Universe IID/IIB™ User Manual

May 12, 2010

6024 Silver Creek Valley Road, San Jose, California 95138
Telephone: (800) 345-7015 • (408) 284-8200 • FAX: (408) 284-2775
Printed in U.S.A.

Contents

Figures

Tables

About this Document

This section discusses the following topics:

Scope

The *Universe IID/IIB User Manual* discusses the features, capabilities, and configuration requirements for the Universe II. It is intended for hardware and software engineers who are designing system interconnect applications with the device.

Document Conventions

This document uses the following conventions.

Signal Notation

Signals are either active high or active low. Active low signals are defined as true (asserted) when they are at a logic low. Similarly, active high signals are defined as true at a logic high. Signals are considered asserted when active and negated when inactive, irrespective of voltage levels. For voltage levels, the use of 0 indicates a low voltage while a 1 indicates a high voltage.

For voltage levels, the use of 0 indicates a low voltage while a 1 indicates a high voltage. For voltage levels, the use of 0 indicates a low voltage while a 1 indicates a high voltage.

Each signal that assumes a logic low state when asserted is followed by an underscore sign, "_". For example, SIGNAL_ is asserted low to indicate an active low signal. Signals that are not followed by an underscore are asserted when they assume the logic high state. For example, SIGNAL is asserted high to indicate an active high signal.

The asterisk sign "*" is used in this manual to show that a signal is asserted low and that is used on the on the VMEbus backplane. For example, SIGNAL* is asserted to low to indicate an active low signal on the VMEbus backplane.

Object Size Notation

- A *byte* is an 8-bit object.
- A *word* is a 16-bit object.
- A *doubleword* (Dword) is a 32-bit object.
- A quadword is a 64-bit (8 byte) object.
- A Kword is 1024 16-bit words.

Numeric Notation

- Hexadecimal numbers are denoted by the prefix *0x* (for example, 0x04).
- Binary numbers are denoted by the prefix *0b* (for example, 0b010).
- Registers that have multiple iterations are denoted by {x..y} in their names; where *x* is first register and address, and *y* is the last register and address. For example, REG{0..1} indicates there are two versions of the register at different addresses: REG0 and REG1.

Symbols

This symbol indicates a basic design concept or information considered helpful.

This symbol indicates important configuration information or suggestions.

This symbol indicates procedures or operating levels that may result in misuse or damage to the device.

Document Status Information

- Advance – Contains information that is subject to change, and is available once prototypes are released to customers.
- Preliminary – Contains information about a product that is near production-ready, and is revised as required.
- Formal – Contains information about a final, customer-ready product, and is available once the product is released to production.

Revision History

May 12, 2010, Formal

This document fixed a number of minor typographical errors. No technical changes were made.

October 2009, Formal

This document was rebranded as IDT. No technical changes were made.

June 2009, Formal

There have been changes throughout the manual.

August 2007, Formal

There have been numerous edits throughout the manual. The formatting of the document has also been updated.

November 2002, Formal

This document information applies to both the Universe IIB and the Universe IID devices. The Universe IID is recommended for all new designs. For more information about the two devices, see the *Universe IID/IIB Differences Summary.*

The following chapter was updated for the release of this manual:

- “Reliability Prediction” on page 355

October 2002, Formal

This document information applies to both the Universe IIB and the Universe IID devices. The Universe IID is recommended for all new designs. For more information about the two devices, see the *Universe IID/IIB Differences Summary.*

There was an erratum found in the 361 DBGA package drawing.

1. Functional Overview

This chapter outlines the functionality of the Universe II. This chapter discusses the following topics:

