 or A32 space and the VDW bits are programmed for 8-bit, 16-bit, or 32-bit.

If VAS bits are programmed to A24 or A32 and the VDW bits are programmed for 64-bit, the Universe II may perform MBLT transfers independent of the state of the VCT bit.

The setting of the PWEN bit is ignored if the LAS bit is programmed for PCI Bus I/O Space, forcing all transactions through this image to be coupled.

Table B-92: PCI Target Image 7 Base Address Register (LSI7_BS)

Register Name: LSI7_BS	Offset:1E0
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-93: LSI7_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	All	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-94: PCI Target Image 7 Bound Address Register (LSI7_BD)

Register Name: LSI7_BD	Offset:1E4
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-95: LSI7_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	All	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

The bound address for PCI Target Image 0 and PCI Target Image 4 have a 4Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64Kbyte resolution.

Table B-96: PCI Target Image 7 Translation Offset (LSI7_TO)

Register Name: LSI7_TO	Offset:1E8
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-97: LSI7_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	All	0	Translation offset

Address bits [31:16] generated on the VMEbus in response to an image decode are a two's complement addition of address bits [31:16] on the PCI Bus and bits [31:16] of the image's translation offset.

Table B-98: DMA Transfer Control Register (DCTL)

<table>
<tr><th colspan="7">Register Name: DCTL Offset:200</th></tr>
<tr><th>Bits</th><th colspan="6">Function</th></tr>
<tr><td>31-24</td><td>L2V</td><td colspan="5">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">VDW</td><td colspan="2">Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td colspan="2">PGM</td><td>SUPER</td><td colspan="2">Reserved</td><td>VCT</td></tr>
<tr><td>07-00</td><td>LD64EN</td><td colspan="5">Reserved</td></tr>
</table>

Table B-99: DCTL Description

Name	Type	Reset By	Reset State	Function
L2V	R/W	All	0	Direction 0=Transfer from VMEbus to PCI Bus, 1=Transfer from PCI Bus to VMEbus
VDW	R/W	All	0	VMEbus Maximum Datawidth 00=8-bit data width, 01=16 bit data width, 10=32-bit data width, 11=64-bit data width
VAS	R/W	All	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
PGM	R/W	All	0	Program/Data AM Code 00=Data, 01=Program, others=Reserved
SUPER	R/W	All	0	Supervisor/User AM Code 00=Non-Privileged, 01=Supervisor, others=Reserved
VCT	R/W	All	0	VMEbus Cycle Type 0=no BLTs on VMEbus, 1=BLTs on VMEbus
LD64EN	R/W	All	1	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable

This register is programmed from either bus or is programmed by the DMAC when it loads the command packet. The DMA only accesses PCI Bus Memory space.

The VCT bit determines whether or not the Universe II VME Master will generate BLT transfers. The value of this bit only has meaning if the address space is A24 or A32 and the data width is not 64 bits. If the data width is 64 bits the Universe II may perform MBLT transfers independent of the state of the VCT bit.

Table B-100: DMA Transfer Byte Count Register (DTBC)

Register Name: DTBC	Offset:204
Bits	**Function**
31-24	Reserved
23-16	DTBC
15-08	DTBC
07-00	DTBC

Table B-101: DTBC Description

Name	Type	Reset By	Reset State	Function
DTBC[23:0]	R/W	All	0	DMA Transfer Byte Count

This register specifies the number of bytes to be moved by the DMA before the start of the DMA transfer, or the number of remaining bytes in the transfer while the DMA is active. This register is programmed from either bus or is programmed by the DMA Controller when it loads a command packet from a linked-list.

In direct mode the user must reprogram the DTBC register before each transfer.

When using the DMA to perform linked-list transfers, it is essential that the DTBC register contains a value of zero before setting the GO bit of the DGCS register or undefined behaviors may occur.

Table B-102: DMA PCI Bus Address Register (DLA)

Register Name: DLA	Offset:208
Bits	**Function**
31-24	LA
23-16	LA
15-08	LA
07-00	LA

Table B-103: DLA Description

Name	Type	Reset By	Reset State	Function
LA[31:3]	R/W	All	0	PCI Bus Address
LA[2:0]	R/W	All	0	PCI Bus Address

This register is programmed from either bus or by the DMA Controller when it loads a command packet. In direct mode the user must reprogram the DLA register before each transfer. In linked-list mode, this register is only updated when the DMA is stopped, halted, or at the completion of processing a command packet.

After a Bus Error, a Target-Abort, or a Master-Abort, the value in the DLA register must not be used to reprogram the DMA because it has no usable information. Some offset from its original value must be used.

Address bits [2:0] must be programmed the same as those in the DVA.

Table B-104: DMA VMEbus Address Register (DVA)

Register Name: DVA	Offset:210
Bits	**Function**
31-24	VA
23-16	VA
15-08	VA
07-00	VA

Table B-105: DVA Description

Name	Type	Reset By	Reset State	Function
VA[31:0]	R/W	All	0	VMEbus Address

This register is programmed from either bus or is programmed by the DMA Controller when it loads a command packet. In direct mode the user must reprogram the DVA register before each transfer. In linked-list operation, this register is only updated when the DMA is stopped, halted, or at the completion of processing a command packet.

After a Bus Error, a Target-Abort, or a Master-Abort, the value in the DLA register must not be used to reprogram the DMA because it has no usable information. Some offset from its original value must be used.

Address bits [2:0] must be programmed the same as those in the DLA.

Table B-106: DMA Command Packet Pointer (DCPP)

Register Name: DCPP		Offset: 218
Bits	**Function**	
31-24	DCPP	
23-16	DCPP	
15-08	DCPP	
07-00	DCPP	Reserved

Table B-107: DCPP Description

Name	Type	Reset By	Reset State	Function
DCPP[31:5]	R/W	All	0	DMA Command Packet Pointer

This register contains the pointer into the current command packet. Initially it is programmed to the starting packet of the linked-list, and is updated with the address to a new command packet at the completion of a packet. The packets must be aligned to a 32-byte address.

Table B-108: DMA General Control/Status Register (DGCS)

Register Name: DGCS								Offset: 220
Bits	**Function**							
31-24	GO	STOP_REQ	HALT_REQ	0	CHAIN	0	0	0
23-16	Reserved	VON			VOFF			
15-08	ACT	STOP	HALT	0	DONE	LERR	VERR	P_ERR
07-00	0	INT_STOP	INT_HALT	0	INT_DONE	INT_LERR	INT_VERR	INT_P_ERR

Table B-109: DGCS Description

Name	Type	Reset By	Reset State	Function
GO	W/Read 0 always	All	0	DMA Go Bit 0=No effect, 1=Enable DMA Transfers
STOP_REQ	W/Read 0 always	All	0	DMA Stop Request 0=No effect, 1=Stop DMA transfer when all buffered data has been written
HALT_REQ	W/Read 0 always	All	0	DMA Halt Request 0=No effect, 1=Halt the DMA transfer at the completion of the current command packet
CHAIN	R/W	All	0	DMA Chaining 0=DMA Direct Mode, 1=DMA Linked List mode
VON [2:0]	R/W	All	0	VMEbus “On” counter 000=Until done, 001=256 bytes, 010=512 bytes, 011=1024 bytes, 100=2048 bytes, 101=4096 bytes, 110=8192 bytes, 111=16384 bytes, others=Reserved
VOFF [3:0]	R/W	All	0	VMEbus “Off” Counter 0000=0µs, 0001=16µs, 0010=32µs, 0011=64µs, 0100=128µs, 0101=256µs, 0110=512µs, 0111=1024µs, 1000=2µs, 1001=4µs, 1010=8µs, others=Reserved The DMA will not re-request the VME Master until this timer expires.
ACT	R	All	0	DMA Active Status Bit 0=Not Active, 1=Active
STOP	R/Write 1 to Clear	All	0	DMA Stopped Status Bit 0=Not Stopped, 1=Stopped
HALT	R/Write 1 to Clear	All	0	DMA Halted Status Bit 0=Not Halted, 1=Halted
DONE	R/Write 1 to Clear	All	0	DMA Done Status Bit 0=Not Complete, 1=Complete
LERR	R/Write 1 to Clear	All	0	DMA PCI Bus Error Status Bit 0=No Error, 1=Error
VERR	R/Write 1 to Clear	All	0	DMA VMEbus Error Status Bit 0=No Error, 1=Error
P_ERR	R/Write 1 to Clear	All	0	DMA Programming Protocol Error Status Bit Asserted if PCI master interface disabled or lower three bits of PCI and VME addresses differ 0=No Error, 1=Error

Table B-109: DGCS Description (continued)

Name	Type	Reset By	Reset State	Function
INT_STOP	R/W	All	0	Interrupt when Stopped 0=Disable, 1=Enable
INT_HALT	R/W	All	0	Interrupt when Halted 0=Disable, 1=Enable
INT_DONE	R/W	All	0	Interrupt when Done 0=Disable, 1=Enable
INT_LERR	R/W	All	0	Interrupt on LERR 0=Disable, 1=Enable
INT_VERR	R/W	All	0	Interrupt on VERR 0=Disable, 1=Enable
INT_P_ERR	R/W	All	0	Interrupt on Master Enable Error 0=Disable, 1=Enable

STOP, HALT, DONE, LERR, VERR, and P_ERR must be cleared before the GO bit is enabled.

Table B-110: DMA Linked List Update Enable Register (D_LLUE)

Register Name: D_LLUE		Offset:224
Bits	**Function**	
31-24	UPDATE	Reserved
23-16	Reserved	
15-08	Reserved	
07-00	Reserved	

Table B-111: D_LLUE Description

Name	Type	Reset By	Reset State	Function
UPDATE	R/W	All	0	DMA Linked List Update Enable 0=PCI Resource not Updating Linked List 1=PCI Resource Updating Linked List

The PCI Resource must read back a logic 1 in the UPDATE field before proceeding to modify the linked list. After the Linked List has been modified the PCI Resource must clear the UPDATE field by writing a logic 0. the Universe II does not prevent an external master, from the PCI bus or the VMEbus, from writing to the other DMA registers. See Linked List Updating on page 136.

Table B-112: PCI Interrupt Enable Register (LINT_EN)

Register Name: LINT_EN								Offset:300
Bits	**Function**							
31-24	Reserved							
23-16	LM3	LM2	LM1	LM0	MBOX3	MBOX2	MBOX1	MBOX0
15-08	ACFAIL	SYSFAIL	SW_INT	SW_IACK	Reserved	VERR	LERR	DMA
07-00	VIRQ7	VIRQ6	VIRQ5	VIRQ4	VIRQ3	VIRQ2	VIRQ1	VOWN

Table B-113: LINT_EN Description

Name	Type	Reset By	Reset State	Function
LM3	R/W	All	0	Location Monitor 3 Mask 0=LM3 Interrupt Masked, 1=LM3 Interrupt Enabled
LM2	R/W	All	0	Location Monitor 2 Mask 0=LM2 Interrupt Masked, 1=LM2 Interrupt Enabled
LM1	R/W	All	0	Location Monitor 1 Mask 0=LM1 Interrupt Masked, 1=LM1 Interrupt Enabled
LM0	R/W	All	0	Location Monitor 0 Mask 0=LM0 Interrupt Masked, 1=LM0 Interrupt Enabled
MBOX3	R/W	All	0	Mailbox 3 Mask 0=MBOX3 Interrupt Masked, 1=MBOX3 Interrupt Enabled
MBOX2	R/W	All	0	Mailbox 2 Mask 0=MBOX2 Interrupt Masked, 1=MBOX2 Interrupt Enabled
MBOX1	R/W	All	0	Mailbox 1 Mask 0=MBOX1 Interrupt Masked, 1=MBOX1 Interrupt Enabled
MBOX0	R/W	All	0	Mailbox 0 Mask 0=MBOX0 Interrupt Masked, 1=MBOX0 Interrupt Enabled
ACFAIL	R/W	All	0	ACFAIL Interrupt Mask 0=ACFAIL Interrupt masked 1=ACFAIL Interrupt enabled
SYSFAIL	R/W	All	0	SYSFAIL Interrupt Mask 0=SYSFAIL Interrupt masked 1=SYSFAIL Interrupt enabled
SW_INT	R/W	All	0	Local Software Interrupt Mask 0=PCI Software Interrupt masked 1=PCI Software Interrupt enabled A zero-to-one transition will cause the PCI software interrupt to be asserted. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the PCI Software Interrupt Status bit.
SW_IACK	R/W	All	0	“VME Software IACK” Mask 0 =“VME Software IACK” Interrupt masked 1 =“VME Software IACK” Interrupt enabled
VERR	R/W	All	0	PCI VERR Interrupt Mask 0 =PCI VERR Interrupt masked 1=PCI VERR Interrupt enabled
LERR	R/W	All	0	PCI LERR Interrupt Mask 0 =PCI LERR Interrupt masked 1 =PCI LERR Interrupt enabled

Table B-113: LINT_EN Description (continued)

Name	Type	Reset By	Reset State	Function
DMA	R/W	All	0	PCI DMA Interrupt Mask 0=PCI DMA Interrupt masked 1=PCI DMA Interrupt enabled
VIRQ7-VIRQ1	R/W	All	0	VIRQx Interrupt Mask 0=VIRQx Interrupt masked 1 =VIRQx Interrupt enabled
VOWN	R/W	All	0	VOWN Interrupt Mask 0=VOWN Interrupt masked 1=VOWN Interrupt Enabled

Bits VIRQ7-VIRQ1 enable the Universe II to respond as a VME Interrupt Handler to interrupts on the VIRQ[x] lines. When a VIRQx interrupt is enabled, and the corresponding VIRQ[x] pin is asserted, the Universe II requests the VMEbus and performs a VME IACK cycle for that interrupt level. When the interrupt acknowledge cycle completes, the STATUS/ID is stored in the corresponding VINT_ID register, the VIRQx bit of the LINT_STAT register is set, and a PCI interrupt is generated. The Universe II does not acquire further interrupt STATUS/ID vectors at the same interrupt level until the VIRQx bit in the LINT_STAT register is cleared.

The other bits enable the respective internal or external sources to interrupt the PCI side.

Table B-114: PCI Interrupt Status Register (LINT_STAT)

Register Name: LINT_STAT								Offset: 304
Bits	Function							
31-24	Reserved							
23-16	LM3	LM2	LM1	LM0	MBOX3	MBOX2	MBOX1	MBOX0
15-08	ACFAIL	SYSFAIL	SW_INT	SW_IACK	Reserved	VERR	LERR	DMA
07-00	VIRQ7	VIRQ6	VIRQ5	VIRQ4	VIRQ3	VIRQ2	VIRQ1	VOWN

Table B-115: LINT_STAT Description

Name	Type	Reset By	Reset State	Function
LM3	R/Write 1 to Clear	All	0	Location Monitor 3 Status/Clear 0=no Location Monitor 3 Interrupt, 1=Location Monitor 3 Interrupt active
LM2	R/Write 1 to Clear	All	0	Location Monitor 2 Status/Clear 0=no Location Monitor 2 Interrupt, 1=Location Monitor 2 Interrupt active
LM1	R/Write 1 to Clear	All	0	Location Monitor 1 Status/Clear 0=no Location Monitor 1 Interrupt, 1=Location Monitor 1 Interrupt active
LM0	R/Write 1 to Clear	All	0	Location Monitor 0 Status/Clear 0=no Location Monitor 0 Interrupt, 1=Location Monitor 0 Interrupt active
MBOX3	R/Write 1 to Clear	All	0	Mailbox 3 Status/Clear 0=no Mailbox 3 Interrupt, 1=Mailbox 3 Interrupt active

Table B-115: LINT_STAT Description (continued)

Name	Type	Reset By	Reset State	Function
MBOX2	R/Write 1 to Clear	All	0	Mailbox 2 Status/Clear 0=no Mailbox 2 Interrupt, 1=Mailbox 2 Interrupt active
MBOX1	R/Write 1 to Clear	All	0	Mailbox 1 Status/Clear 0 =no Mailbox 1 Interrupt, 1=Mailbox 1 Interrupt active
MBOX0	R/Write 1 to Clear	All	0	Mailbox 0 Status/Clear 0=no Mailbox 0 Interrupt, 1=Mailbox 0 Interrupt active
ACFAIL	R/Write 1 to Clear	All	0	ACFAIL Interrupt Status/Clear 0=no ACFAIL Interrupt, 1=ACFAIL Interrupt active
SYSFAIL	R/Write 1 to Clear	All	0	SYSFAIL Interrupt Status/Clear 0=no SYSFAIL Interrupt, 1=SYSFAIL Interrupt active
SW_INT	R/Write 1 to Clear	All	0	Local Software Interrupt Status/Clear 0=no PCI Software Interrupt, 1=PCI Software Interrupt active
SW_IACK	R/Write 1 to Clear	All	0	“VME Software IACK” Status/Clear 0=no “VME Software IACK” Interrupt, 1=“VME Software IACK” Interrupt active
VERR	R/Write 1 to Clear	All	0	Local VERR Interrupt Status/Clear 0=Local VERR Interrupt masked, 1=Local VERR Interrupt enabled
LERR	R/Write 1 to Clear	All	0	Local LERR Interrupt Status/Clear 0=Local LERR Interrupt masked, 1=Local LERR Interrupt enabled
DMA	R/Write 1 to Clear	All	0	Local DMA Interrupt Status/Clear 0=Local DMA Interrupt masked, 1=Local DMA Interrupt enabled
VIRQ7-VIRQ1	R/Write 1 to Clear	All	0	VIRQx Interrupt Status/Clear 0=VIRQx Interrupt masked, 1=VIRQx Interrupt enabled
VOWN	R/Write 1 to Clear	All	0	VOWN Interrupt Status/Clear 0=no VOWN Interrupt masked, 1=VOWN Interrupt enabled

Status bits indicated as “R/Write 1 to Clear” are edge sensitive: the status is latched when the interrupt event occurs. These status bits can be cleared independently of the state of the interrupt source by writing a “1” to the status register. Clearing the status bit does not imply the source of the interrupt is cleared.

However, ACFAIL and SYSFAIL are level-sensitive. Clearing ACFAIL or SYSFAIL while their respective pins are sill asserted will have no effect.

Table B-116: PCI Interrupt Map 0 Register (LINT_MAP0)

Register Name: LINT_MAP0	**Offset: 308**							
Bits	**Function**							
31-24	Reserved	VIRQ7			Reserved	VIRQ6		
23-16	Reserved	VIRQ5			Reserved	VIRQ4		
15-08	Reserved	VIRQ3			Reserved	VIRQ2		
07-00	Reserved	VIRQ1			Reserved	VOWN		

Table B-117 LINT_MAP0 Description

Name	Type	Reset By	Reset State	Function
VIRQ7-VIRQ1	R/W	All	0	PCI interrupt destination (LINT[7:0]) for VIRQx
VOWN	R/W	All	0	VMEbus ownership bit interrupt map to PCI interrupt

Caution

Do not map any VMEbus interrupt to LINT#(2-7).

This register maps various interrupt sources to one of the eight PCI interrupt pins. A value of 000 maps the corresponding interrupt source to LINT# [0], a value of 001 maps to LINT# [1], etc.

Table B-118 PCI Interrupt Map 1 Register (LINT_MAP1)

Register Name: LINT_MAP1				**Offset: 30C**
Bits	**Function**			
31-24	Reserved	ACFAIL	Reserved	SYSFAIL
23-16	Reserved	SW_INT	Reserved	SW_IACK
15-08	Reserved			VERR
07-00	Reserved	LERR	Reserved	DMA

Table B-119: LINT_MAP1 Description

Name	Type	Reset By	Reset State	Function
ACFAIL	R/W	All	0	ACFAIL interrupt destination
SYSFAIL	R/W	All	0	SYSFAIL interrupt destination
SW_INT	R/W	All	0	PCI software interrupt destination
SW_IACK	R/W	All	0	VMEbus Software IACK interrupt destination
VERR	R/W	All	0	VMEbus Error interrupt destination
LERR	R/W	All	0	PCI Bus Error interrupt destination
DMA	R/W	All	0	DMA interrupt destination

Caution

Do not map any VMEbus interrupt to LINT#(2-7).

This register maps various interrupt sources to one of the eight PCI interrupt pins. A value of 000 maps the corresponding interrupt source to LINT# [0], a value of 001 maps to LINT# [1], etc.

Table B-120: VMEbus Interrupt Enable Register (VINT_EN)

Register Name: VINT_EN								**Offset:310**
Bits	**Function**							
31-24	SW_INT7	SW_INT6	SW_INT5	SW_INT4	SW_INT3	SW_INT2	SW_INT1	Reserved
23-16	Reserved				MBOX3	MBOX2	MBOX1	MBOX0
15-08	Reserved			SW_INT	Reserved	VERR	LERR	DMA
07-00	LINT7	LINT6	LINT5	LINT4	LINT3	LINT2	LINT1	LINT0

Table B-121: VINT_EN Description

Name	**Type**	**Reset By**	**Reset State**	**Function**
SW_INT7	R/W	All	0	VME Software 7 Interrupt Mask 0=VME Software 7 Interrupt masked, 1=VME Software 7 Interrupt enabled A zero-to-one transition will cause a VME level 7 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 7 Interrupt Status bit.
SW_INT6	R/W	All	0	VME Software 6 Interrupt Mask 0=VME Software 6 Interrupt masked, 1=VME Software 6 Interrupt enabled A zero-to-one transition will cause a VME level 6 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 6 Interrupt Status bit.
SW_INT5	R/W	All	0	VME Software 5 Interrupt Mask 0=VME Software 5 Interrupt masked, 1=VME Software 5 Interrupt enabled A zero-to-one transition will cause a VME level 5 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 5 Interrupt Status bit.
SW_INT4	R/W	All	0	VME Software 4 Interrupt Mask 0=VME Software 4 Interrupt masked, 1=VME Software 4 Interrupt enabled A zero-to-one transition will cause a VME level 4 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 4 Interrupt Status bit.
SW_INT3	R/W	All	0	VME Software 3 Interrupt Mask 0=VME Software 3 Interrupt masked, 1=VME Software 3 Interrupt enabled A zero-to-one transition will cause a VME level 3 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 3 Interrupt Status bit.

Table B-121: VINT_EN Description (continued)

Name	Type	Reset By	Reset State	Function
SW_INT2	R/W	All	0	VME Software 2 Interrupt Mask 0=VME Software 2 Interrupt masked, 1=VME Software 2 Interrupt enabled A zero-to-one transition will cause a VME level 2 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 2 Interrupt Status bit.
SW_INT1	R/W	All	0	VME Software 1 Interrupt Mask 0=VME Software 1 Interrupt masked, 1=VME Software 1 Interrupt enabled A zero-to-one transition will cause a VME level 1 interrupt to be generated. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the VME Software 1 Interrupt Status bit.
MBOX3	R/W	All	0	Mailbox 3 Mask 0=MBOX3 Interrupt masked, 1=MBOX3 Interrupt enabled
MBOX2	R/W	All	0	Mailbox 2 Mask 0=MBOX2 Interrupt masked, 1=MBOX2 Interrupt enabled
MBOX1	R/W	All	0	Mailbox 1 Mask 0=MBOX1 Interrupt masked, 1=MBOX1 Interrupt enabled
MBOX0	R/W	All	0	Mailbox 0 Mask 0=MBOX0 Interrupt masked, 1=MBOX0 Interrupt enabled
SW_INT	R/W	All	Power-up Option	“VME Software Interrupt” Mask 0 = VME Software Interrupt masked 1 =VME Software Interrupt enabled A zero-to-one transition causes the VME software interrupt to be asserted. Subsequent zeroing of this bit causes the interrupt to be masked and the VMEbus interrupt negated, but does not clear the VME software interrupt status bit.
VERR	R/W	All	0	VERR Interrupt Mask 0 =PCI VERR Interrupt masked 1=PCI VERR Interrupt enabled
LERR	R/W	All	0	LERR Interrupt Mask 0 =PCI LERR Interrupt masked 1 =PCI LERR Interrupt enabled
DMA	R/W	All	0	DMA Interrupt Mask 0=PCI DMA Interrupt masked 1=PCI DMA Interrupt enabled
LINT7-LINT0	R/W	All	0	PCI Interrupt Mask 0=LINTx Interrupt masked 1 =LINTx Interrupt enabled

This register enables the various sources of VMEbus interrupts. SW_INT can be enabled with the VME64AUTO power-up option.

Table B-122: VMEbus Interrupt Status Register (VINT_STAT)

Register Name: VINT_STAT								**Offset:314**
Bits	**Function**							
31-24	SW_INT7	SW_INT6	SW_INT5	SW_INT4	SW_INT3	SW_INT2	SW_INT1	Reserved
23-16	Reserved				MBOX3	MBOX2	MBOX1	MBOX0
15-08	Reserved			SW_INT	Reserved	VERR	LERR	DMA
07-00	LINT7	LINT6	LINT5	LINT4	LINT3	LINT2	LINT1	LINT0

Table B-123: VINT_STAT Description

Name	**Type**	**Reset By**	**Reset State**	**Function**
SW_INT7	R/Write 1 to clear	All	0	VME Software 7 Interrupt Status/Clear 0=no VME Software 7 Interrupt, 1=VME Software 7 Interrupt active
SW_INT6	R/Write 1 to clear	All	0	VME Software 6 Interrupt Status/Clear 0=no VME Software 6 Interrupt, 1=VME Software 6 Interrupt active
SW_INT5	R/Write 1 to clear	All	0	VME Software 5 Interrupt Status/Clear 0=no VME Software 5 Interrupt, 1=VME Software 5 Interrupt active
SW_INT4	R/Write 1 to clear	All	0	VME Software 4 Interrupt Status/Clear 0=no VME Software 4 Interrupt, 1=VME Software 4 Interrupt active
SW_INT3	R/Write 1 to clear	All	0	VME Software 3 Interrupt Status/Clear 0=no VME Software 3 Interrupt, 1=VME Software 3 Interrupt active
SW_INT2	R/Write 1 to clear	All	0	VME Software 2 Interrupt Status/Clear 0=no VME Software 2 Interrupt, 1=VME Software 2 Interrupt active
SW_INT1	R/Write 1 to clear	All	0	VME Software 1 Interrupt Status/Clear 0=no VME Software 1 Interrupt, 1=VME Software 1 Interrupt active
MBOX3	R/Write 1 to clear	All	0	Mailbox 3 Status/Clear 0=no Mailbox 3 Interrupt, 1=Mailbox 3 Interrupt active
MBOX2	R/Write 1 to clear	All	0	Mailbox 2 Status/Clear 0=no Mailbox 2 Interrupt, 1=Mailbox 2 Interrupt active
MBOX1	R/Write 1 to clear	All	0	Mailbox 1 Status/Clear 0=no Mailbox 1 Interrupt, 1=Mailbox 1 Interrupt active
MBOX0	R/Write 1 to clear	All	0	Mailbox 0 Status/Clear 0=no Mailbox 0 Interrupt, 1=Mailbox 0 Interrupt active
SW_INT	R/Write 1 to Clear	All	Power-up Option	VME Software Interrupt Status/Clear 0=VME Software Interrupt inactive, 1=VME Software Interrupt active
VERR	R/Write 1 to Clear	All	0	VERR Interrupt Status/Clear 0=VME VERR Interrupt masked, 1=VME VERR Interrupt enabled

Table B-123: VINT_STAT Description (continued)

Name	Type	Reset By	Reset State	Function
LERR	R/Write 1 to Clear	All	0	LERR Interrupt Status/Clear 0=VME LERR Interrupt masked, 1=VME LERR Interrupt enabled
DMA	R/Write 1 to Clear	All	0	DMA Interrupt Status/Clear 0=VME DMA Interrupt masked, 1=VME DMA Interrupt enabled
LINT7-LINT0	R/Write 1 to Clear	All	0	LINTx Interrupt Status/Clear 0=LINTx Interrupt masked, 1=LINTx Interrupt enabled

SW_INT can be set with the VME64AUTO power-up option.

Table B-124: VME Interrupt Map 0 Register (VINT_MAP0)

Register Name: VINT_MAP0				Offset: 318
Bits	**Function**			
31-24	Reserved	LINT7	Reserved	LINT6
23-16	Reserved	LINT5	Reserved	LINT4
15-08	Reserved	LINT3	Reserved	LINT2
07-00	Reserved	LINT1	Reserved	LINT0

Table B-125: VINT_MAP0 Description

Name	Type	Reset By	Reset State	Function
LINT7-LINT0	R/W	All	0	VMEbus destination of PCI Bus interrupt source

This register maps various interrupt sources to one of the seven VMEbus interrupt pins. A value of 001 maps the corresponding interrupt source to VIRQ*[1], a value of 002 maps to VIRQ*[2], etc. A value of 000 effectively masks the interrupt since there is no corresponding VIRQ*[0].

Table B-126: VME Interrupt Map 1 Register (VINT_MAP1)

Register Name: VINT_MAP1				Offset: 31C
Bits	**Function**			
31-24	Reserved			
23-16	Reserved			SW_INT
15-08	Reserved			VERR
07-00	Reserved	LERR	Reserved	DMA

Table B-127: VINT_MAP1 Description

Name	Type	Reset By	Reset State	Function
SW_INT	R/W	All	Power-up Option	VMEbus Software interrupt destination
VERR	R/W	All	0	VMEbus Error interrupt destination
LERR	R/W	All	0	PCI Bus Error interrupt destination
DMA	R/W	All	0	DMA interrupt destination

This register maps various interrupt sources to one of the seven VMEbus interrupt pins. A value of 001 maps the corresponding interrupt source to VIRQ*[1], a value of 002 maps to VIRQ*[2], etc. A value of 000 effectively masks the interrupt since there is no corresponding VIRQ*[0].

SW_INT is set to 010 with the VME64AUTO power-up option.

Table B-128: Interrupt STATUS/ID Out Register (STATID)

Register Name: STATID	Offset: 320
Bits	**Function**
31-24	STATID [7:0]
23-16	Reserved
15-08	Reserved
07-00	Reserved

Table B-129: STATID Description

Name	Type	Reset By	Reset State	Function
STATID [7:1]	R/W	All	1111111	Bits [7:1] of the STATUS/ID byte are returned when the Universe II responds to a VMEbus IACK cycle.
STATID [0]	R	All	See below	0 = the Universe II is generating a SW_IACK at the same level as the interrupt acknowledge cycle. 1 = the Universe II is not generating a SW_IACK at the same level as the interrupt acknowledge cycle.

When the Universe II responds to an interrupt acknowledge cycle on VMEbus it returns an 8-bit STATUS/ID. STATID [7:1] can be written by software to uniquely identify the VMEbus module within the system. STATID [0] is a value of 0 if the Universe II is generating a software interrupt (SW_IACK) at the same level as the interrupt acknowledge cycle, otherwise it is a value of 1.

The reset state is designed to support the VME64 Auto ID STATUS/ID value.

Table B-130: VIRQ1 STATUS/ID Register (V1_STATID)

Register Name: V1_STATID		Offset: 324
Bits	**Function**	
31-24	Reserved	
23-16	Reserved	
15-08	Reserved	ERR
07-00	STATID [7:0]	

Table B-131: V1_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 1 VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table B-132: VIRQ2 STATUS/ID Register (V2_STATID)

<table>
<tr><th colspan="3">Register Name: V2_STATID   Offset: 328</th></tr>
<tr><th>Bits</th><th colspan="2">Function</th></tr>
<tr><td>31-24</td><td colspan="2">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">Reserved</td></tr>
<tr><td>15-08</td><td>Reserved</td><td>ERR</td></tr>
<tr><td>07-00</td><td colspan="2">STATID [7:0]</td></tr>
</table>

Table B-133: V2_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 1 VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table B-134: VIRQ3 STATUS/ID Register (V3_STATID)

Register Name: V3_STATID								**Offset: 32C**
Bits	**Function**							
31-24	Reserved							
23-16	Reserved							
15-08	Reserved							ERR
07-00	STATID [7:0]							

Table B-135: V3_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 3VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table B-136: VIRQ4 STATUS/ID Register (V4_STATID)

Register Name: V4_STATID		**Offset: 330**
Bits	**Function**	
31-24	Reserved	
23-16	Reserved	
15-08	Reserved	ERR
07-00	STATID [7:0]	

Table B-137: V4_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 4 VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table 138: VIRQ5 STATUS/ID Register (V5_STATID)

Register Name: V5_STATID		**Offset: 334**
Bits	**Function**	
31-24	Reserved	
23-16	Reserved	
15-08	Reserved	ERR
07-00	STATID [7:0]	

Table 139: V5_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 5 VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table B-140: VIRQ6 STATUS/ID Register (V6_STATID)

Register Name: V6_STATID		**Offset: 338**
Bits	**Function**	
31-24	Reserved	
23-16	Reserved	
15-08	Reserved	ERR
07-00	STATID [7:0]	

Table B-141: V6_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 6 VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table B-142: VIRQ7 STATUS/ID Register (V7_STATID)

Register Name: V7_STATID		Offset: 33C
Bits	**Function**	
31-24	Reserved	
23-16	Reserved	
15-08	Reserved	ERR
07-00	STATID [7:0]	

Table B-143: V7_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	All	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	All	0	STATUS/ID acquired during IACK cycle for level 7 VMEbus interrupt

The Vx_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe II performs a IACK cycle for a given interrupt level. The Universe II is enabled as the interrupt handler for a given interrupt level via the VIRQx bits of the LINT_EN register. Once a vector for a given level is acquired, the Universe II does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQx bit in the LINT_STAT register is cleared.

The acquisition of a level x STATUS/ID by the Universe II updates the STATUS/ID field of the corresponding Vx_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.

Table B-144: PCI Interrupt Map 2 Register (LINT_MAP2)

Register Name: LINT_MAP2				**Offset: 340**
Bits	**Function**			
31-24	Reserved	LM3	Reserved	LM2
23-16	Reserved	LM1	Reserved	LM0
15-08	Reserved	MBOX3	Reserved	MBOX2
07-00	Reserved	MBOX1	Reserved	MBOX0

Table B-145: LINT_MAP2 Description

Name	Type	Reset By	Reset State	Function
LM3 [2:0]	R/W	All	0	Location Monitor 3 Interrupt destination
LM2 [2:0]	R/W	All	0	Location Monitor 2 Interrupt destination
LM1 [2:0]	R/W	All	0	Location Monitor 1 Interrupt destination
LM0 [2:0]	R/W	All	0	Location Monitor 0 Interrupt destination
MBOX3 [2:0]	R/W	All	0	Mailbox 3 Interrupt destination
MBOX2 [2:0]	R/W	All	0	Mailbox 2 Interrupt destination
MBOX1 [2:0]	R/W	All	0	Mailbox 1 Interrupt destination
MBOX0 [2:0]	R/W	All	0	Mailbox 0 Interrupt destination

Caution

Do not map any VMEbus interrupt to LINT#(2-7).

This register maps various interrupt sources to one of the eight PCI interrupt pins. A value of 000 maps the corresponding interrupt source to LINT# [0], a value of 001 maps to LINT# [1], etc.

Table B-146: VME Interrupt Map 2 Register (VINT_MAP2)

Register Name: VINT_MAP2				**Offset: 344**
Bits	**Function**			
31-24	Reserved			
23-16	Reserved			
15-08	Reserved	MBOX3	Reserved	MBOX2
07-00	Reserved	MBOX1	Reserved	MBOX0

Table B-147: VINT_MAP2 Description

Name	Type	Reset By	Reset State	Function
MBOX3 [2:0]	R/W	All	0	Mailbox 3 Interrupt destination
MBOX2 [2:0]	R/W	All	0	Mailbox 2 Interrupt destination
MBOX1 [2:0]	R/W	All	0	Mailbox 1 Interrupt destination
MBOX0 [2:0]	R/W	All	0	Mailbox 0 Interrupt destination

This register maps various interrupt sources to one of the seven VMEbus interrupt pins. A value of 001 maps the corresponding interrupt source to VIRQ*[1], a value of 002 maps to VIRQ*[2], etc. A value of 000 effectively masks the interrupt since there is no corresponding VIRQ*[0].

Table B-148: Mailbox 0 Register (MBOX0)

Register Name: MBOX0	**Offset:348**
Bits	**Function**
31-24	MBOX0
23-16	MBOX0
15-08	MBOX0
07-00	MBOX0

Table B-149: DVA Description

Name	Type	Reset By	Reset State	Function
MBOX0 [31:0]	R/W	All	0	Mailbox

General purpose mailbox register. Writes to this register will cause interrupt generation on PCI or VMEbus if enabled in LINT_EN register.

Table B-150: Mailbox 1 Register (MBOX1)

Register Name: MBOX1	**Offset:34C**
Bits	**Function**
31-24	MBOX1
23-16	MBOX1
15-08	MBOX1
07-00	MBOX1

Table B-151: DVA Description

Name	Type	Reset By	Reset State	Function
MBOX1 [31:0]	R/W	All	0	Mailbox

General purpose mailbox register. Writes to this register will cause interrupt generation on PCI or VMEbus if enabled in LINT_EN register.

Table B-152: Mailbox 2 Register (MBOX2)

Register Name: MBOX2	**Offset:350**
Bits	**Function**
31-24	MBOX2
23-16	MBOX2
15-08	MBOX2
07-00	MBOX2

Table B-153: DVA Description

Name	Type	Reset By	Reset State	Function
MBOX2 [31:0]	R/W	All	0	Mailbox

General purpose mailbox register. Writes to this register will cause interrupt generation on PCI or VMEbus if enabled in LINT_EN register.