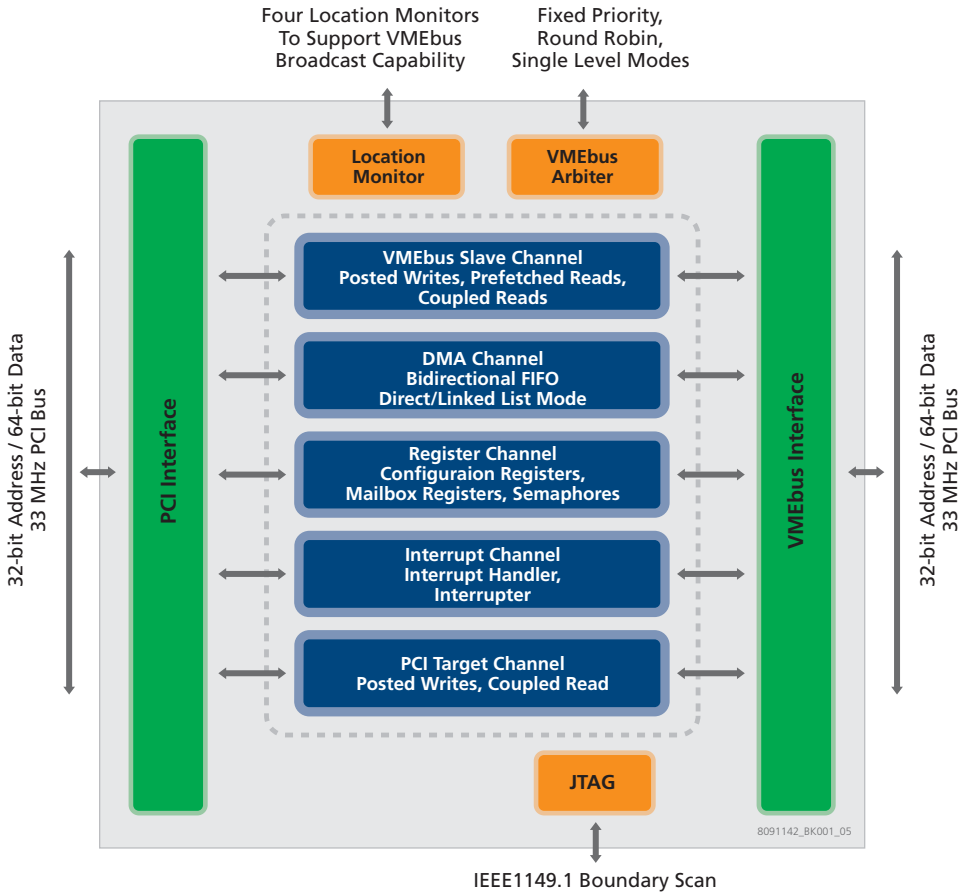
1.1 Overview

The IDT Universe II is the industry's leading high performance PCI-to-VMEbus interconnect. Universe II is fully compliant with the VME64 bus standard, and tailored for the next-generation of advanced PCI processors and peripherals. With a zero-wait state implementation, multi-beat transactions, and support for bus-parking, Universe II provides high performance on the PCI bus.

The Universe II eases development of multi-master, multi-processor architectures on VMEbus and PCI bus systems. The device is ideally suited for CPU boards functioning as both master and slave in the VMEbus system, and that require access to PCI systems. Bridging is accomplished through a decoupled architecture with independent FIFOs for inbound, outbound, and DMA traffic. With this architecture, throughput is maximized without sacrificing bandwidth on either bus.

With the Universe II, you know that as your system becomes more complex, you have proven silicon that continues to provide everything you need in a PCI-to-VME bridge.

Figure 1: Universe II Block Diagram



1.1.1 Universe II Features

The Universe II has the following features:

- Industry-proven, high performance 64-bit VMEbus interconnect
- Fully compliant, 32-bit or 64-bit, 33 MHz PCI bus interconnect
- Integral FIFOs for write posting to maximize bandwidth utilization
- Programmable DMA controller with Linked-List mode (Scatter/Gather) support
- Flexible interrupt logic
- Sustained transfer rates up to 60-70 Mbytes/s
- Extensive suite of VMEbus address and data transfer modes
- Automatic initialization for slave-only applications
- Flexible register set, programmable from both the PCI bus and VMEbus ports
- Full VMEbus system controller
- Support for RMWs, ADOH, PCI LOCK_ cycles, and semaphores

- Commercial, industrial, and extended temperature variants
- IEEE 1149.1 JTAG
- Available packaging:
  - 35mm x 35mm, 313-contact plastic BGA (PBGA) package

1.1.2 Universe II Benefits

The Universe II offers the following benefits to designers:

- Industry proven device
- Reliable customer support with experience in hundreds of customer designs