Table B-154: Mailbox 3 Register (MBOX3)

Register Name: MBOX3	Offset:354
Bits	**Function**
31-24	MBOX3
23-16	MBOX3
15-08	MBOX3
07-00	MBOX3

Table B-155: DVA Description

Name	Type	Reset By	Reset State	Function
MBOX3 [31:0]	R/W	All	0	Mailbox

General purpose mailbox register. Writes to this register will cause interrupt generation on PCI or VMEbus if enabled in LINT_EN register.

Table B-156: Semaphore 0 Register (SEMA0)

Register Name: SEMA0		Offset: 358
Bits	**Function**	
31-24	SEM3	TAG3
23-16	SEM2	TAG2
15-08	SEM1	TAG1
07-00	SEM0	TAG0

Table B-157: SEMA0 Description

Name	Type	Reset By	Reset State	Function
SEM3	R/W	All	0	Semaphore 3
TAG3 [6:0]	R/W	All	0	Tag 3
SEM2	R/W	All	0	Semaphore 2
TAG2 [6:0]	R/W	All	0	Tag2
SEM1	R/W	All	0	Semaphore 1
TAG1 [6:0]	R/W	All	0	Tag 1
SEM0	R/W	All	0	Semaphore 0
TAG0 [6:0]	R/W	All	0	Tag 0

If a semaphore bit is a value of 0, the associated tag field can be written to. If a semaphore bit is a value of 1, the associated tag field cannot be written to.

This register can only be accessed via byte-wide access.

Table B-158: Semaphore 1 Register (SEMA1)

Register Name: SEMA1		**Offset: 35C**
Bits	**Function**	
31-24	SEM7	TAG6
23-16	SEM6	TAG6
15-08	SEM5	TAG5
07-00	SEM4	TAG4

Table B-159: SEMA1 Description

Name	Type	Reset By	Reset State	Function
SEM3	R/W	All	0	Semaphore 7
TAG3 [6:0]	R/W	All	0	Tag 7
SEM2	R/W	All	0	Semaphore 6
TAG2 [6:0]	R/W	All	0	Tag 6
SEM1	R/W	All	0	Semaphore 5
TAG1 [6:0]	R/W	All	0	Tag 5
SEM0	R/W	All	0	Semaphore 4
TAG0 [6:0]	R/W	All	0	Tag 4

If a semaphore bit is a value of 0, the associated tag field can be written to. If a semaphore bit is a value of 1, the associated tag field cannot be written to.

This register can only be accessed via byte-wide access.

Table B-160: Master Control Register (MAST_CTL)

<table>
<tr><th colspan="4">Register Name: MAST_CTL</th><th colspan="3">Offset: 400</th></tr>
<tr><th>Bits</th><th colspan="6">Function</th></tr>
<tr><td>31-24</td><td colspan="3">MAXRTRY</td><td colspan="3">PWON</td></tr>
<tr><td>23-16</td><td>VRL</td><td>VRM</td><td>VREL</td><td>VOWN</td><td>VOWN_ACK</td><td>Reserved</td></tr>
<tr><td>15-08</td><td>Reserved</td><td colspan="2">PABS</td><td colspan="3">Reserved</td></tr>
<tr><td>07-00</td><td colspan="6">BUS_NO</td></tr>
</table>

Table B-161: MAST_CTL Description

Name	Type	Reset By	Reset State	Function
MAXRTRY [3:0]	R/W	All	1000	Maximum Number of Retries 0000=Retry Forever, Multiples of 64 (0001 through 1111). Maximum Number of retries before the PCI master interface signals error condition
PWON [3:0]	R/W	All	0000	Posted Write Transfer Count 0000=128 bytes, 0001=256 bytes, 0010=512 bytes, 0011=1024 bytes, 0100=2048 bytes, 0101=4096 bytes, 0110 - 1110 = Reserved, 1111=Early release of BBSY*. Transfer count at which the PCI Slave Channel Posted Writes FIFO gives up the VME Master Interface.
VRL [1:0]	R/W	All	11	VMEbus Request Level 00=Level 0,01=Level 1,10=Level 2, 11=Level 3
VRM	R/W	All	0	VMEbus Request Mode 0=Demand,1=Fair
VREL	R/W	All	0	VMEbus Release Mode 0=Release When Done (RWD), 1=Release on Request (ROR)
VOWN	W	All	0	VME Ownership Bit 0=Release VMEbus, 1=Acquire and Hold VMEbus
VOWN_ACK	R	All	0	VME Ownership Bit Acknowledge 0=VMEbus not owned, 1=VMEbus acquired and held due to assertion of VOWN
PABS [1:0]	R/W	All	00	PCI Aligned Burst Size 00=32-byte, 01=64-byte, 10=128-byte, 11=Reserved Controls the PCI address boundary at which the Universe II breaks up a PCI transaction in the VME Slave channel and the DMA Channel. This field also determines when the PCI Master Module as part of the VME Slave Channel will request the PCI bus (i.e., when 32, 64, or 128 bytes are available). It does not have this effect on the DMA Channel, which has a fixed watermark of 128 bytes.
BUS_NO [7:0]	R/W	All	0000 0000	PCI Bus Number

Writing a 1 to the VOWN bit in the MAST_CTL register has the effect of asserting BBSY* until a 0 is written to the VOWN bit. It does not affect the transactions in the PCI Target Channel. The Universe II will not do an early release of BBSY* if the VMEbus was owned during a transaction by means of VOWN, regardless of the value of PWON.

It is important to wait until VOWN_ACK is a value of 0 before writing a value of 1 to the VOWN bit.

In the event that BERR* is asserted on the VMEbus once the Universe II owns the VMEbus, the user must release ownership by programming the VOWN bit to a value of 0, if the VMEbus was gained by setting the VOWN bit. VMEbus masters must not write a value of 1 to the VOWN bit since this will lock up the VMEbus.

Once the value programmed in the PWON field is reached during dequeuing of posted writes, the Universe II will do an early release of BBSY*. If the PWON field is programmed to a value of 1111, the Universe II will do an early release of BBSY* at the completion of each transaction. Note that the VOWN setting described above overrides the POWN setting.

BUS_NO is used by the VMEbus Slave Channel when mapping VME transactions into PCI Configuration space. If the bus number of the VMEbus address (bits [23:16]) is equal to the BUS_NO field, then the Universe II generates a Type 0 configuration cycle, otherwise Type 1 is generated.

The PABS[1:0] field also determines when the PCI Master Module as part of the VME Slave Channel will request the PCI bus (i.e., when 32, 64, or 128 bytes are available). It does not have this effect on the DMA Channel, which has a fixed watermark of 128 bytes (see FIFO Operation and Bus Ownership on page 137).

Table B-162: Miscellaneous Control Register (MISC_CTL)

<table>
<tr><th colspan="5">Register Name: MISC_CTL</th><th colspan="4">Offset: 404</th></tr>
<tr><th>Bits</th><th colspan="8">Function</th></tr>
<tr><td>31-24</td><td colspan="4">VBTO</td><td>Reserved</td><td>VARB</td><td colspan="2">VARBTO</td></tr>
<tr><td>23-16</td><td>SW_LRST</td><td>SW_SRST</td><td>Reserved</td><td>BI</td><td>ENGBI</td><td>RESCIND</td><td>SYSCON</td><td>V64AUTO</td></tr>
<tr><td>15-08</td><td colspan="8">Reserved</td></tr>
<tr><td>07-00</td><td colspan="8">Reserved</td></tr>
</table>

Table B-163: MISC_CTL Description

Name	Type	Reset By	Reset State	Function
VBTO	R/W	All	0011	VME Bus Time-out 0000=Disable, 0001=16 µsec, 0010=32 µsec, 0011=64 µsec, 0100=128 µsec, 0101=256 µsec, 0110=512 µsec, 0111=1024 µsec, others=RESERVED
VARB	R/W	All	0	VMEbus Arbitration Mode 0=Round Robin, 1=Priority
VARBTO	R/W	All	01	VMEbus Arbitration Time-out 00=Disable Timer, 01=16 µs (minimum 8µs), 10=256 µs, others=Reserved
SW_LRST	W	All	0	Software PCI Reset 0=No effect, 1=Initiate LRST# A read always returns 0.
SW_SYSRST	W	All	0	Software VMEbus SYSRESET 0=No effect, 1=Initiate SYSRST* A read always returns 0.
BI	R/W	All	Power-up Option	BI-Mode 0=Universe II is not in BI-Mode, 1=Universe II is in BI-Mode Write to this bit to change the Universe II BI-Mode status. This bit is also affected by the global BI-Mode initiator VRIRQ1*, if this feature is enabled.
ENGBI	R/W	All	0	Enable Global BI-Mode Initiator 0=Assertion of VIRQ1 ignored, 1=Assertion of VIRQ1 puts device in BI-Mode

Table B-163: MISC_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
RESCIND	R/W	All	1	RESCIND is unused in the Universe II.
SYSCON	R/W	All	Power-up Option	SYSCON 0=Universe II is not VMEbus System Controller, 1=Universe II is VMEbus System Controller
V64AUTO	R/W	All	Power-up Option	VME64 Auto ID Write: 0=No effect, 1=Initiate sequence This bit initiates Universe II VME64 Auto ID Slave participation.

VMEbus masters must not write to SW_SYSRST, and PCI masters must not write to SW_LRST.

The bits VBTO, VARB and VARBTO support SYSCON functionality.

Universe II participation in the VME64 Auto ID mechanism is controlled by the VME64AUTO bit. When this bit is detected high, the Universe II uses the SW_IACK mechanism to generate VXIRQ2 on the VMEbus, then releases VXSYSFAIL. Access to the CR/CSR image is enabled when the level 2 interrupt acknowledge cycle completes. This sequence can be initiated with a power-up option or by software writing a 1 to this bit.

Table B-164: Miscellaneous Status Register (MISC_STAT)

Register Name: MISC_STAT								Offset: 408
Bits	Function							
31-24	Reserved	LCLSIZE	Reserved		DY4AUTO	Reserved		
23-16	Reserved		MYBBSY	Reserved	DY4_DONE	TXFE	RXFE	Reserved
15-08	DY4AUTOID							
07-00	Reserved							

Table B-165: MISC_STAT Description

Name	Type	Reset By	Reset State	Function
LCLSIZE	R	All	Power-up Option	PCI Bus Size At the trailing edge of RST#, the Universe II samples REQ64# to determine the PCI Bus size. This bit reflects the result. 0=32-bit, 1=64-bit
DY4AUTO	R	All	Power-up Option	DY4 Auto ID Enable 0=Disable, 1=Enable
MYBBSY	R	All	1	Universe II BBSY 0=Asserted,1=Negated
DY4DONE	R	All	0	DY4 Auto ID Done 0=Not done,1=Done
TXFE	R	All	1	PCI Target Channel Posted Writes FIFO 0=no data in the FIFO, 1=data in the FIFO
RXFE	R	All	1	VME Slave Channel Posted Writes FIFO 0=no data in the FIFO, 1=data in the FIFO
DY4AUTOID	R	None	0	DY4 Auto ID

Table B-166: User AM Codes Register (USER_AM)

Register Name: USER_AM				Offset: 40C
Bits	**Function**			
31-24	0	1	USER1AM	Reserved
23-16	0	1	USER2AM	Reserved
15-08	Reserved			
07-00	Reserved			

Table B167: USER_AM Description

Name	Type	Reset By	Reset State	Function
USER1AM [3:0]	R/W	All	0000	User AM Code 1
USER2AM [3:0]	R/W	All	0000	User AM Code 2

The reset state is one of the VME64 user-defined AM codes.

The USER_AM register can only be used to generate and accept AM codes 0x10 through 0x1F. These AM codes are designated as USERAM codes in the VMEbus specification.

Table B-168: VMEbus Slave Image 0 Control(VSI0_CTL)

Register Name: VSI0_CTL							Offset: F00
Bits	**Function**						
31-24	EN	PWEN	PREN	Reserved			
23-16	PGM		SUPER		Reserved	VAS	
15-08	Reserved						
07-00	LD64EN	LLRMW	Reserved				LAS

Table B-169: VSI0_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2

Table B-169: VSI0_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
LD64EN	R/W	PWR VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus). This image has 4 Kbyte resolution.

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-170: VMEbus Slave Image 0 Base Address Register (VSI0_BS)

Register Name: VSI0_BS			Offset: F04
Bits	**Function**		
31-24	BS		
23-16	BS		
15-08	BS	Reserved	
07-00	Reserved		

Table B-171: VSI0_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

This image has 4 Kbyte resolution.

Table B-172: VMEbus Slave Image 0 Bound Address Register (VSI0_BD)

Register Name: VSI0_BD			Offset: F08
Bits	**Function**		
31-24	BD		
23-16	BD		
15-08	BD	Reserved	
07-00	Reserved		

Table B-173: VSI0_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:12]	R/W	PWR VME	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

This image has 4 Kbyte resolution.

Table B-174: VMEbus Slave Image 0 Translation Offset (VSI0_TO)

Register Name: VSI0_TO		Offset: F0C
Bits	**Function**	
31-24	TO	
23-16	TO	
15-08	TO	Reserved
07-00	Reserved	

Table B-175: VSI0_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:12]	R/W	PWR VME	0	Translation Offset

This image has 4 Kbyte resolution.

Table B-176: VMEbus Slave Image 1 Control (VSI1_CTL)

Register Name: VSI1_CTL					Offset: F14
Bits	**Function**				
31-24	EN	PWEN	PREN	Reserved	
23-16	PGM		SUPER	Reserved	VAS
15-08	Reserved				
07-00	LD64EN	LLRMW	Reserved		LAS

Table B-177: VSI1_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable

Table B-177: VSI1_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	0	VMEbus Address Space 000=Reserved, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR VME	1	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	1	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus).

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-178: VMEbus Slave Image 1 Base Address Register (VSI1_BS)

Register Name: VSI1_BS	Offset: F18
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-179: VSI1_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-180: VMEbus Slave Image 1 Bound Address Register (VSI1_BD)

Register Name: VSI1_BD	Offset: F1C
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-181: VSI1_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	PWR VME	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

Table B-182: VMEbus Slave Image 1 Translation Offset (VSI1_TO)

Register Name: VSI1_TO	Offset: F20
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-183: VSI1_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	PWR VME	0	Translation Offset

Table B-184: VMEbus Slave Image 2 Control (VSI2_CTL)

Register Name: VSI2_CTL							Offset: F28
Bits	**Function**						
31-24	EN	PWEN	PREN	Reserved			
23-16	PGM		SUPER		Reserved	VAS	
15-08	Reserved						
07-00	LD64EN	LLRMW	Reserved				LAS

Table B-185: VSI2_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	0	VMEbus Address Space 000=Reserved, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus).

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-186: VMEbus Slave Image 2 Base Address Register (VSI2_BS)

Register Name: VSI2_BS	Offset: F2C
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-187: VSI2_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	PWR VME	0	Base Address

Table B-188: VMEbus Slave Image 2 Bound Address Register (VSI2_BD)

Register Name: VSI2_BD	Offset: F30
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-189: VSI2_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	PWR VME	0	Bound Address

Table B-190: VMEbus Slave Image 2 Translation Offset (VSI2_TO)

Register Name: VSI2_TO	Offset: F34
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-191: VSI2_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	PWR VME	0	Translation Offset

Table B-192: VMEbus Slave Image 3 Control (VSI3_CTL)

<table>
<tr><th colspan="8">Register Name: VSI3_CTL Offset: F3C</th></tr>
<tr><th>Bits</th><th colspan="7">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td>PREN</td><td colspan="4">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">PGM</td><td colspan="2">SUPER</td><td>Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td colspan="7">Reserved</td></tr>
<tr><td>07-00</td><td>LD64EN</td><td>LLRMW</td><td colspan="4">Reserved</td><td>LAS</td></tr>
</table>

Table B-193: VSI3_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both

Table B-193: VSI3_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	0	VMEbus Address Space 000=Reserved, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus).

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-194: VMEbus Slave Image 3 Base Address Register (VSI3_BS)

Register Name: VSI3_BS	Offset: F40
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-195: VSI3_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	PWR VME	0	Base Address

Table B-196: VMEbus Slave Image 3 Bound Address Register (VSI3_BD)

Register Name: VSI3_BD	Offset: F44
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-197: VSI3_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	PWR VME	0	Bound Address

Table B-198: VMEbus Slave Image 3 Translation Offset (VSI3_TO)

Register Name: VSI3_TO	**Offset: F48**
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-199: VSI3_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	PWR VME	0	Translation Offset

Table B-200: Location Monitor Control Register (LM_CTL)

<table>
<tr><th colspan="2">Register Name: LM_CTL</th><th colspan="4">Offset: F64</th></tr>
<tr><th>Bits</th><th colspan="5">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td colspan="4">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">PGM</td><td>SUPER</td><td>Reserved</td><td>VAS</td></tr>
<tr><td>15-08</td><td colspan="5">Reserved</td></tr>
<tr><td>07-00</td><td colspan="5">Reserved</td></tr>
</table>

Table B-201: LM_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011=Reserved, 100=Reserved, 101=Reserved, others=Reserved

This register specifies the VMEbus controls for the location monitor image. This image has a 4 Kbyte resolution and a 4 Kbyte size. The image responds to a VME read or write within the 4 Kbyte space and matching one of the address modifier codes specified. BLTs and MBLTs are not supported.

VMEbus address bits [4:3] are used to set the status bit in LINT_STAT for one of the four location monitor interrupts. If the Universe II VMEbus master is the owner of the VMEbus, the Universe II VMEbus slave will generate DTACK* to terminate the transaction.

The Location Monitor does not store write data and read data is undefined.

Table B-202: Location Monitor Base Address Register (LM_BS)

Register Name: LM_BS	**Offset: F68**
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-203: LM_BS Description

Name	Type	Reset By	Reset State	Function
BS [31:16]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the 4 Kbyte range that will be decoded as a location monitor access.

Table B-204: VMEbus Register Access Image Control Register (VRAI_CTL)

Register Name: VRAI_CTL				**Offset: F70**
Bits	**Function**			
31-24	EN	Reserved		
23-16	PGM	SUPER	Reserved	VAS
15-08	Reserved			
07-00	Reserved			

Table B-205: VRAI_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	Power-up Option	Image Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	Power-up Option	VMEbus Address Space 00=A16, 01=A24, 10=A32, all others are reserved

The VME Register Access Image allows access to the Universe II registers with standard VMEbus cycles. Only single cycle and lock AM codes are accepted. When a register is accessed with a RMW, it is locked for the duration of the transaction.

Table B-206: VMEbus Register Access Image Base Address Register (VRAI_BS)

Register Name: VRAI_BS		**Offset: F74**
Bits	**Function**	
31-24	BS	
23-16	BS	
15-08	BS	Reserved
07-00	Reserved	

Table B-207: VRAI_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	PWR VME	Power-up Option	The base address specifies the lowest address in the 4 Kbyte VMEbus Register Access Image.
VRAI_CTL: VAS	**BS [31:24]**	**BS [23:16]**	**BS [15:12]**	The reset state is a function of the Power-up Option behaviour of the VAS field in VRAI_CTL
A16	0	0	Power-up Option VA [28:25]	
A24	0	Power-up Option VA [28:21]	0	
A32	Power-up Option VA [28:21]	0	0	

Table B-208: VMEbus CSR Control Register (VCSR_CTL)

Register Name: VCSR_CTL		**Offset: F80**
Bits	**Function**	
31-24	EN	Reserved
23-16	Reserved	
15-08	Reserved	
07-00	Reserved	LAS

Table B-209: VCSR_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR,VME	0	Image Enable 0=Disable, 1=Enable
LAS	R/W	PWR VME	Power-up Option	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

The EN bit allows software to enable or disable the Universe II CR/CSR image. This image can also be enabled by the VME64 Auto ID process.

For CSR’s not supported in the Universe II and for CR accesses, the LAS field determines the PCI Bus command that will be generated when the cycle is mapped to the PCI Bus.

Table B-210: VMEbus CSR Translation Offset (VCSR_TO)

Register Name: VCSR_TO		Offset: F84
Bits	**Function**	
31-24	TO	
23-16	TO	Reserved
15-08	Reserved	
07-00	Reserved	

Table B-211: VCSR_TO Description

Name	Type	Reset By	Reset State	Function
TO [31:24]	R/W	PWR VME	0	Translation Offset
TO [23:19]	R/W	PWR VME	Power-up Option	Translation Offset

For CSR's not supported in the Universe II and for CR accesses, the translation offset is added to the 24-bit VMEbus address to produce a 32-bit PCI Bus address. Negative offsets are not supported.

Table B-212: VMEbus AM Code Error Log (V_AMERR)

Register Name: V_AMERR				Offset: F88
Bits	**Function**			
31-24	AMERR		IACK	M_ERR
23-16	V_STAT	Reserved		
15-08	Reserved			
07-00	Reserved			

Table B-213: V_AMERR Description

Name	Type	Reset By	Reset State	Function
AMERR [5:0]	R	PWR, VME	0	VMEbus AM Code Error Log
IACK	R	PWR, VME	0	VMEbus IACK Signal
M_ERR	R	PWR, VME	0	Multiple Error Occurred 0=Single error, 1=At least one error has occurred since the logs were frozen
V_STAT	R/W	PWR, VME	0	VME Error Log Status Reads: 0=logs invalid, 1=logs are valid and error logging halted Writes: 0=no effect, 1=clears V_STAT and enables error logging

The Universe II VMEbus Master Interface is responsible for logging the parameters of a posted write transaction that results in a bus error. This register holds the address modifier code and the state of the IACK* signal. The register contents are qualified by the V_STAT bit.

Table B-214: VMEbus Address Error Log (VAERR)

Register Name: VAERR	Offset: F8C
Bits	**Function**
31-24	VAERR
23-16	VAERR
15-08	VAERR
07-00	VAERR

Table B-215: VAERR Description

Name	Type	Reset By	Reset State	Function
VAERR [31:0]	R	PWR, VME	0	VMEbus address error log

The Universe II PCI Master Interface is responsible for logging the parameters of a posted write transaction that results in a bus error. This register holds the address. The register contents are qualified by the V_STAT bit of the V_AMERR register.

Table B-216: VMEbus Slave Image 4 Control (VSI4_CTL)

Register Name: VSI4_CTL							Offset: F90
Bits	**Function**						
31-24	EN	PWEN	PREN	Reserved			
23-16	PGM		SUPER		Reserved	VAS	
15-08	Reserved						
07-00	LD64EN	LLRMW	Reserved				LAS

Table B-217: VSI4_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR, VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR, VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR, VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR, VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR, VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR, VME	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR, VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable

Table B-217: VSI4_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
LLRMW	R/W	PWR, VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR, VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus). This image has 4 Kbyte resolution.

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-218: VMEbus Slave Image 4 Base Address Register (VSI4_BS)

Register Name: VSI4_BS		Offset: F94
Bits	Function	
31-24	BS	
23-16	BS	
15-08	BS	Reserved
07-00	Reserved	

Table B-219: VSI4_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

This image has 4 Kbyte resolution.

Table B-220: VMEbus Slave Image 4 Bound Address Register (VSI4_BD)

Register Name: VSI4_BD		Offset: F98
Bits	Function	
31-24	BD	
23-16	BD	
15-08	BD	Reserved
07-00	Reserved	

Table B-221: VSI4_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:12]	R/W	PWR VME	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound register is 0, then the addresses decoded are those greater than or equal to the base address.

This image has 4 Kbyte resolution.

Table B-222: VMEbus Slave Image 4 Translation Offset (VSI4_TO)

Register Name: VSI4_TO		**Offset: F9C**
Bits	**Function**	
31-24	TO	
23-16	TO	
15-08	TO	Reserved
07-00	Reserved	

Table B-223: VSI4_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:12]	R/W	PWR VME	0	Translation Offset

The translation offset is added to the source address that is decoded and this new address becomes the destination address. If a negative offset is desired, the offset must be expressed as a two’s complement.

This image has 4 Kbyte resolution.

Table B-224: VMEbus Slave Image 5 Control (VSI5_CTL)

Register Name: VSI5_CTL							**Offset: FA4**
Bits	**Function**						
31-24	EN	PWEN	PREN	Reserved			
23-16	PGM		SUPER		Reserved	VAS	
15-08	Reserved						
07-00	LD64EN	LLRMW	Reserved				LAS

Table B-225: VSI5_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both

Table B-225: VSI5_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
VAS	R/W	PWR VME	0	VMEbus Address Space 000=Reserved, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus).

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-226: VMEbus Slave Image 5 Base Address Register (VSI5_BS)

Register Name: VSI5_BS	Offset: FA8
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-227: VSI5_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-228: VMEbus Slave Image 5 Bound Address Register (VSI5_BD)

Register Name: VSI5_BD	Offset: FAC
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-229: VSI5_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	PWR VME	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound register is 0, then the addresses decoded are those greater than or equal to the base address,

Table B-230: VMEbus Slave Image 5 Translation Offset (VSI5_TO)

Register Name: VSI5_TO Offset: FB0	
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-231: VSI5_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	PWR VME	0	Translation Offset

The translation offset is added to the source address that is decoded and this new address becomes the destination address. If a negative offset is desired, the offset must be expressed as a two’s complement.

Table B-232: VMEbus Slave Image 6 Control (VS16_CTL)

Register Name: VSI6_CTL				Offset: FB8
Bits	**Function**			
31-24	EN	PWEN	PREN	Reserved
23-16	PGM	SUPER	Reserved	VAS
15-08	Reserved			
07-00	LD64EN	LLRMW	Reserved	LAS

Table B-233: VSI6_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both

Table B-233: VSI6_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	PWR VME	0	VMEbus Address Space 000=Reserved, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus).

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-234: VMEbus Slave Image 6 Base Address Register (VSI6_BS)

Register Name: VSI6_BS	Offset: FBC
Bits	Function
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-235: VSI6_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-236: VMEbus Slave Image 6 Bound Address Register (VSI6_BD)

Register Name: VSI6_BD	Offset: FC0
Bits	Function
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-237: VSI6_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	PWR VME	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound register is 0, then the addresses decoded are those greater than or equal to the base address,

Table B-238: VMEbus Slave Image 6 Translation Offset (VSI6_TO)

Register Name: VSI6_TO	Offset: FC4
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-239: VSI6_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	PWR VME	0	Translation Offset

The translation offset is added to the source address that is decoded and this new address becomes the destination address. If a negative offset is desired, the offset must be expressed as a two's complement.

Table B-240: VMEbus Slave Image 7 Control (VSI7_CTL)

<table>
<tr><th colspan="2">Register Name: VSI7_CTL</th><th colspan="5">Offset: FCC</th></tr>
<tr><th>Bits</th><th colspan="6">Function</th></tr>
<tr><td>31-24</td><td>EN</td><td>PWEN</td><td>PREN</td><td colspan="3">Reserved</td></tr>
<tr><td>23-16</td><td colspan="2">PGM</td><td>SUPER</td><td>Reserved</td><td colspan="2">VAS</td></tr>
<tr><td>15-08</td><td colspan="6">Reserved</td></tr>
<tr><td>07-00</td><td>LD64EN</td><td>LLRMW</td><td colspan="3">Reserved</td><td>LAS</td></tr>
</table>

Table B-241: VSI7_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR VME	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	PWR VME	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	PWR VME	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	PWR VME	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	PWR VME	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both

Table B-241: VSI7_CTL Description (continued)

Name	Type	Reset By	Reset State	Function
VAS	R/W	PWR VME	0	VMEbus Address Space 000=Reserved, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	PWR VME	0	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	PWR VME	0	Enable PCI Bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	PWR VME	0	PCI Bus Address Space 00=PCI Bus Memory Space, 01=PCI Bus I/O Space, 10=PCI Bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI Bus).

In order for a VMEbus slave image to respond to an incoming cycle, the BM bit in the PCI_CSR register must be enabled.

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Table B-242: VMEbus Slave Image 7 Base Address Register (VSI7_BS)

Register Name: VSI7_BS	Offset: FD0
Bits	**Function**
31-24	BS
23-16	BS
15-08	Reserved
07-00	Reserved

Table B-243: VSI7_BS Description

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	PWR VME	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

Table B-244: VMEbus Slave Image 7 Bound Address Register (VSI7_BD)

Register Name: VSI7_BD	Offset: FD4
Bits	**Function**
31-24	BD
23-16	BD
15-08	Reserved
07-00	Reserved

Table B-245: VSI7_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	PWR VME	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound register is 0, then the addresses decoded are those greater than or equal to the base address.

Table B-246: VMEbus Slave Image 7 Translation Offset (VSI7_TO)

Register Name: VSI7_TO	Offset: FD8
Bits	**Function**
31-24	TO
23-16	TO
15-08	Reserved
07-00	Reserved

Table B-247: VSI7_TO Description

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	PWR VME	0	Translation Offset

Table B-248: VMEbus CSR Bit Clear Register (VCSR_CLR)

Register Name: VCSR_CLR				Offset: FF4
Bits	**Function**			
31-24	RESET	SYSFAIL	FAIL	Reserved
23-16	Reserved			
15-08	Reserved			
07-00	Reserved			

Table B-249: VCSR_CLR Description

Name	Type	Reset By	Reset State	Function
RESET	R/W	PWR VME	0	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=negate LRST#
SYSFAIL	R/W	All	Power-up Option	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes:0=no effect, 1=negate VXSYSFAIL
FAIL	R	PWR VME	0	Board Fail 0=Board has not failed

This register implements the Bit Clear Register as defined in the VME64 specification. Note that the RESET bit can be written to only from the VMEbus.

Table B-250: VMEbus CSR Bit Set Register (VCSR_SET)

Register Name: VCSR_SET				Offset: FF8
Bits	**Function**			
31-24	RESET	SYSFAIL	FAIL	Reserved
23-16	Reserved			
15-08	Reserved			
07-00	Reserved			

Table B-251: VCSR_SET Description

Name	Type	Reset By	Reset State	Function
RESET	R/W	PWR VME	0	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=assert LRST#
SYSFAIL	R/W	All	Power-up Option	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes:0=no effect, 1=assert VXSYSFAIL
FAIL	R	PWR VME	0	Board Fail 0=Board has not failed

This register implements the Bit Set Register as defined in the VME64 specification. Note that the RESET bit can be written to only from the VMEbus. Writing 1 to the RESET bit asserts LRST#. The PCI reset remains asserted until a 1 is written to the RESET bit of the VCSR_CLR register.

Table B-252: VMEbus CSR Base Address Register (VCSR_BS)

Register Name: VCSR_BS		Offset: FFC
Bits	**Function**	
31-24	BS	Reserved
23-16	Reserved	
15-08	Reserved	
07-00	Reserved	

Table B-253: VCSR_BS Description

Name	Type	Reset By	Reset State	Function
BS [23:19]	R/W	PWR VME	0	Base Address

The base address specifies one of thirty-one available CR/CSR windows as defined in the VME64 specification. Each window consumes 512 Kbytes of CR/CSR space.

Appendix C Performance

Introduction

As a VMEbus bridge, the Universe II’s most important function is data transfer. This function is performed by its three channels: the PCI Slave Channel, the VME Slave Channel, and the DMA Channel. Since each channel operates independently of the others and because each has its own unique characteristics, the following analysis reviews the data transfer performance for each channel:

“PCI Slave Channel” on page C-3

“VME Slave Channel” on page C-8

“DMA Channel” on page C-14

“Summary” on page C-17

Where relevant, descriptions of factors affecting performance and how they might be controlled in different environments are discussed.

The performance characteristics specified herein are provided by Tundra corporation and were characterized under ideal conditions. These specifications do not take into account performance lowering situations such as: the use of non-ideal slaves, the implementation of endian-conversion, and the implementation of hardware designed to fix Universe II errata. Performance will thus be lower in light of these real-world conditions.

The decoupled nature of the Universe II can cause some confusion in discussing performance parameters. This is because, in a fully decoupled bus bridge each of the two opposing buses operates at its peak performance independently of the other. The Universe II, however, because of the finite size of its FIFOs does not represent a 100% decoupled bridge. As the FIFOs fill or empty (depending on the direction of data movement) the two buses tend to migrate to matched performance where the higher performing bus is forced to slow down to match the other bus. This limits the sustained performance of the device. Some factors such as the PCI Aligned Burst Size and VME request/release modes can limit the effect of FIFO size and enhance performance.

Another aspect in considering the performance of a device is bandwidth consumption. The greater bandwidth consumed to transfer a given amount of data, the less is available for other bus masters. Decoupling significantly improves the Universe II's bandwidth consumption, and on the PCI bus allows it to use the minimum permitted by the PCI specification.

To simplify the analysis and allow comparison with other devices, Universe II performance has been calculated using the following assumptions:

As a PCI master:

- one clock bus grant latency
- zero wait state PCI target

As a VME master:

- ideal VME slave response (DS* to DTACK* = 30ns)

Assumed as part of any calculation on VME performance is the inclusion of VME transceivers with propagation delay of 4 ns.

This appendix presents sustained performance values. In contrast, the Universe User manual (9000000.MD303.01) provided peak performance numbers. This explains why some of the performance numbers in this document appear to be lower than for the original Universe.

PCI Slave Channel

Coupled Cycles

Request of VMEbus

The Universe II has a “Coupled Window Timer” (CWT in the LMISC register) which permits the coupled channel to maintain ownership of the VMEbus for an extended period beyond the completion of a cycle. This permits subsequent coupled accesses to the VMEbus to occur back-to-back without requirement for re-arbitration.

The CWT should be set for the expected latency between sequential coupled accesses attempted by the CPU. In calculating the latency expected here, the designer needs to account for latency across their host PCI bridge as well as latency encountered in re-arbitration for the PCI bus between each coupled access. Care must be taken not to set the CWT greater than necessary as the Universe II blocks all decoupled write transactions with target-retry, while the coupled channel owns the VMEbus. It is only when the CWT has expired that the PCI bus is permitted to enqueue transactions in the TXFIFO.

When a coupled access to the VMEbus is attempted, the Universe II generates a target-retry to the PCI initiator if the coupled path does not currently own the VMEbus. This occurs if the Universe II is not currently VMEbus master, or if the DMA is currently VMEbus master or if entries exist in the TXFIFO.

If the Universe II does not have ownership of the VMEbus when a coupled access is attempted, the Universe II generates a target-retry with a single wait state (See Figure C-2). The request for the VMEbus occurs shortly after the cycle is retried.

Read Cycles

Once the coupled channel owns the VMEbus, the Universe II propagates the cycle out to the VMEbus. Figure C-2 shows such a coupled read cycle against an ideal VME slave. There are 10 wait states inserted by the Universe II on the PCI bus before it responds with TRDY#. Further wait states are inserted for each extra 30ns in slave response.

Performing 32-bit PCI reads from VME gives a sustained performance of approximately 8.5 MB/s. Figure C-2 shows several of these accesses occurring consecutively.