1.1.3 Universe II Typical Applications

The Universe II is targeted at today’s technology demands, such as the following:

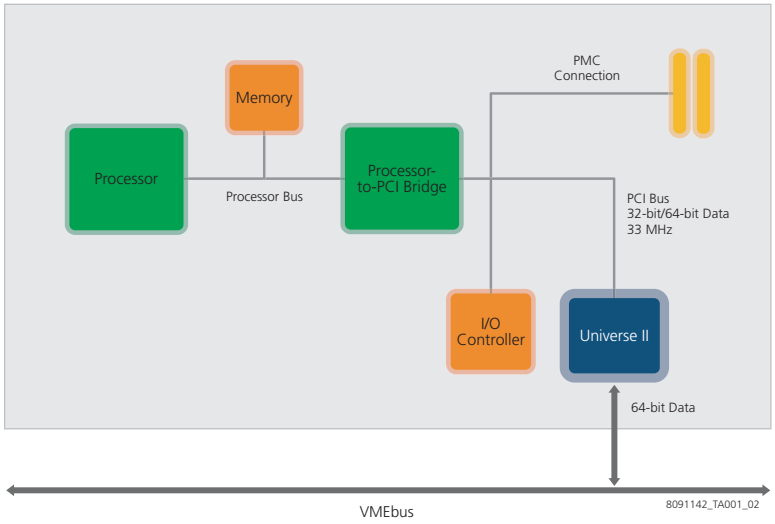
- Single-board computers
- Telecommunications equipment
- Test equipment
- Command and control systems
- Factory automation equipment
- Medical equipment
- Military
- Aerospace

1.1.3.1 Typical Application Example: Single Board Computers

The Universe II is widely used on VME-based Single Board Computers (SBC) that employ PCI as their local bus and VME as the backplane bus, as shown in the accompanying diagram. These SBC cards support a variety of applications including telecom, datacom, medical, industrial, and military equipment.

The Universe II high performance architecture seamlessly bridges the PCI and VME busses, and is the VME industry's standard for single board computer interconnect device.