Figure C-1: Coupled Read Cycle - Universe II as VME Master

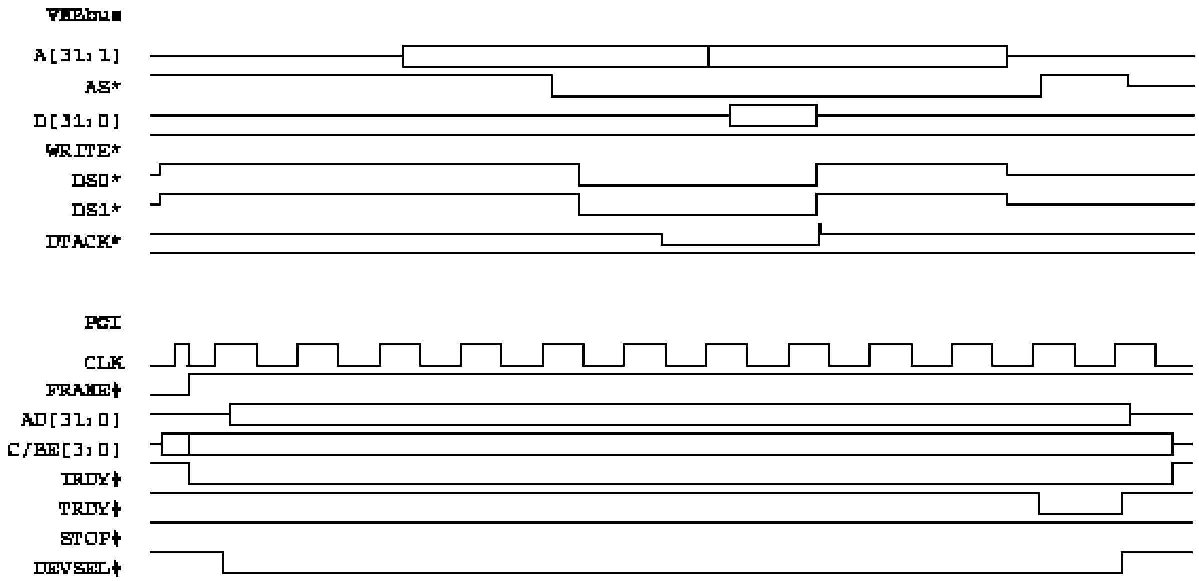
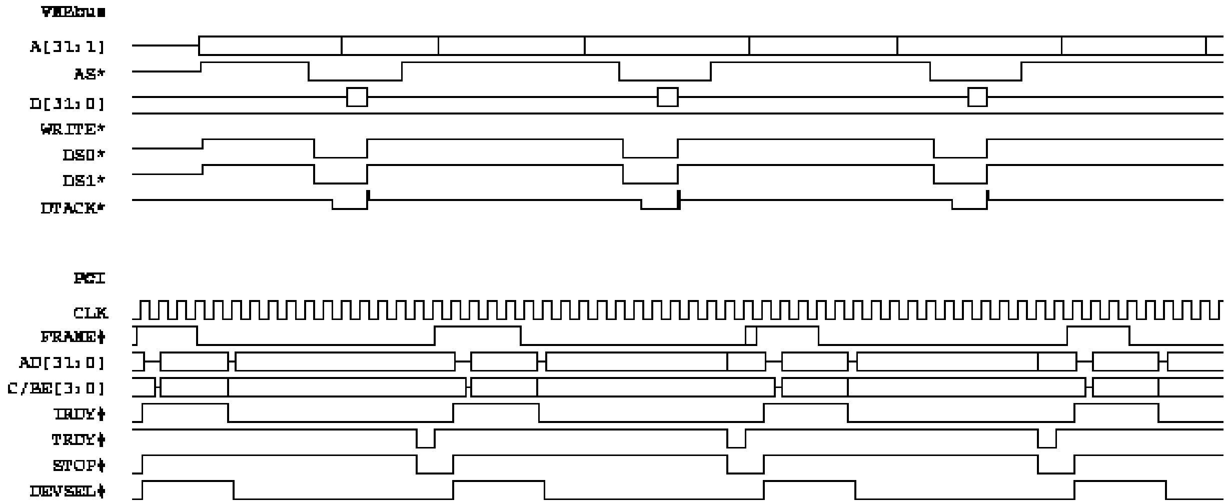


Figure C-2: Several Coupled Read Cycles - Universe II as VME Master

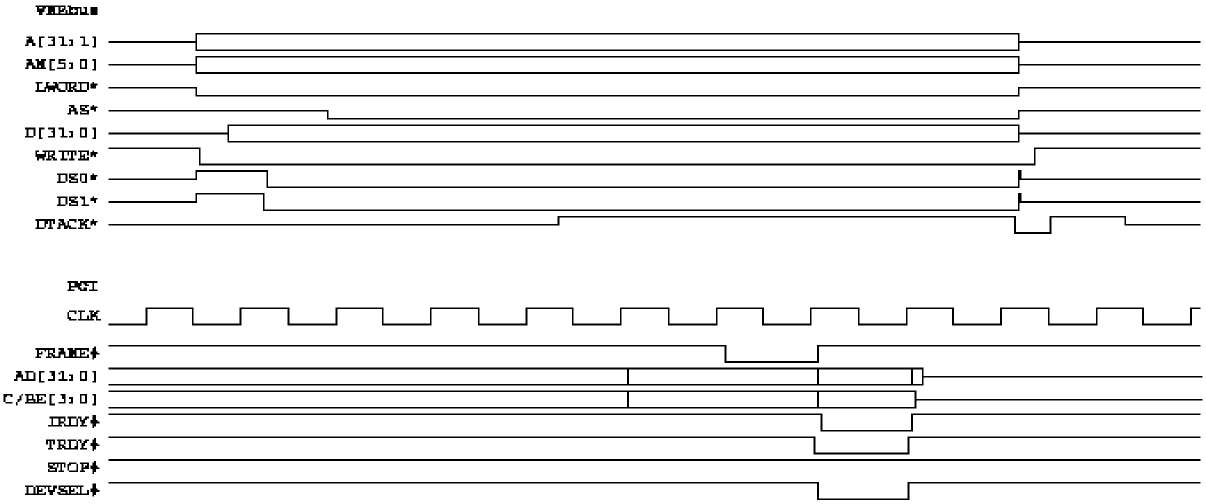


Write Cycles

The performance of coupled write cycles is similar to that of coupled read cycles except that an extra wait state is inserted. Figure C-3 shows a coupled write cycle against an ideal VME slave. Ten wait states are inserted on the PCI bus by the Universe II before it responds with TRDY#. A slower VME slave response translates directly to more wait states on the PCI bus.

The sustained performance, when generating write cycles from a 32-bit PCI bus against an ideal VME slave is approximately 9.3 MB/s.

Figure C-3: Coupled Write Cycle - Universe II as VME Master



Decoupled Cycles

Only write transactions can be decoupled in the PCI Target Channel.

Effect of the PWON Counter

The Posted Write On Counter (PWON in the MAST_CTL register) controls the maximum tenure that the PCI Slave Channel will have on the VMEbus. Once this channel has gained ownership of the VMEbus for use by the TXFIFO, it only relinquishes it if the FIFO becomes empty or if the number of bytes programmed in the counter expires. In most situations, the FIFO empties before the counter expires. However, if a great deal of data is being transferred by a PCI initiator to the VMEbus, then this counter ensures that only a fixed amount of VME bandwidth is consumed.

Limiting the size of the PWON counter imposes greater arbitration overhead on data being transferred out from the FIFO. This is true even when programmed for ROR mode since an internal arbitration cycle will still occur. The value for the PWON counter must be weighed from the system perspective with the impact of imposing greater latency on other channels (the DMA and Interrupt Channels) and other VME masters in gaining ownership of the VMEbus. On a Universe II equipped card which is only performing system control functions, the counter would be set to minimum. On a card which is responsible for transferring considerable amounts of performance-critical data the counter will be set much higher at the expense of system latency.

PCI Target Response

As the PCI target during decoupled write operations to the VMEbus, the Universe II responds in one of two manners:

1. It immediately issues a target retry because the FIFO does not have sufficient room for a burst of one address phase and 128 bytes of data. (There are no programmable watermarks in the PCI Target Channel. The PCI Aligned Burst Size (PABS) does not affect the PCI Target Channel.)
2. It responds as a zero-wait state target receiving up to 256 bytes in a transaction. When the FIFO is full or a 256-byte boundary has been reached, the Universe II issues a Target-Disconnect.

In either case, the Universe II will consume the minimum possible PCI bandwidth, never inserting wait states.

VME Master Performance

As a VME master, the Universe II waits until a full transaction has been enqueued in the Tx-FIFO before requesting the VMEbus and generating a VME cycle. If the VMEbus is already owned by the decoupled path (see “Effect of the PWON Counter” on page C-5), the Universe II still waits until a full transaction is enqueued in the FIFO before processing it.

If configured to generate non-block transfers, the Universe II can generate back-to-back VME transfers with cycle times of approximately 180ns (AS* to AS*) against an ideal VME slave (30-45 ns). A greater cycle time is required between the termination of one full enqueued transaction and the start of the next. This inter-transaction time is approximately 210ns. As such, the longer the PCI transaction, the greater the sustained performance on the VMEbus. With 64-byte PCI transactions, the sustained rate is 43 MB/s. With 32-byte transactions, this drops to 23 MB/s. Each of these numbers is calculated with no initial arbitration or re-arbitration for the bus. Figure C-4 shows the Universe II dequeueing a transaction with multiple non-block VME transfers.

Block transfers significantly increase performance. The inter-transaction period remains at approximately 210 ns for BLTs and MBLTs, but the data beat cycle time (DS* to DS*) drops to about 120ns against the same ideal slave. Again the length of the burst size affects the sustained performance because of the inter-transaction time. For BLTs operating with a burst size of 64 bytes, the sustained performance is 37 MB/s, dropping to 33 MB/s for a burst size of 32 bytes. MBLTs operating with 64-byte bursts perform at a sustained rate of 66 MB/s, dropping to 50 MB/s for 32 bytes.

Figure C-4: Several Non-Block Decoupled Writes - Universe II as VME Master

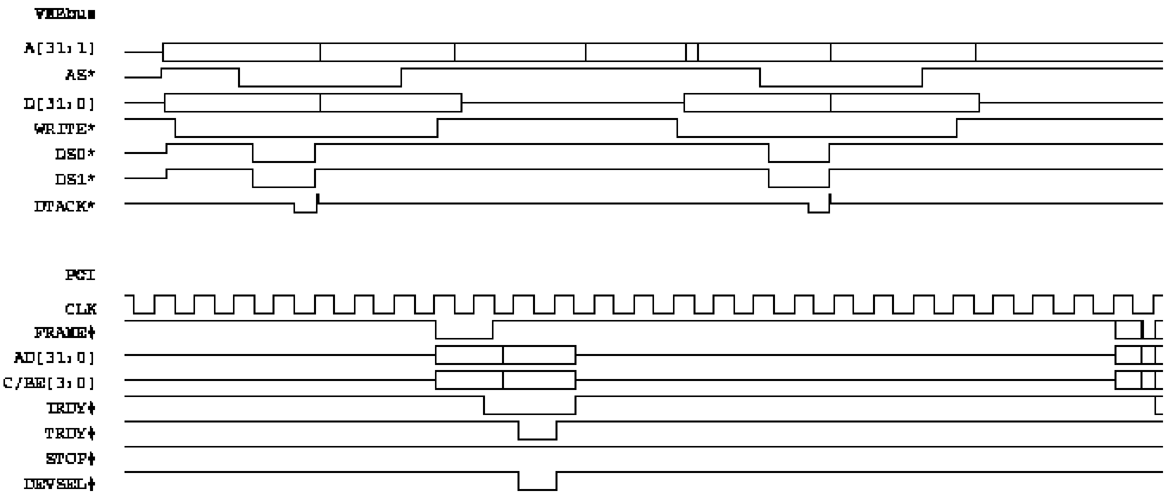
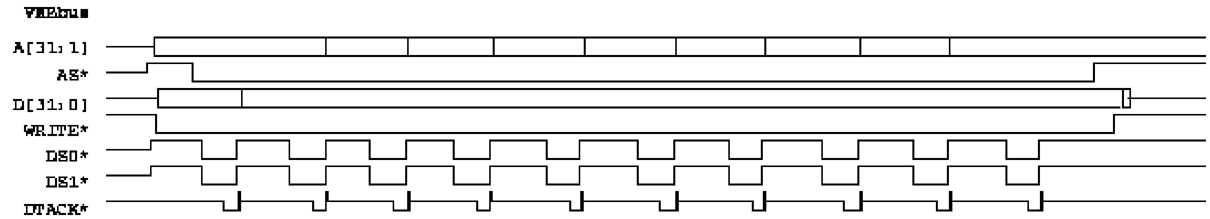


Figure C-5: BLT Decoupled Write - Universe II as VME Master



VME Slave Channel

Coupled Cycles

Block vs. non-Block Transfers

The Universe II VME Slave Channel handles both block and non-block coupled accesses in similar manners. Each data beat is translated to a single PCI transaction. Once the transaction has been acknowledged on the PCI bus, the Universe II asserts DTACK* to terminate the VME data beat.

A non-block transfer and the first beat of a BLT transfer have identical timing. In each, the Universe II decodes the access and then provides a response to the data beat. Subsequent data beats in the BLT transfer are shorter than the first due to the fact that no address decoding need be performed in these beats.

MBLT transfers behave somewhat differently. The first beat of an MBLT transfer is address only, and so the response is relatively fast. Subsequent data beats require acknowledgment from the PCI bus. With a 32-bit PCI bus, the MBLT data beat (64 bits of data) requires a two data beat PCI transaction. Because of this extra data beat required on the PCI bus, the slave response of the Universe II during coupled MBLT cycles is at least one PCI clock greater (depending upon the response from the PCI target) than that during BLT cycles.

Read Cycles

During coupled cycles, the Universe II does not acknowledge a VME transaction until it has been acknowledged on the PCI bus. Because of this the VME slave response during coupled reads is directly linked to the response time for the PCI target. Each clock of latency in the PCI target response translates directly to an extra clock of latency in the Universe II's VME coupled slave response.

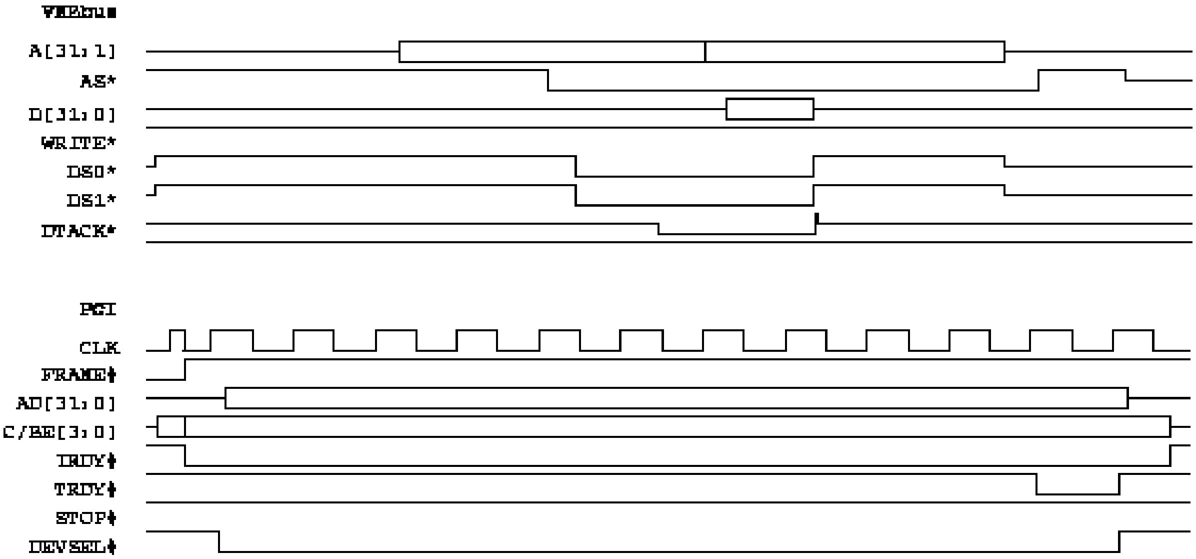
The address of an incoming VME transaction is decoded and translated to an equivalent PCI transaction. Typically, four PCI clock periods elapse between the initial assertion of AS* on the VMEbus and the assertion of REQ# on the PCI bus. During the data only portion of subsequent beats in block transfers, the time from DS* assertion to REQ# is about 4 clocks. If the PCI bus is parked at the Universe II, no REQ# is asserted and FRAME# is asserted 4 clocks after AS*.

From assertion of REQ#, the Universe II does not insert any extra wait states in its operations as an initiator on the PCI bus. Upon receiving GNT# asserted, the Universe II asserts FRAME# in the next clock and after the required turn-around phase, asserts IRDY# to begin data transfer.

Once TRDY# is sampled asserted, the Universe II responds back to the VMEbus by asserting DTACK*. If the initiating VME transaction is 64-bit and the PCI bus or PCI bus target are 32 bit, then two data transfers are required on PCI before the Universe II can respond with DTACK*. No wait states are inserted by the Universe II between these two data beats on PCI. The assertion of DTACK* from the assertion of TRDY# has a latency of 1 clock. Figure C-6 shows a typical non-block coupled read cycle.

When accessing a PCI target with a zero wait state response, the Universe II VME response becomes approximately 10 PCI clock periods (about 301ns in a 33MHz system) during single cycles, and the first beat of a BLT. During pure data beats in both BLT and MBLTs, the slave response becomes 8 clocks.

Figure C-6: Coupled Read Cycle - Universe II as VME Slave



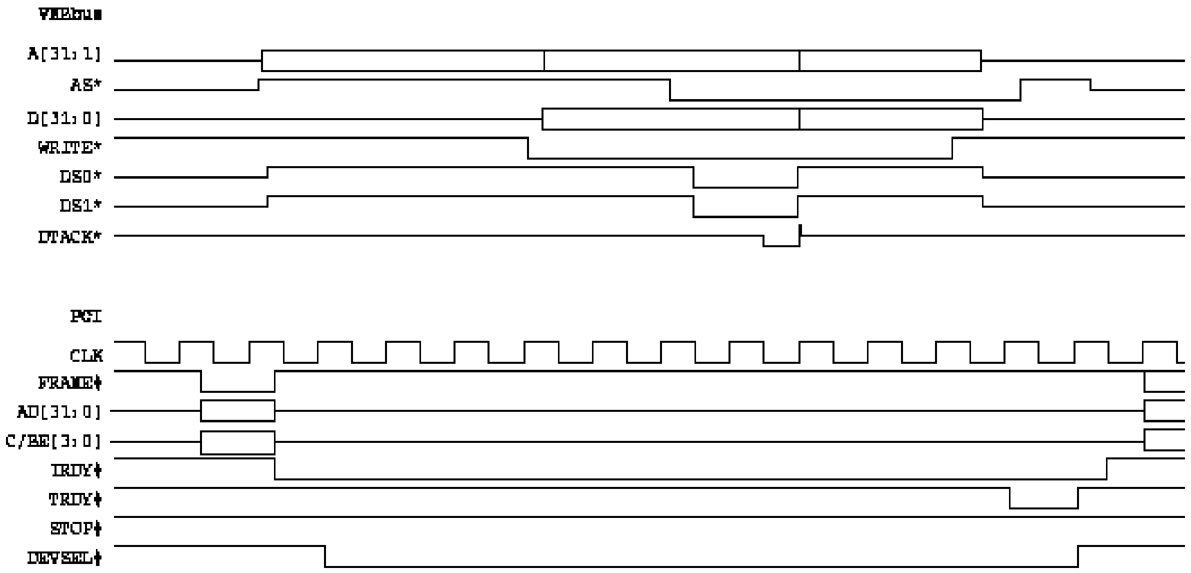
Write Cycles

Coupled writes in the VME Slave Channel operate in a similar fashion to the coupled reads. The VME slave response is directly linked to the response of the PCI target. In generating the request to the PCI bus, coupled write cycles require one further clock over reads. Hence, during single cycles, or the first beat of a BLT, the time from AS* to REQ# asserted is 3-4 PCI clocks, while DS* to REQ# is 3 clocks for the data beat portion of a block transfer. If the PCI bus is parked at the Universe II, REQ# is not asserted and the transaction begins immediately with assertion of FRAME#.

As with reads, the response from the PCI target’s assertion of TRDY# to DTACK* assertion by the Universe II adds one clock to the transfer. Figure C-7 shows a typical non-block coupled write cycle.

Because write cycles on the PCI bus require one less clock than reads, due to the absence of the turn-around phase between address and data phases, the overall slave response during coupled writes works out to the same as coupled reads against an identical target. In accessing a zero-wait state PCI target, the Universe II’s coupled write slave response then is approximately 10 PCI clocks. During subsequent data beats of a block transfer (either BLT or MBLT), the slave response (DS* to DTACK*) is 8 clocks.

Figure C-7: Coupled Write Cycle - Universe II as VME Slave (bus parked at Universe II)



Decoupled Cycles

Write Cycles

Effect of the PCI Aligned Burst Size

The PCI Aligned Burst Size (PABS in the MAST_CTL register) affects the maximum burst size that the Universe II generates onto the PCI bus; either 32, 64, or 128 bytes. Note that the VME Slave Channel only generates PCI bursts in response to incoming block transfers.

The greater burst size means less arbitration and addressing overhead. However, incumbent in this is the greater average latency for other devices in the PCI system. Hence, in the VME Slave Channel, the burst size is a trade-off between performance and latency.

VME Slave Response

As a VME slave, the Universe II accepts data into its RXFIFO with minimum delay provided there is room in the FIFO for a further data beat. Assertion of DTACK* is delayed if there is insufficient room in the FIFO for the next data beat.

During non-block transfers, the Universe II must both decode the address and enqueue the data before asserting DTACK* to acknowledge the transfer. Because of this, the slave response during non-block transfers is considerably slower than block transfers. This slave response time is 127ns.

During BLT transfers, the slave response in the first data beat being both address decode and data transfer is the same as a non-block transfer, i.e., 127ns. Subsequent data beats, however, are much faster. Response time for these is 50 to 56ns.

During MBLT transfers, the first phase is address only and the slave response is 127ns. Subsequent phases are data only and so the slave response is the same as with BLTs i.e., 50 to 56ns.

Note that the slave response is independent of the data size. D16 non-block transfers have a slave response identical to D32. BLT data beats have slave responses identical to MBLT data beats.

Figure C-8: Non-Block Decoupled Write Cycle - Universe II as VME Slave

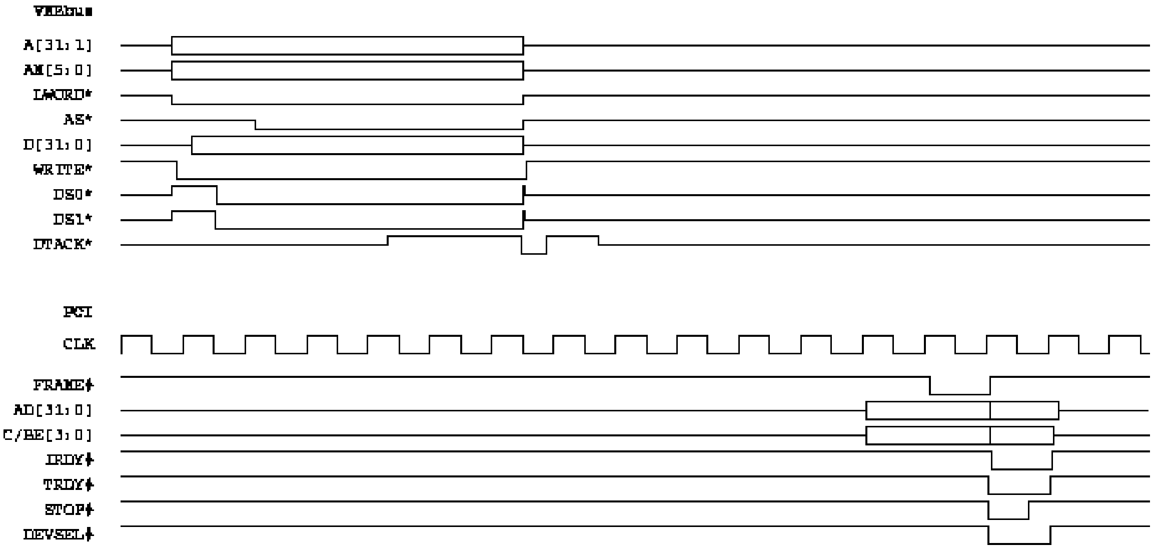


Figure C-9: BLT Decoupled Write Cycle - Universe II as VME Slave

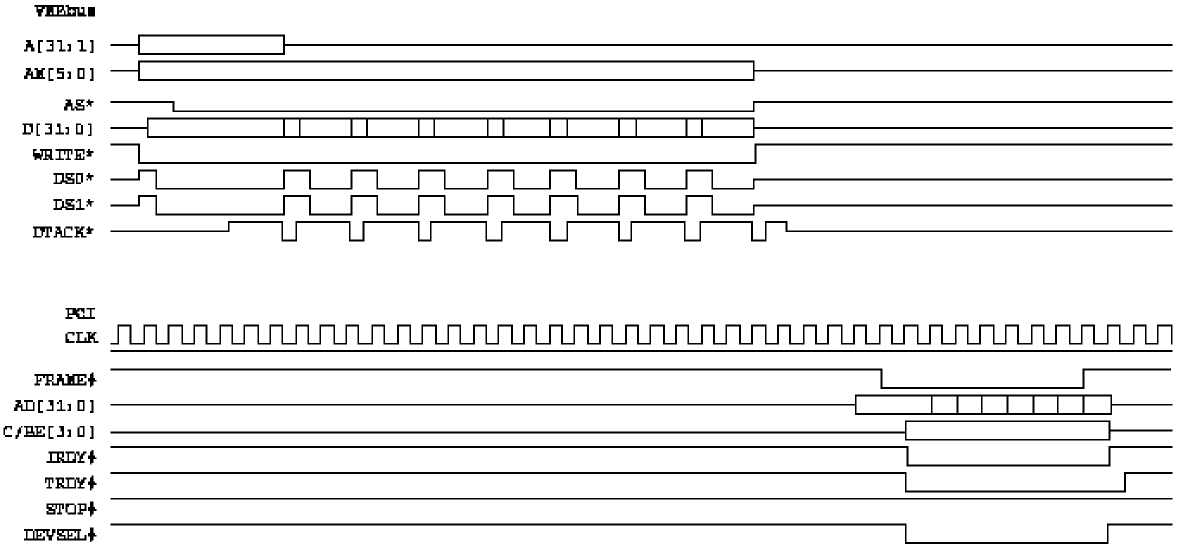
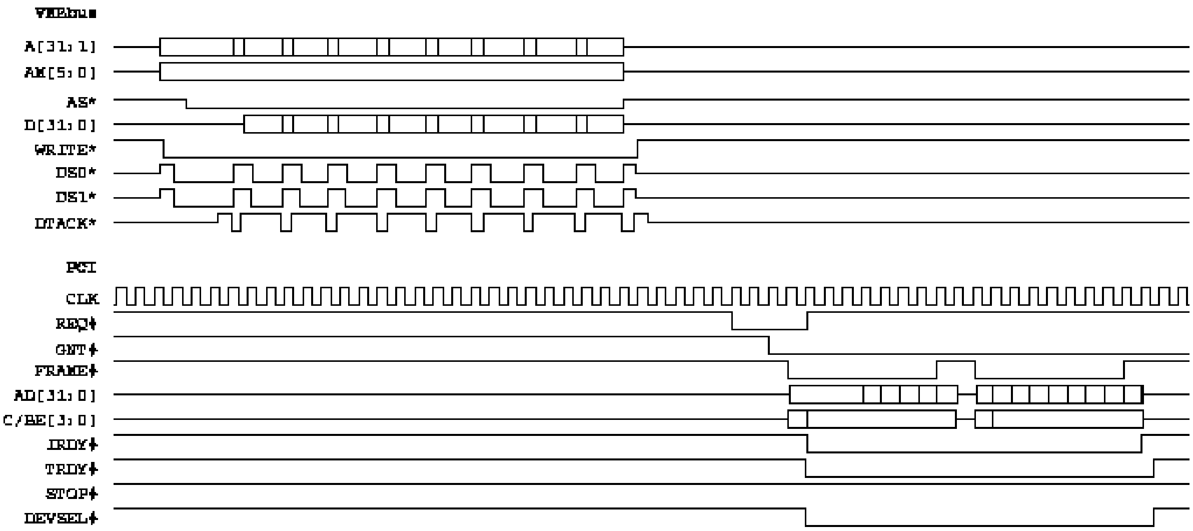


Figure C-10: MBLT Decoupled Write Cycle - Universe II as VME Slave



PCI Master Performance

The Universe II supports bus parking. If the Universe II requires the PCI bus it will assert REQ# only if its GNT# is not currently asserted. When the PCI Master Module is ready to begin a transaction and its GNT# is asserted, the transfer begins immediately. This eliminates a possible one clock cycle delay before beginning a transaction on the PCI bus which would exist if the Universe II did not implement bus parking. Bus parking is described in Section 3.4.3 of the PCI Specification (Rev. 2.1).

On the PCI bus, the Universe II dequeues data from the RXFIFO once a complete VME transaction has been enqueued or once sufficient data has been enqueued to form a PCI transaction of length defined by the PABS field.

Since the Universe II does not perform any address phase deletion, non-block transfers are dequeued from the RXFIFO as single data beat transactions. Only block transfers result in multi-data beat PCI transactions; typically 8, 16 or 32 data beats. In either case, the Universe II does not insert any wait states as a PCI master. The clock, after the bus has been granted to the Universe II, drives out FRAME# to generate the address phase. The data phases begin immediately on the next clock. If there is more than one data phase, each phase will immediately follow the acknowledgment of the previous phase.

In each case, because of the lack of any wait states as a PCI master, the Universe II is consuming the minimum possible bandwidth on the PCI bus, and data will be written to the PCI bus at an average sustained rate equal to the rate at which the VME master is capable of writing it.

The sustained performance on the PCI bus performing single data beat write transactions to a 32-bit PCI bus is 15 MB/s; double this for a 64-bit bus. When performing 32-byte transactions the sustained performance increases to 106 MB/s, 120 MB/s with 64-byte transactions. Again, these can be doubled for a 64-bit PCI bus. Bear in mind that the PCI bus can only dequeue data as fast as it is being enqueued on the VMEbus. Hence, as the RXFIFO empties, the sustained performance on the PCI will drop down to match the lower performance on the VME side. However, even with the decreased sustained performance, the consumed bandwidth will remain constant (no extra wait states are inserted while the Universe II is master of the PCI bus.)

These numbers assume the PCI bus is granted to the Universe II immediately and that the writes are to a zero-wait state PCI target capable of accepting the full burst length. Figure C-2 through Figure C-10 show the Universe II responding to non-block, BLT and MBLT write transactions to a 32-bit PCI bus. Even better performance is obtained with PCI bus parking.

Prefetched Read Cycles

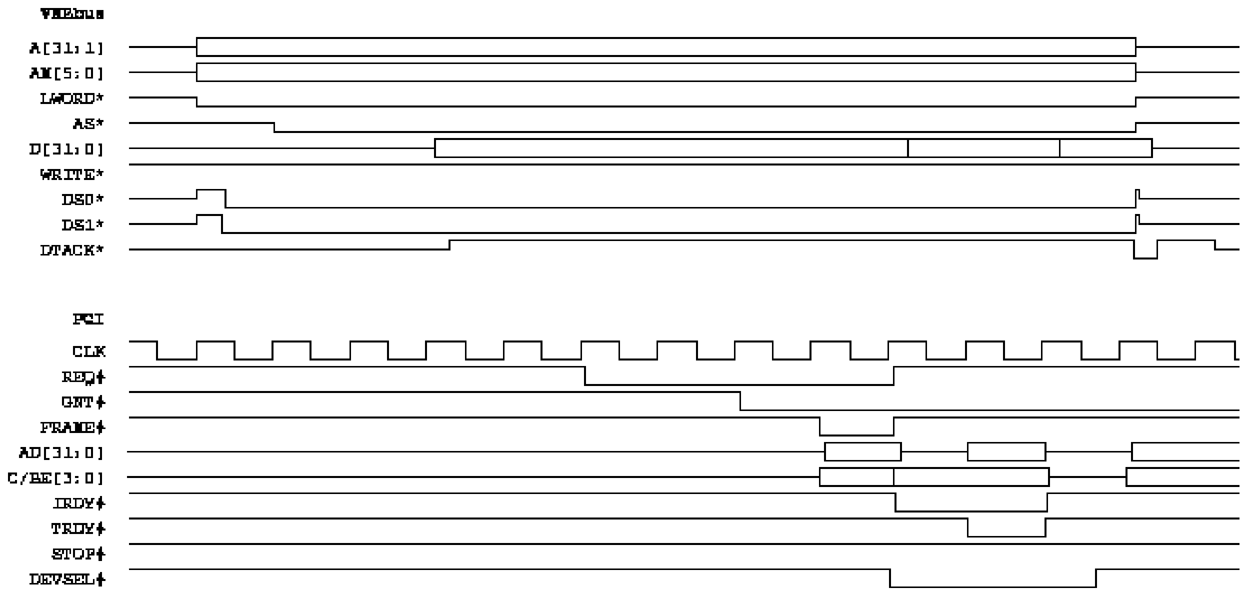
To minimize its slave response, the Universe II generates prefetched reads to the PCI bus in response to BLT and MBLT reads coming in from the VMEbus. This option must first be enabled on a per image basis.

When enabled, the Universe II will respond to a block read by performing burst reads on the PCI bus of length defined by the PCI Aligned Burst Size (PABS in the MAST_CTL register). These burst reads continue while the block transfer is still active on the VMEbus (AS* not negated) and there is room in the RDFIFO. If there is insufficient room in the RDFIFO to continue (a common occurrence since the Universe II is capable of fetching data from the PCI bus at a much faster rate than a VME master is capable of receiving it), then pre-fetching stops and only continues once enough room exists in the RDFIFO for another full burst size.

The first data beat of a block transfer must wait for the first data beat to be retrieved from the PCI bus—this is essentially a coupled transfer. See the section on coupled transfers for details on coupled performance. However, once the pre-fetching begins, data is provided by the Universe II in subsequent data beats with a slave response of 57ns. This continues while there is data in the RDFIFO. If the RDFIFO empties because data is being fetched from the PCI bus too slowly, wait states are inserted on the VMEbus awaiting the enqueueing of more data.

On the PCI bus, the Universe II fetches data at 89 MB/s with PABS set to 32-byte transactions, 106 MB/s when set to 64-byte transactions. Even better performance is obtained if PABS is set for 128-byte transactions. Once the RDFIFO fills, pre-fetching slows to match the rate at which it is being read by the external VMEbus master. Bandwidth consumption, however, remains constant, only the idle time between transactions increases.

Figure C-11: BLT Pre-fetched Read Cycle - Universe II as VME Slave



DMA Channel

Relative FIFO sizes

Two fixed “watermarks” in the DMA Channel control the Universe’s II requisition of the PCI bus and VMEbus. The DMAFIFO PCI Watermark is 128 bytes. This means that during reads from the PCI bus, the Universe II will wait for 128 bytes to be free in the DMAFIFO before requesting the PCI bus. For PCI writes, the Universe II waits for 128 bytes of data to be in the FIFO before requesting the PCI bus. The DMAFIFO VMEbus watermark is 64 bytes. This means that during reads from the VMEbus, the Universe II will wait for 64 bytes to be free in the DMAFIFO before requesting the Vmebus. For VMEbus writes, the Universe II waits for 64 bytes of data to be in the FIFO before requesting the VMEbus.

These watermarks have been tailored for the relative speeds of each bus, and provide near optimal use of the DMA channel.

VMEbus Ownership Modes

The DMA has two counters that control its access to the VMEbus: the VON (VMEbus On) counter and the VOFF (VMEbus Off) timer. The VON counter controls the number of bytes that are transferred by the DMA during any VMEbus tenure, while the VOFF timer controls the period before the next request after a VON time-out.

While the bus is more optimally shared between various masters in the system, and average latency drops as the value programmed for the VON counter drops, the sustained performance of the DMA also drops. The DMA is typically limited by its performance on the VMEbus. As this drops off with greater re-arbitration cycles, the average VMEbus throughput will drop. Even if the Universe II is programmed for ROR mode, and no other channels or masters are requesting the bus, there will be a period of time during which the DMA will pause its transfers on the bus, due to the VON counter expiring.

An important point to consider when programming these timers is the more often the DMA relinquishes its ownership of the bus, the more frequently the PCI Slave Channel will have access to the VMEbus. If DMA tenure is too long, the TXFIFO may fill up causing any further accesses to the bus to be retried. In the same fashion, all coupled accesses will be retried while the DMA has tenure on the bus. This can significantly affect transfer latency and should be considered when calculating the overall system latency.

VME Transfers

On the VMEbus, the Universe II can perform D08 through D64 transactions in either block or non-block mode. The time to perform a single beat, however, is independent of the bus width being used. Hence, a D08 transaction will transfer data at 25% the rate of a D32, which in turn is half that for D64.

There is a significant difference between the performance for block vs. non-block operations. Because of the extra addressing required for each data transfer in non-block operations, the DMA performance is about half that compared to operating in block mode. Moreover, considering that most VME slaves respond less quickly in non-block mode, the overall performance may drop to one-quarter of that achievable in block mode.

When programmed for Release-When-Done operation, the Universe II will perform an early release of BBSY* when the VON counter reaches its programmed limit. This gives other masters a chance to use the VMEbus (and possibly access the VME Slave Channel), but may decrease performance of the DMA Channel; this factor may also play in favor of the DMA Channel, by pausing the PCI Target Channel's use of the VMEbus.

Read Transfers

When performing non-block reads on the VMEbus, the Universe II cycle time (AS* to next AS*) is approximately 209ns, which translates to about 20 MB/s when performing D32 transfers. For block transfers the cycle time (DS* to next DS*) falls to about 156ns, or 25 MB/s for D32 transfers. For multiplexed block transfers (MBLTs) the cycle time remains the same, but because the data width doubles, the transfer rate increases to about 50MB/s.

Write Transfers

Non-block writes to the VMEbus occur at 180ns cycle time (AS* to next AS*), or 23MB/s during D32 transfers. Block writes, however, are significantly faster with a 116ns cycle time (DS* to next DS*), or 36 MB/s. Multiplexed block transfers have slightly longer cycle times at about 112ns (DS* to next DS*), or 62 MB/s with D64 MBLTs.

PCI Transfers

As a master on the PCI bus, the Universe II DMA follows the same general set of rules as the VME Slave channel does: it never inserts any wait states into the transfer (i.e., it never negates IRDY# until the transaction is complete) and will whenever possible, generate full aligned bursts as set in the PABS field of the MAST_CTL register.

Between transactions on the PCI bus, the Universe II DMA typically sits idle for 6 clocks. Hence, minimizing the number of idle periods and re-arbitration times by setting PABS to its maximum value of 128 bytes may increase the performance of the DMA on this bus. Higher PABS values imply that the Universe II will hold on to both the PCI bus and the VMEbus for longer periods of time. The reason that PABS also may impact on VMEbus tenure is that (in the case of PCI writes), the DMA FIFO is less likely to fill, and (in the case of PCI reads) the DMA is less likely to go empty. However, given the relative speeds of the buses, and the relative watermarks, the effect of PABS on VMEbus utilization is not as significant as its effects on the PCI bus.

While higher values of PABS increase DMA throughput, they may increase system latency. That is, there will be a longer latency for other PCI transactions, including possible transactions coming through the VME Slave Channel (since the DMA channel will own the PCI bus for longer periods of time). Also, accesses between other PCI peripherals will, on average, have a longer wait before being allowed to perform their transactions. PCI latency must be traded off against possible DMA performance.

Although both read and write transactions occur on the PCI bus with zero wait states, there is a period of six PCI clocks during which the Universe II remains idle before re-requesting the bus for the next transaction. PCI bus parking may be used to eliminate the need for re-arbitration.