Figure 2: Universe II In Single Board Computer Application



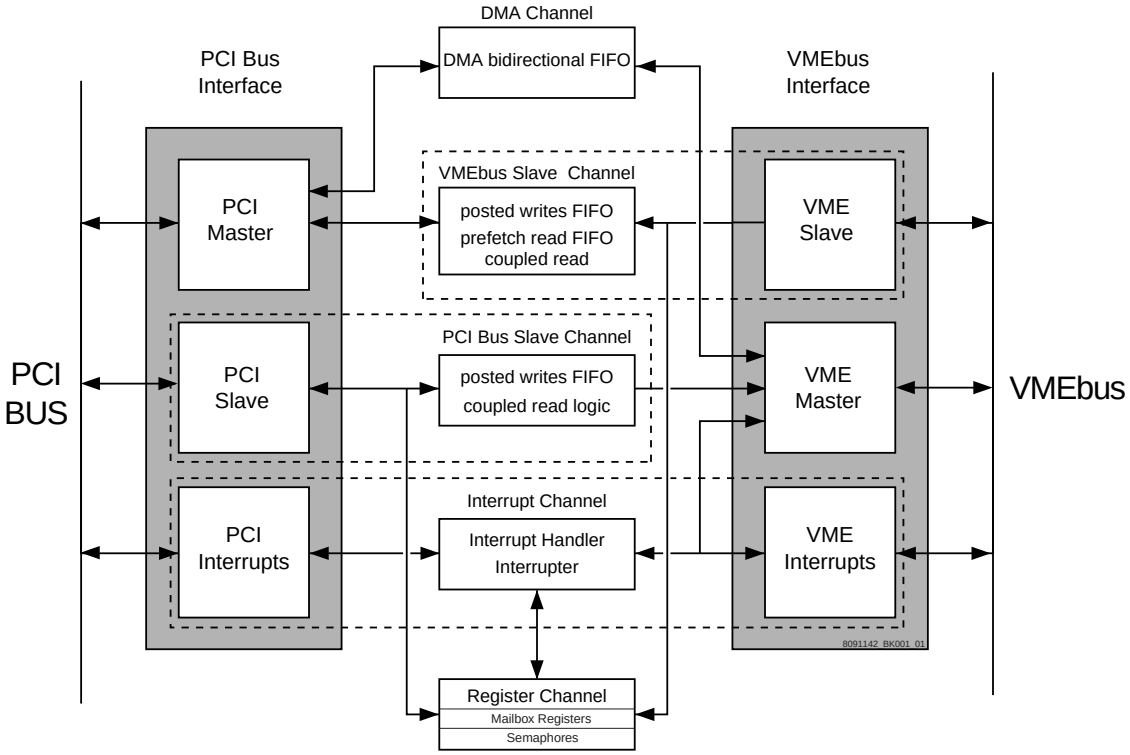
1.2 Main Interfaces

The Universe II has two main interfaces: the PCI Bus Interface and the VMEbus Interface. 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. The device had the following channels:

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

Figure 3 shows the Universe II in terms of the different modules and channels.

Figure 3: Universe II Data Flow Diagram



1.2.1 VMEbus Interface

The VME Interface is a *VME64 Specification* compliant interface.

1.2.1.1 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. Incoming write transactions from the VMEbus can be treated as either coupled or posted, depending upon the programming of the VMEbus slave image (see “VME Slave Image Programming” on page 67). 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 33 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. 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 instead of directly from the PCI resources.

As VMEbus slave, the Universe II does not assert RETRY* as a termination of the transaction.

1.2.1.2 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. 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.

1.2.2 PCI Bus Interface

The PCI Interface is a *PCI 2.1 Specification* compliant interface

1.2.2.1 Universe II as PCI Target

Read transactions from the PCI bus are always processed as coupled transactions. Write transactions can be either coupled or posted, depending upon the setting of the PCI bus target image (see “PCI Bus Target Images” on page 70). 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 60 for a full description of this operation).

The Universe II has a Special Cycle Generator that enables PCI masters to perform RMW and ADOH cycles. The Special Cycle Generator must be used in combination with a VMEbus ownership function to guarantee PCI masters exclusive access to VMEbus resources over several VMEbus transactions (see “Special Cycle Generator” on page 61 and “Using the VOWN bit” on page 64 for a full description of this functionality).

1.2.2.2 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.

1.2.3 Interrupter and Interrupt Handler

The Universe II has both interrupt generation and interrupt handling capability.

1.2.3.1 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 111 and “Interrupt Handling” on page 116 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 VIRQ_ 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. The interrupter also generates an internal interrupt in this situation if the SW_IACK bit, in the PCI Interrupt Status (LINT_STAT) register, is set to 1 (see “VMEbus Interrupt Generation” on page 113).

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 111).

1.2.3.2 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.

1.2.4 DMA Controller

The Universe II has 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 85).

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 101), 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 linked-list mode. The DMA works 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.

2. VMEbus Interface

This chapter explains the operation of the VMEbus Interface.This chapter discusses the following topics:

2.1 Overview

The VMEbus Interface incorporates all operations associated with the VMEbus. This includes master and slave functions, VMEbus configuration and system controller functions.

2.2 VMEbus Requester

There are different channels in the Universe II which require the use of the VMEbus. They are referred to as VMEbus requesters and are described in the following sections.

2.2.1 Internal Arbitration for VMEbus Requests

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.

2.2.1.1 Interrupt Channel

The Interrupt Channel always has the highest priority for access to the VMEbus Master Interface (see Figure 3 on page 21). 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 has completed the transaction. The definition of a complete transaction for each channel is in “VMEbus Release” on page 27.

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

2.2.1.2 PCI Target Channel

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

- TXFIFO contains a complete transaction
- A coupled cycle request

2.2.1.3 DMA Channel

The DMA Channel requests the VMEbus Master Interface in the following instances:

- 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)
- The DMA block is complete (see “DMA Controller” on page 85).

In the case of the DMA Channel, the DMA Channel VMEbus-off-timer can be used 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 101).

The Universe II provides a software mechanism for VMEbus acquisition through the VMEbus ownership bit (VOWN in the “Master Control Register (MAST_CTL)” on page 271). 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) and optionally generates an interrupt to the PCI bus (see “VME Lock Cycles—Exclusive Access to VMEbus Resources” on page 63). 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.