With PABS set for 32-byte transactions on a 32-bit PCI bus, this translates to a peak transfer rate of 97 MB/s for reads (including pre-fetching), 98 MB/s for writes, doubling to 194 and 196 for a 64-bit PCI bus. With PABS set for 64-byte transactions, the peak transfer rate increases to 118 MB/s for reads, 125 MB/s for writes on a 32-bit PCI bus—236 MB/s and 250 MB/s respectively for 64-bit PCI buses. The numbers for writes to PCI assume that data are read from VME using BLTs.

Figure C-12: PCI Read Transactions During DMA Operation

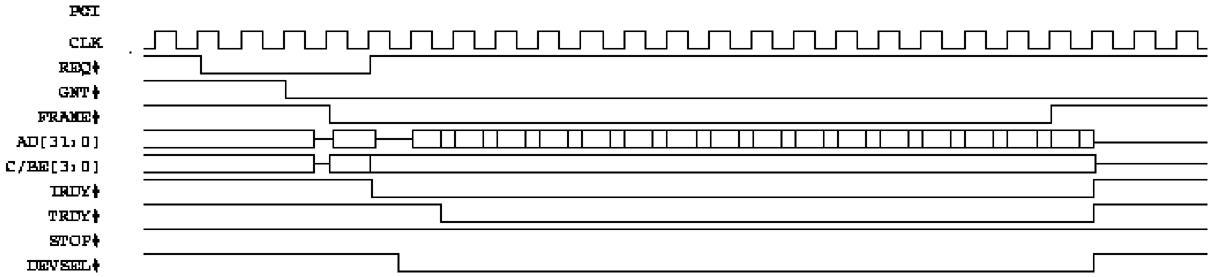
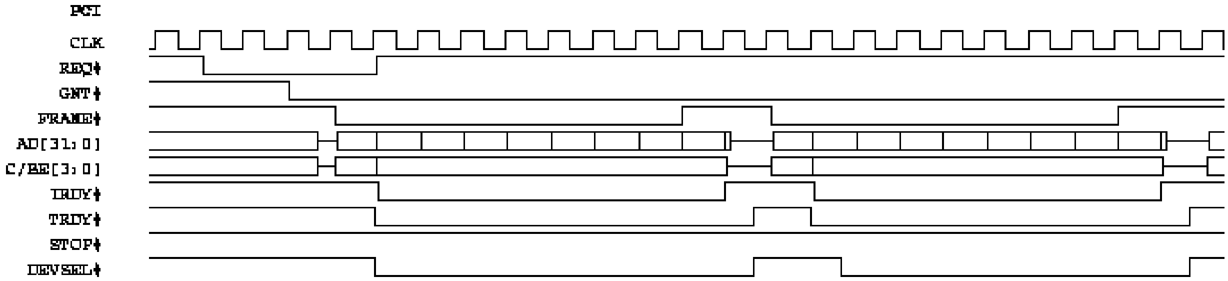


Figure C-13: Multiple PCI Read Transactions During DMA Operation



Summary

Table C-1: PCI Slave Channel Performance

Cycle Type	Performance
Coupled Read - PCI target response	8 PCI clocks
Coupled Write - PCI target response	9 PCI clocks
Decoupled Write	
- non-block D32	
- VME cycle time	180 ns
- sustained perf (32-byte PABS)	23 MB/s
- sustained perf (64-byte PABS)	43 MB/s
- D32 BLT	
- VME cycle time	119 ns
- sustained perf (32-byte PABS)	32 MB/s
- sustained perf (64-byte PABS)	35 MB/s
- D64 MBLT	
- VME cycle time	119 ns
- sustained perf (32-byte PABS)	53 MB/s
- sustained perf (64-byte PABS)	59 MB/s

Table C-2: VME Slave Channel Performance

Cycle Type	Performance
	VME Slave Response (ns)
Coupled Read	
- non-block	301
- D32 BLT	293
- D64 BLT	322
Coupled Write	
- non-block	278
- D32 BLT	264
- D64 BLT	292
Pre-fetched Read	
- VME slave response (1st data beat)	293
- VME slave response (other data beats)	57
Decoupled Write	
- non-block slave response	127
- block slave response (1st data beat)	127
- block slave response (other data beats)	50

Table C-3: DMA Channel Performance

Cycle Type	Performance MB/s
PCI Reads	
- 32-byte PABS	97 (194)[a]
- 64byte PABS	118 (236)
PCI Writes	
- 32-byte PABS	98 (196)
- 64byte PABS	125 (250)
VME Reads	
- non-block D32	18
- D32 BLT	22
- D64 MBLT	45
VME Writes	
- non-block D32	22
- D32 BLT	32
- D64 MBLT	65

a.64-bit PCI performance in brackets.

Appendix D Typical Applications

VME Interface

Being a bridge between standard interfaces, the Universe II requires minimal external logic to interface to either the VMEbus or to the PCI bus. In most applications, only transceivers to buffer the Universe II from the VMEbus, plus some reset logic are all that is required. The following information should be used only as a guide in designing the Universe II into a PCI/VME application. Each application will have its own set-up requirements.

Transceivers

The Universe II has been designed such that it requires full buffering from VMEbus signals. Necessary drive current to the VMEbus is provided by the transceivers while at the same time isolating the Universe II from potentially noisy VMEbus backplanes. In particular, complete isolation of the Universe II from the VMEbus backplane allows use of ETL transceivers which provide high noise immunity as well as use in live insertion environments. The VME community has recently standardized “VME64 Extensions” (ANSI VITA 1.1) which among other new VME features, facilitates live insertion environments.

If neither live insertion nor noise immunity are a concern, those buffers that provide input only (U15 and U17 in Figure D-1) may be omitted. The daisy chain input signals, BGIN[3:0] and IACKIN, have Schmitt trigger inputs, which should rectify any minor noise on these signals. If considerable noise is expected, the designer may wish to put external filters on these signals. Bear in mind that any filtering done on these signals will detrimentally affect the propagation of bus grants down the daisy chain. Only extremely noisy systems or poorly designed backplanes should require these filters.

Figure D-1 shows one example of how to connect the Universe II to the VMEbus. The transceivers in this example were chosen to meet the following criteria:

- provide sufficient drive strength as required by the VME specification (see Table D-1)
- meet Universe II skew requirements
- minimize part counts

U15 and U17 in Figure D-1 are optional devices. They will provide better noise immunity.

The Universe II, with the addition of external transceivers, is designed to meet the timing requirements of the VME specification. Refer to the VME64 specification (ANSI VITA 1.0) for details on the VME timing. In order to meet the requirements outlined in this specification, the external transceivers must meet certain characteristics as outlined in Table D-2.

Table D-1: VMEbus Signal Drive Strength Requirements

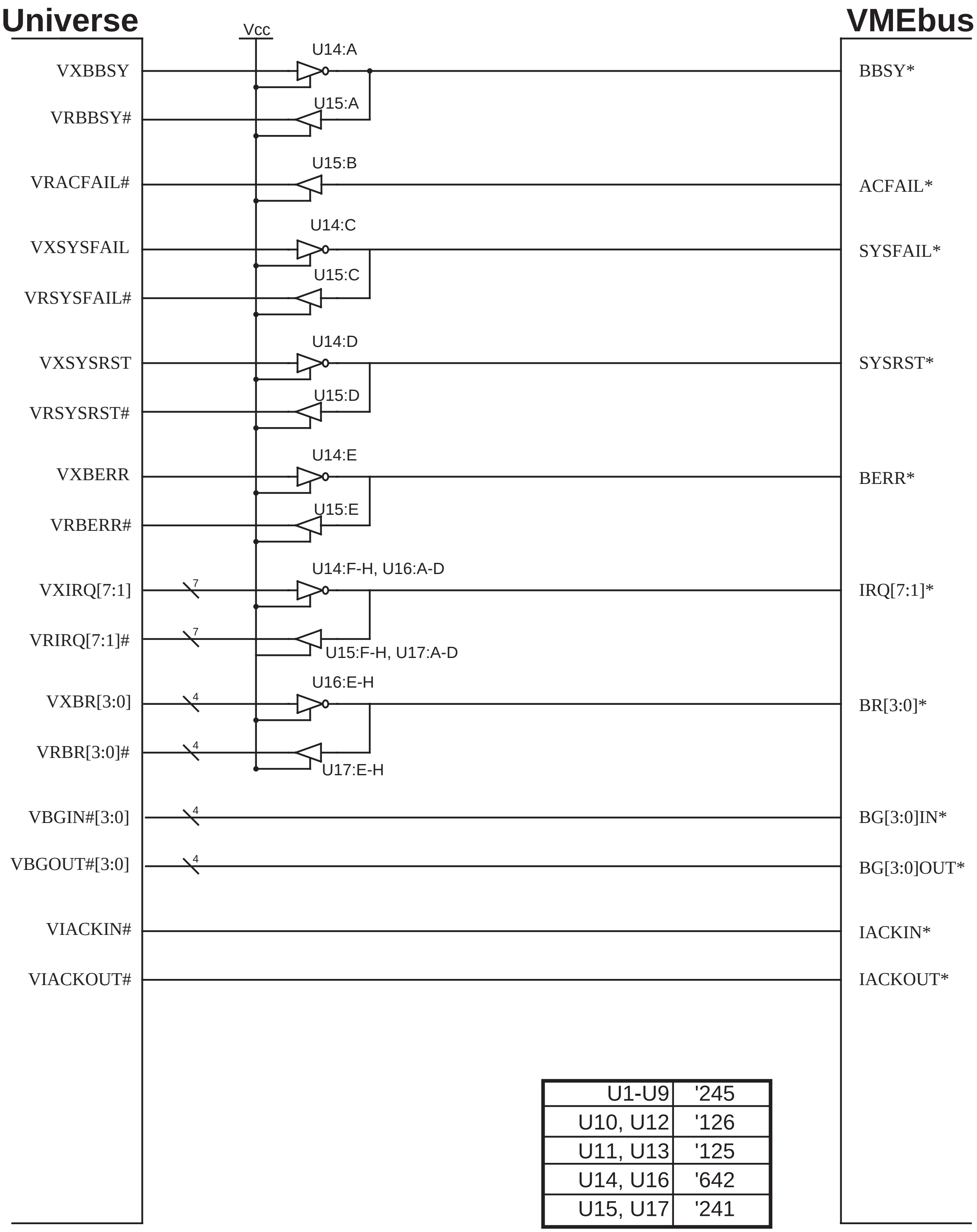
VME bus Signal	Required Drive Strength
A[31:1], D[31:0], AM[5:0], IACK*, LWORD*, WRITE*, DTACK*	IOL ≥ 48mA IOH ≥ 3mA
AS*, DS[1:0]*,	IOL ≥ 64mA IOH ≥ 3mA
SYSCLK*	IOL ≥ 64mA IOH ≥ 3mA
BR[3:0]*, BSY*, IRQ[7:0]*, BERR*, SYSFAIL*, SYSRESET*	IOL ≥ 48mA

Table D-2: VMEbus Transceiver Requirements

Parameter	From (Input)	To (Output)	Timing	
			Min	Max
VA, VD, VAM, VIACK, VLWORD, VWRITE, VAS, VDSx, VDTACK[a]				
skew (pkg to pkg)	A	B		8 ns
skew (pkg to pkg)	B	A		4 ns
^{t}Prop	DIR	A	1 ns	5ns
^{t}Prop	DIR	B	2 ns	10ns
Cin	A			25 pf

There are no limits on propagation delay or skew on the remaining buffered VME signals: VSYSCLK, VBCLR, VXBBSY, VRBBSY, VRACFAIL, VXSYSFAIL, VRSYSFAIL, VXSYSRST, VRSYSRST, VXBERR, VRBERR, VXIRQ, VRIRQ, VXBR, VRBR.

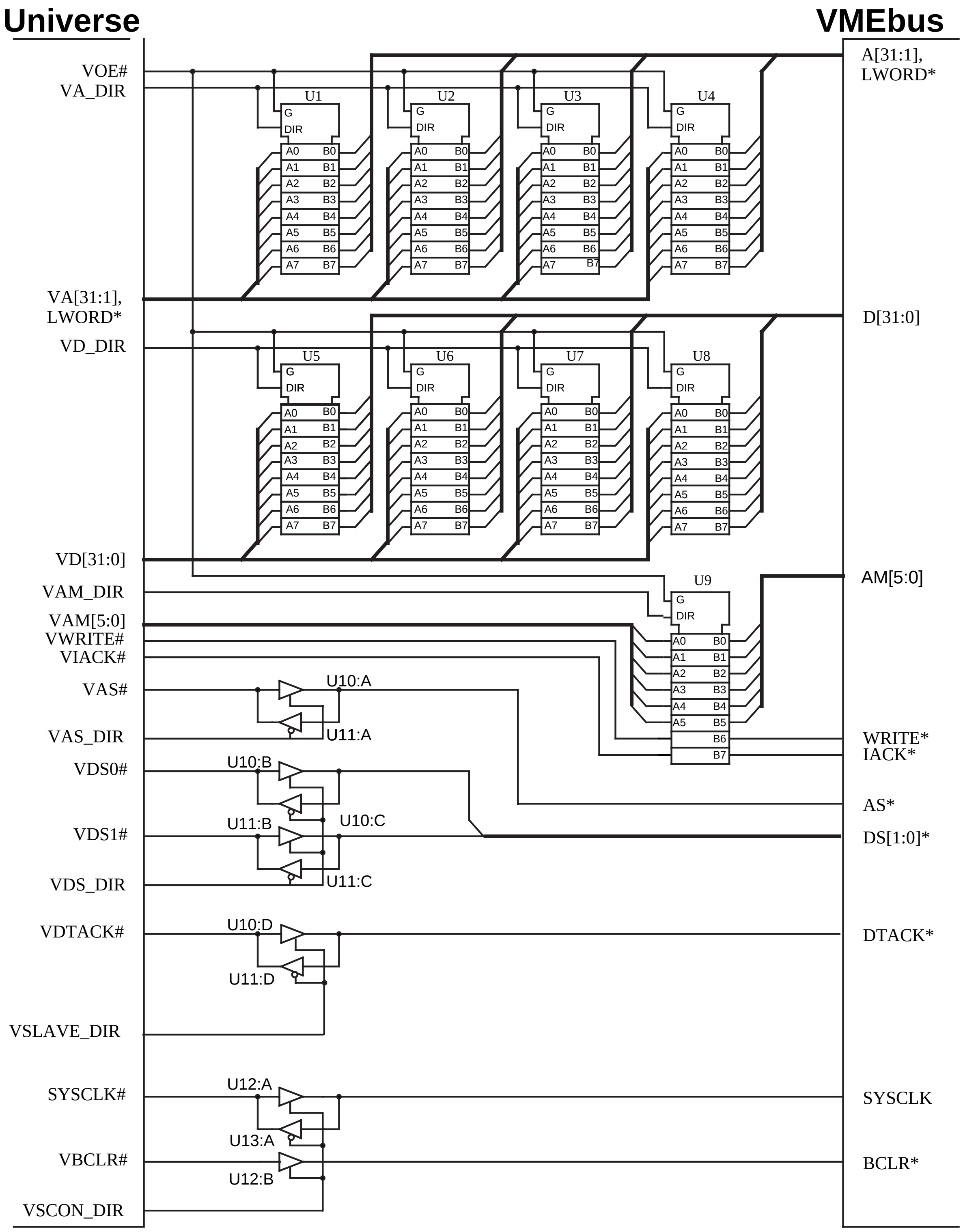
FigureD-1: Universal II Connections to the VMEbus Through TTL Buffers



U1-U9	'245
U10, U12	'126
U11, U13	'125
U14, U16	'642
U15, U17	'241

Note: U15 & U17 are optional

Figure D-1: Universe II Connections to the VMEbus Through TTL Buffers (continued)



F Series transceivers meet the requirements specified in Table D-1 and Table D-2. A faster family such as ABT, may also be used. Care should be taken in the choice of transceivers to avoid ground bounces and also to minimize crosstalk incurred during switching. To limit the effects of crosstalk, the amount of routing under these transceivers must be kept to a minimum. Daisy chain signals can be especially susceptible to crosstalk.

Should the designer wish to put any further circuitry between the Universe II and the VMEbus, that circuitry must meet the same timing requirements as the transceivers in order for the combined circuit to remain compliant with the VME64 specification.

Pull-down resistors

The Universe II has internal pull-down resistors which are used for its default power-up option state. The pins requiring pull-ups or pull-downs are indicated in Table 8-2. (Note that REQ64# has an internal pull-up.) These internal pull-down resistors, ranging from 25kW-500kW, are designed to sink between 10µA-200µA. F-series buffers, however, can source up to 650 µA of current (worst case). This sourced current has the ability to override the internal power up resistors on the Universe II. This may cause the Universe II to incorrectly sample a logic “1” on the pins. To counteract this potential problem, assuming a worst case scenario of a 650 µA current, Tundra recommends connecting a 1K resistor to ground, in parallel, with the internal pull-down resistor.

Tundra recommends that any pins controlling the power-up options which are critical to the application at powerup be connected to ground with a pull-down resistor as described above. If these options are not critical and if it is possible to reprogram these options after reset, additional resistors need not be added.

Direction control

When the Universe II is driving VMEbus lines, it drives the direction control signals high (i.e., VA_DIR, VAM_DIR, VAS_DIR, VD_DIR, VDS_DIR, VSLAVE_DIR, and VSCON_DIR). When the VMEbus is driving the Universe II, these signals are driven low. The control signals in the Universe II do not all have the same functionality. Since the Universe II implements early bus release, VAS_DIR must be a separate control signal.

Contention between the Universe II and the VME buffers is handled since the Universe II tristates its outputs one 64MHz clock period before the buffer direction control is faced inwards.

Power-up Options

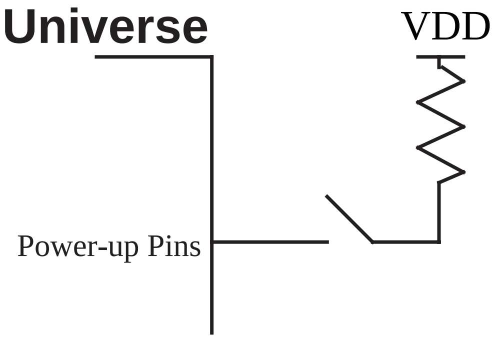
Power-up options for the automatic configuration of slave images and other Universe II features are provided through the state of the VME address and data pins, VA[31:1] and VD[31:27]. All of these signals are provided with internal pull-downs to bias these signals to their default conditions. Should values other than the defaults be required here, either pull-ups or active circuitry may be applied to these signals to provide alternate configurations. Power-up options are described in Power-Up Options in Chapter 2.

Since the power-up configurations lie on pins that may be driven by the Universe II or by the VME transceivers, care must be taken to ensure that there is no conflict. During any reset event, the

Universe II does not drive the VA or VD signals. As well, during any VMEbus reset (SYSRST*) and for several CLK64 periods after, the Universe II negates VOE# to tri-state the transceivers. During the period that these signals are tri-stated, the power-up options are loaded with their values latched on the rising edge of PWRRST#.