2.2.2 Request Modes

The Universe II has configurable request modes of operation.

2.2.2.1 Request Levels

The Universe II is software configurable to request on any one of the four 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 VMEbus Release Mode (VRL) field in the “Master Control Register (MAST_CTL)” on page 271. 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.

2.2.2.2 Fair and Demand Modes

The Universe II requester can be programmed for either Fair or Demand mode. The request mode is a global programming option set through the VMEbus Request Mode (VRM) bits in the “Master Control Register (MAST_CTL)” on page 271.

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 the default setting of Demand mode, 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.

2.2.3 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 "Master Control Register (MAST_CTL)" on page 271. 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 can 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 (see below). Note that the MYBBSY status bit in the "Miscellaneous Status Register (MISC_STAT)" on page 275 is 0 when the Universe II asserts BBSY*.

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.

2.2.3.1 Transaction Complete

The interrupt is complete when a single interrupt acknowledge cycle is complete.

The PCI Target Channel is complete under the following conditions:

- when the TXFIFO is empty, the TXFE bit is clear (the TXFE bit is set by the Universe II in the MISC_STAT register,
- when the maximum number of bytes per PCI Target Channel tenure has been reached (as programmed with the PWON field in the "Master Control Register (MAST_CTL)" on page 271[1],
- after each posted write, if the PWON is equal to 0b1111, as programmed in the MAST_CTL register
- when the coupled cycle is complete and the Coupled Window Timer has expired,
- if the Coupled Request Timer (page 58) 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 63).

The DMA Channel is complete under the following conditions:

- DMAFIFO full during VMEbus to PCI bus transfers,
- DMAFIFO empty during PCI bus to VMEbus transfers,

1. This setting is overridden if the VOWN mechanism is used.

- if an error is encountered during the DMA operation,
- the DMA VMEbus Tenure Byte Counter has expired, or
- DMA block is complete.

Refer to “FIFO Operation and Bus Ownership” on page 101 and “DMA Error Handling” on page 105 for more information.

Universe II ownership of the VMEbus is not affected by the assertion of BCLR* because it does not monitor BCLR*.

2.3 Universe II as VMEbus Master

The Universe II becomes VMEbus master under the following circumstances:

1. 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
3. Universe II Master Interface requests and obtains ownership of the VMEbus

The Universe II also becomes VMEbus master if the VMEbus ownership bit is set (see “VME Lock Cycles—Exclusive Access to VMEbus Resources” on page 63) and in its role in VMEbus interrupt handling (see “VMEbus Interrupt Handling” on page 116).

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.

2.3.1 Addressing Capabilities

Depending upon the programming of the PCI target image (see “PCI Bus Target Images” on page 70), 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. The *VMEbus Specification* does not permit pipelining during MBLT cycles.

The address and Address Modifier (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 70) or through DMA programming. Table 1 shows the AM codes used for the VMEbus.

Table 1: VMEbus Address Modifier Codes

Address Modifier	Address Bits	Description
0x3F	24	A24 supervisory block transfer (BLT)
0x3E	24	A24 supervisory program access
0x3D	24	A24 supervisory data access
0x3C	24	A24 supervisory 64-bit block transfer (MBLT)

Table 1: VMEbus Address Modifier Codes

Address Modifier	Address Bits	Description
0x3B	24	A24 non-privileged block transfer (BLT)
0x3A	24	A24 non-privileged program access
0x39	24	A24 non-privileged data access
0x38	24	A24 non-privileged 64-bit block transfer (MBLT)
0x37	40	A40BLT [MD32]
0x35	40	A40 lock command (LCK)
0x34	40	A40 access
0x32	24	A24 lock command
0x2F	24	CR/CSR
0x2D	16	A16 supervisory access
0x2C	16	A16 lock command
0x29	16	A16 non-privileged access
0x21	32 or 40	2eVME for 3U bus modules (address size in XAM code)
0x20	32 or 40	2eVME for 6U bus modules (address size in XAM code)
0x10 - 0x1F	Undefined	User-defined
0xF	32	A32 supervisory block transfer (BLT)
0xE	32	A32 supervisory program access
0xD	32	A32 supervisory data access
0xC	32	A32 supervisory 64-bit block transfer (MBLT)
0xB	32	A32 non-privileged block transfer (BLT)
0xA	32	A32 non-privileged program access
0x9	32	A32 non-privileged data access
0x8	32	A32 non-privileged 64-bit block transfer (MBLT)
0x5	32	A32 lock command
0x4	64	A64 lock command
0x3	64	A64 block transfer (BLT)
0x1	64	A64 single transfer access
0x0	64	A64 64-bit block transfer (MBLT)

The Universe II generates Address-Only-with-Handshake (ADOH) cycles in support of lock commands for A16, A24, and A32 spaces. ADOH cycles can only be generated through the Special Cycle Generator (see “Special Cycle Generator” on page 61).

There are two User Defined AM codes that can be programmed through the “User AM Codes Register (USER_AM)” on page 276. The USER_AM register can only be used to generate and accept AM codes 0x10 through 0x1F. The default USER_AM code is 0x10. 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 code is used with the VMEbus Slave Interface, the cycles must use 32-bit addressing, and only single cycle accesses are used. BLTs and MBLTs with USER_AM codes will lead to unpredictable behavior.

2.3.2 Data Transfer Capabilities

PCI and VMEbus protocols have different data transfer capabilities. The maximum data width for a VMEbus data transfer is programmed with the VMEbus Maximum Datawidth (VDW) field in the “PCI Target Image 0 Control (LSI0_CTL)” on page 181. 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-, 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 is broken into three aligned VMEbus cycles: three single-byte cycle (the ordering of the 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-bit, then the three-byte PCI data beat is 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 70). The length of the BLT/MBLT transactions on the VMEbus is determined by the initiating PCI transaction. 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 is queued 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 64).

During DMA operations, the Universe II attempts block transfers to the maximum length permitted by the VMEbus specification (256 bytes for BLT, 2 Kbytes for MBLT) and is limited by the VON counter (see “DMA VMEbus Ownership” on page 91).

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 63).

2.3.3 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. The assertion of BERR* during an IACK also causes the error to be logged.

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 or during an IACK cycle, the Universe II uses the “VMEbus AM Code Error Log (V_AMERR)” on page 307 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 “VMEbus Address Error Log (VAERR)” on page 308. 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 116).

DTACK* signals the successful completion of the transaction.

2.4 Universe II as VMEbus Slave

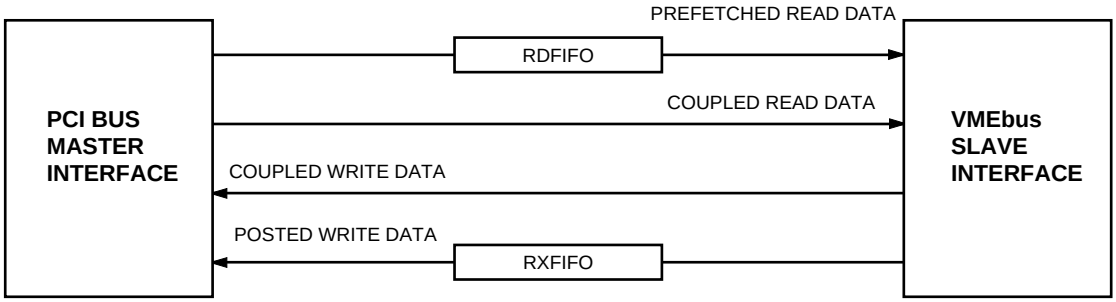
This section describes the VMEbus Slave Channel and other aspects of the Universe II as VMEbus slave.

The Universe II becomes VMEbus slave when one of its eight programmed slave images or register images are accessed by a VMEbus master. Depending upon the programming of the slave image, different possible transaction types can result (see “VME Slave Image Programming” on page 67).

The Universe II cannot reflect a cycle on the VMEbus and access itself.

For reads, the transaction can be coupled or prefetched. 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 4 and “VME Slave Image Programming” on page 67). To ensure sequential consistency, prefetched reads, coupled reads, and coupled write operations are only processed once all previously posted write operations have completed (the RXFIFO is empty).

Figure 4: 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.