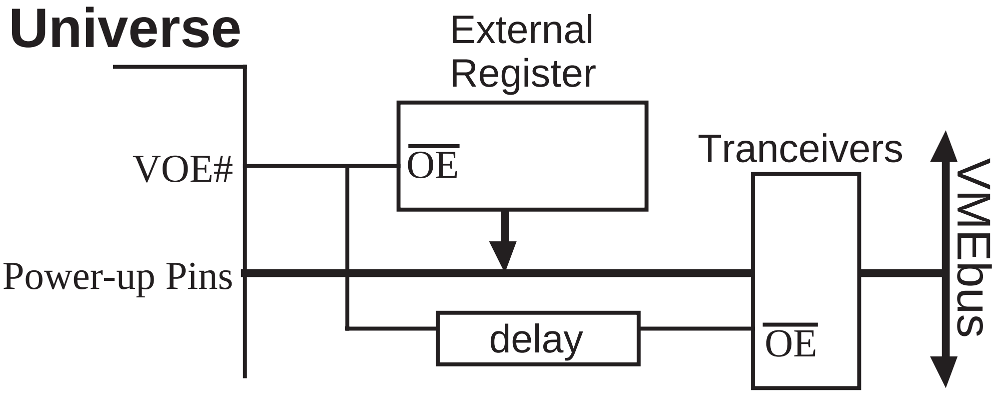
Configuration of power-up options is most easily accomplished through passive 10k pull-up resistors on the appropriate VA and VD pins. The configurations may be made user-configurable through jumpers or switches as shown in Figure D-2.

Figure D-2: Power-up Configuration Using Passive Pull-ups



Alternatively, an active circuit may be designed which drives the VA and VD pins with pre-set (or pre-programmed) values. This sort of circuit would be of value when power-up configurations such as the register access slave image are stored in an external programmable register. To implement this circuit, the VOE# output from the Universe II must be monitored. When the Universe II negates this signal, the appropriate VA and VD signals may be driven and upon re-assertion the drive must be removed. To avoid conflict with the transceivers, logic must be designed such that the enabling of the transceivers does not occur until some point after the configuration options have been removed from the VD and VA signals. Figure D-3 shows one such implementation. The delay for enabling of the VMEbus transceivers could be implemented though clocked latches.

Figure D-3: Power-up Configuration Using Active Circuitry



Auto-Syscon and PCI Bus Width Power-up Options

The VME64 specification provides for automatic enabling of the system controller in a VME system through monitoring of the BGIN3* signal. If at the end of SYSRST* this pin is low, then the system controller is enabled; otherwise it is disabled. The Universe II provides an internal pull-down resistor for this function. If it is in slot one, this pin will be sampled low. If not in slot one, then it will be driven high by the previous board in the system and system controller functions will be disabled. No external logic is required to implement this feature.

PCI Bus Interface

The Universe II provides a fully standard PCI bus interface compliant for both 32-bit and 64-bit designs. No external transceivers or glue logic is required in interfacing the Universe II to any other PCI compliant devices. All signals may be routed directly to those devices.

The Universe II's PCI interface can be used as a 32-bit bus or 64-bit bus. If used as a 32-bit interface, the 64-bit pins, AD[32:63] and ACK64# are left unterminated. On a 32-bit PCI bus, the Universe II drives all its 64-bit extension bi-direct signals (C/BE[7:4]#, AD[63:32], REQ64#, PAR64 and ACK64#) at all times to unknown values. Independent of the setting of the LD64EN bit, the Universe II will never attempt a 64-bit cycle on the PCI bus if it is powered up as 32-bit.

REQ64# must be pulled-down (with a 4.7kW resistor) at reset for 64-bit PCI (see PCI Bus Width on page 164). There is an internal pull-up on this pin which causes the Universe II to default to 32-bit PCI. This power-up option provides the necessary information to the Universe II so that these unused pins may be left unterminated.

Resets

The Universe II provides several reset input and outputs which are asserted under various conditions. These can be grouped into three types as shown in Table D-3.

Table D-3: Reset Signals

Group	Signal Name	Direction
VMEbus	VXSYSRST	output
	VRSYSRST#	input
PCI bus	LRST#	output
	RST#	input
	VME_RESET#	input
Power-up	PWRRST#	input

VMEbus Resets

The VMEbus resets are connected to the VMEbus as indicated in Figure D-1 on page 352 through external buffers.

PCI bus Resets

Use of the PCI bus resets will be application dependent. The RST# input to the Universe II should typically be tied in some fashion to the PCI bus reset signal of the same name. This will ensure that all Universe II PCI related functions are reset together with the PCI bus.

The LRST# pin is a totem-pole output which is asserted due to any of the following initiators:

- PWRRST#,
- VRSYSRST#,
- local software reset (in the MISC_CTL register), or
- VME CSR reset (in the VCSR_SET register).

The designer may wish to disallow the Universe II from resetting the PCI bus in which case this output may be left unconnected. Otherwise LRST# should be grouped with other PCI reset generators to assert the RST# signal such that:

RST# = LRST# & reset_source1 & reset_source2 &...

If the Universe II is the only initiator of PCI reset, LRST# may be directly connected to RST#.

Assertion of VME_RESET causes the Universe II to assert VXSYSRST.

Caution

This signal must not by tied to the PCI RST# signal unless the Universe II LRST# output will not generate a PCI bus reset. Connecting both LRST# and VME_RESET# to RST# will cause a feedback loop on the reset circuitry forcing the entire system into a endless reset.

To reset the VMEbus through this signal it is recommended that it be asserted for several clock cycles, until the Universe II asserts RST#, and then released. This ensures a break is made in the feedback path.

Power-Up Reset

The PWRRST# input is used to provide reset to the Universe II until the power supply has reached a stable level. It should be held asserted for 100 milliseconds after power is stable. Typically this can be achieved through a resistor/capacitor combination although more accurate solutions using under voltage sensing circuits (e.g. MC34064) are often implemented. The power-up options are latched on the rising edge of PWRRST#.

JTAG Reset

The JTAG reset, TRST#, should be tied into the master system JTAG controller. It resets the Universe II internal JTAG controller. If JTAG is not being used, this pin should be tied to ground.

Local Interrupts

The Universe II provides eight local bus interrupts, only one of which has drive strength that is fully PCI compliant. If any of the other seven interrupts are to be used as interrupt outputs to the local bus (all eight may be defined as either input or output), an analysis must be done on the design to determine whether the 4 mA of drive that the Universe II provides on these lines is sufficient for the design. If more drive is required, the lines may simply be buffered.

All Universe II interrupts are initially defined as inputs. To prevent excess power dissipation, any interrupts defined as inputs should always be driven to either high or low. Pull-ups should be used for this purpose rather than direct drive since a mis-programming of the interrupt registers may cause the local interrupts to be configured as outputs and potentially damage the device.

Manufacturing Test Pins

The Universe II has several signals used for manufacturing test purposes. They are listed in Table 2-27 along with the source to which they should be tied.

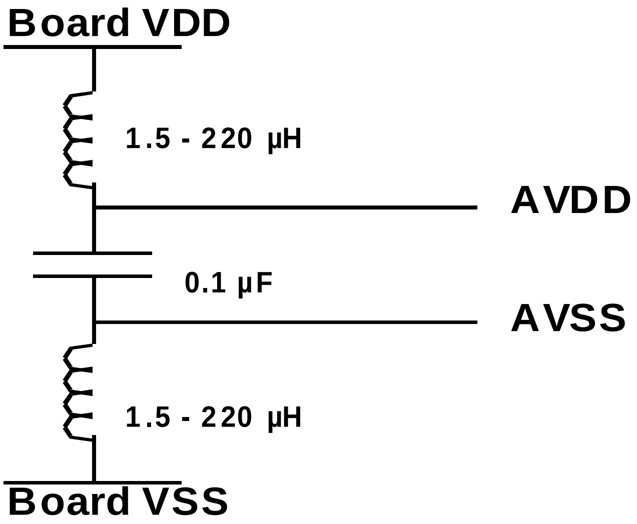
Decoupling V_{DD} and V_{SS} on the Universe II

This section is intended to be a guide for decoupling the power and ground pins on the Universe II. A separate analog power and ground plane is not required to provide power to the analog portion of the Universe II. However, to ensure a jitter free PLL operation, the analog AV_{DD} and AV_{ss} pins must be noise free. The following are recommended solutions for noise free PLL operation. The design could implement one of these solutions, but not both.

The Analog Isolation Scheme consists of the following:

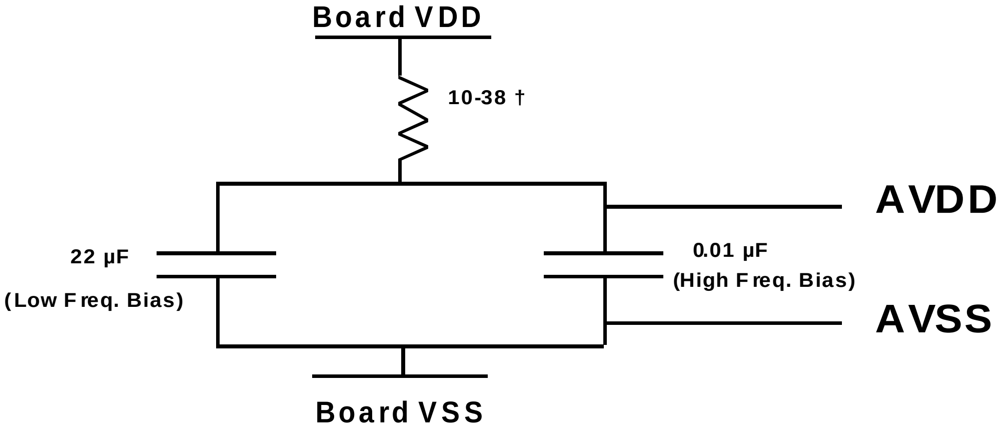
- a 0.1µF capacitor between the AV_{DD} and AV_{ss} pins, and
- corresponding inductors between the pins and the board power and ground planes (See Figure D-4). These inductors are not necessary, but they are recommended.

Figure D-4: Analog Isolation Scheme



The Noise Filter Scheme filters out the noise using two capacitors to filter high and low frequencies (See Figure D-5).

Figure D-5: Noise Filter Scheme



For both schemes, it is recommended that the components involved be tied as close as possible to the associated analog pins.

In addition to the decoupling schemes shown above, it is recommended that 0.1µF bypass capacitors should be tied between every three pairs of V_{DD} pins and the board ground plane. These bypass capacitors should also be tied as close as possible to the package.

Appendix E Reliability Prediction

Introduction

This section is designed to help the user to estimate the inherent reliability of the Universe II, as based on the requirements of MIL HDBK217F. This information is recommended for personnel who are familiar with the methods and limitations contained in MIL HDBK217F. The information serves as a guide only; meaningful results will be obtained only through careful consideration of the device, its operating environment, and its application.

The following discussion pertains to the reliability prediction of the Universe II integrated circuit and not the GE Fanuc SBC assembly. Please refer to the GE Fanuc specification sheet for product reliability information.

Physical Characteristics

- CMOS gate array
- 120,000 two-input nand gate equivalence
- 0.65 µm feature size
- 476 mils x 476 mils scribed die size

Thermal Characteristics

- Idle power consumption: 1.50 Watts
- Typical power consumption* (32-bit PCI): 2.00 Watts
- Maximum power consumption (32-bit PCI): 2.70 Watts
- Typical power consumption (64-bit PCI): 2.20 Watts
- Maximum power consumption* (64-bit PCI): 3.20 Watts

Note

Maximum power consumption is worst case consumption when the Universe II is performing DMA reads from the VME bus with alternating worst case data patterns ($FFFF_FFFF, $0000_0000 on consecutive cycles), and 100pF loading on the PCI bus.

In the majority of system applications, the Universe II will consume typical values or less. Typical power consumption numbers are based on the Universe II remaining idle 30%-50% of the time, which is significantly less than what is considered likely in most systems. For this reason, it is recommended that typical power consumption numbers be used for power estimation and ambient temperature calculations, as described below.

Reliability calculations of the Universe II design in Motorola's H4EPlus Gate Array family show that the Failure In Time (FIT) rate is 68 at a junction temperature of 125°C (maximum junction temperature). (Failure in time is the basic reliability rate expressed as failures per billion (1e-9) device hours. Mean Time Between Failures (MTBF) is the reciprocal of FIT. MTBF is the predicted number of device hours before a failure will occur.)

Universe II Ambient Operating Calculations

The maximum ambient temperature of the Universe II can be calculated as follows:

$T_a \leq T_j - \theta_{ja} * P$

Where,

T_a = Ambient temperature (°C)

T_j = Maximum Universe II Junction Temperature (°C)

θ_{ja} = Ambient to Junction Thermal Impedance (°C / Watt)

P = Universe II power consumption (Watts)

The ambient to junction thermal impedance (θ_{ja}) is dependent on the air flow in linear feet per minute over the Universe II. The values for θ_{ja} over different values of air flow are as follows:

Table E-1: Ambient to Junction Thermal Impedance

Air Flow (LFPM)	0	100	300
313 PBGA	20.10	17.0	15.1
324 CBGA	17.80	15.4	13.5

For example, the maximum ambient temperature of the 313 PBGA, 32-bit PCI environment with 100 LFPM blowing past the Universe II is:

$T_a \leq T_j - \theta_{ja} * P$

$T_a \leq 125 - 17.0 * 2.70$

$T_a \leq 79.1$ °C

Hence the maximum rated ambient temperature for the Universe II in this environment is 79.1°C. The thermal impedance can be improved by approximately 10% by adding thermal conductive tape to the top of the packages and through accounting for heat dissipation into the ground planes. This would improve the maximum ambient temperature to 87°C in the above example. Further improvements can be made by adding heat sinks to the PBGA package.

T_j values of Universe II are calculated as follows ($T_j = \theta_{ja} * P + T_a$)

Table E-2: Maximum Universe II Junction Temperature

Extended (125 C Ambient)	Industrial (85 C Ambient)	Commercial: (70 C Ambient)
T_j = 17.0 * 2.70 + 125 C = 170.9 C	T_j = 17.0 * 2.70 + 85 C = 130.9 C	T_j = 17.0 *2.70 + 70 C = 115.9 C

Thermal vias

The 313-pin plastic BGA package contains thermal vias which directly pipe heat from the die to the solder balls on the underside of the package. The solder balls use the capabilities of the power and ground planes of the printed circuit board to draw heat out of the package.

Appendix F Cycle Mapping

Introduction

The Universe II always performs Address Invariant translation between the PCI and VMEbus ports. Address Invariant mapping preserves the byte ordering of a data structure in a little-endian memory map and a big-endian memory map.

For more information on byte ordering and endian conversion, please refer to Chapter 4.

Little-endian Mode

Table F-1 below shows the byte lane swapping and address translation between a 32-bit little-endian PCI bus and the VMEbus for the address invariant translation scheme.

Table F-1: Mapping of 32-bit Little-Endian PCI Bus to 32-bit VMEbus

PCI Bus						Byte Lane Mapping	VMEbus			
Byte Enables				Address						
3	2	1	0	1	0		DS1	DS0	A1	LW
1	1	1	0	0	0	D0-D7 <-> D8-D15	0	1	0	1
1	1	0	1	0	1	D8-D15 <-> D0-D7	1	0	0	1
1	0	1	1	1	0	D16-D23 <-> D8-D15	0	1	1	1
0	1	1	1	1	1	D24-D31 <-> D0-D7	1	0	1	1
1	1	0	0	0	0	D0-D7 <-> D8-D15	0	0	0	1
						D8-D15 <-> D0-D7				
1	0	0	1	0	1	D8-D15 <-> D16-D23	0	0	1	0
						D16-D23<->D8-D15				
0	0	1	1	1	0	D16-D23 <-> D8-D15	0	0	1	1
						D24-D31 <-> D0-D7				
1	0	0	0	0	0	D0-D7 <-> D24-D31	1	0	0	0
						D8-D15 <-> D16-D23				
						D16-D23 <-> D8-D15				
0	0	0	1	0	1	D8-D15 <-> D16-D23	0	1	0	0
						D16-D23 <-> D8-D15				
						D24-D31 <-> D0-D7				
0	0	0	0	0	0	D0-D7 <-> D24-D31	0	0	0	0
						D8-D15 <-> D16-D23				
						D16-D23 <-> D8-D15				
						D24-D31 <-> D0-D7				

The unpacking of multiplexed 64-bit data from the VMEbus into two 32-bit quantities on a little-endian PCI bus is outlined in Table F-2 below.

Table F-2: Mapping of 32-bit Little-Endian PCI Bus to 64-bit VMEbus

Byte Enables				**Address**			**PCI to VME Byte Lane Mapping**
3	**2**	**1**	**0**	**2**	**1**	**0**	
First Transfer (D32-D63)							
0	0	0	0	0	0	0	D0-D7 <-> A24-A31 (D56-D63)
							D8-D15 <-> A16-A23 (D48-D55)
							D16-D23 <-> A8-A15 (D40-D47)
							D24-D31 <-> LWORD, A1-A7 (D32-D39)
Second Transfer (D0-D31)							
0	0	0	0	1	0	0	D0-D7 <-> D24-D31
							D8-D15 <-> D16-D23
							D16-D23 <-> D8-D15
							D24-D31 <-> D0-D7

Appendix G Operating and Storage Conditions

Introduction

The following discussion pertains to the operating and storage of the Universe II integrated circuit and not the GE Fanuc SBC assembly. Please refer to the specification data sheet for product operating and storage information.

Table G-1: Recommended Operating Conditions

DC Supply Voltage (V_{DD})	5 V
Ambient Operating Temperature (T_A Commercial)	0°C to +70°C
Ambient Operating Temperature (T_A Industrial)	-40°C to +85°C
Ambient Operating Temperature (T_A Extended)	-55°C to +125°C

Table G-2: Absolute Maximum Ratings

DC Supply Voltage (V_{SS} to V_{DD})	-0.5 to 6.0 V
Input Voltage (VIN)	-0.5 to VDD+0.5 V
DC Current Drain per Pin, Any Single Input or Output	±50 mA
DC Current Drain per Pin, Any Paralleled Outputs	±100 mA
DC Current Drain V_{DD} and V_{SS} Pins	±75 mA
Storage Temperature, (T_{STG})	0°C to 70°C

Warning

Stresses beyond those listed above may cause permanent damage to the devices. These are stress ratings only, and functional operation of the devices at these or any other conditions beyond those indicated in the operational sections of this document is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table G-3: Power Dissipation

IDLE	
Power Dissipation (32-bit PCI)	1.97W
Power Dissipation (64-bit PCI)	2.12W
Typical	
Power Dissipation (32-bit PCI)	2.65W
Power Dissipation (64-bit PCI)	3.15W