In order for a VMEbus slave image to respond to an incoming cycle, the PCI Master Interface must be enabled by setting the bit BM in the “PCI Configuration Space Control and Status Register (PCI_CSR)” on page 172. If data is queued in the VMEbus Slave Channel FIFO and the BM bit is cleared, the FIFO empties but no additional transfers are received.

2.4.1 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” on page 33).

A coupled cycle with multiple data beats (such as 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 Image Programming" on page 67). 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*. The Universe II sets the R_TA or R_MA bits in the "PCI Configuration Space Control and Status Register (PCI_CSR)" on page 172 when it receives a Target-Abort or Master-Abort.

2.4.2 Posted Writes

A posted write involves the VMEbus master writing data into the Universe II's RXFIFO, instead of 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 Image Programming" on page 67). If the bit is cleared (default setting) the transaction bypasses the FIFO and is performed as a coupled transfer. Incoming posted writes from the VMEbus are queued in the 64-entry deep RXFIFO. Each entry in the RXFIFO can contain 32 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 Image Programming" on page 67). For this reason, any reprogramming of VMEbus slave image attributes are only reflected in RXFIFO entries queued after the reprogramming. Transactions queued before the re-programming are delivered to the PCI bus with the VMEbus slave image attributes that were in use before the reprogramming.

The RXFIFO is the same structure as the RDFIFO. The different names are used for the FIFO's two roles. In each FIFO, only one role, either the RXFIFO or the RDFIFO, can used at one time.

2.4.2.1 FIFO Entries

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 the address entry is added to the three data entries, this VMEbus block write has been stored in a total of five RXFIFO entries.

Unlike the PCI Target Channel (see “Universe II as PCI Target”), 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 available 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 “Master Control Register (MAST_CTL)” on page 271.

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). The data width for a VMEbus transaction to the PCI bus is programmed in the LD64EN bit in the “VMEbus Slave Image 1 Control (VSI1_CTL)” on page 285. The LD64EN bit enables 64-bit PCI bus transactions 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. VMEbus D16 write transactions are mapped to D32 write transactions on the PCI bus with all byte lanes enabled. (However, 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 packs 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.

2.4.2.2 Errors

If an error occurs during a posted write to the PCI bus, the Universe II uses the “PCI Command Error Log Register (L_CMDERR)” on page 210 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 of errors is not given. The error log is qualified with the L_STAT bit. The address of the errored transaction is latched in the “PCI Address Error Log (LAERR)” on page 211. An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “Error Handling” on page 125 and “Interrupt Generation and Handling” on page 109).

2.4.3 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 Image Programming” on page 67). 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.

The VMEbus lies idle during the arbitration time required for each PCI bus transaction, resulting in a performance degradation.

With prefetching enabled, the VMEbus Slave Channel uses a 64-entry deep RDFIFO to provide read data to the VMEbus with minimum latency. The RDFIFO is 64-bit, with additional bits for control information. If a VMEbus slave image is programmed for prefetching (see “VME Slave Image Programming” on page 67), 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. 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 decoupled (see “Prefetched Reads” on page 127 for the impact on bus error handling).

The RXFIFO is the same structure as the RDFIFO. The different names are used for the FIFO’s two roles. In each FIFO, only one role, either the RXFIFO or the RDFIFO, can used at one time.

2.4.3.1 FIFO Entries

When there is a transactions from the Universe PCI Slave to the Universe VME Master, 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 1 Control (VSI1_CTL)” on page 285 and the capabilities of the accessed PCI target.

Internally, the prefetched read data is packed to 64-bit, 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 (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 be unable to deliver data to the VMEbus master. For this reason, block read performance on the VMEbus is similar to that observed with block writes. However, if the RDFIFO be unable to deliver data to the VMEbus master (which can 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 bus, 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 “Master Control Register (MAST_CTL)” on page 271). 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 removed.

Reading on the PCI bus does not cross a 2048-byte boundary. The PCI Master Interface releases FRAME_ and the VMEbus Slave Channel relinquishes internal ownership of the PCI Master Interface when it reaches this boundary. The VMEbus Slave Channel re-requests internal ownership of the PCI Master Interface as soon as possible, in order to continue reading from the external PCI target.

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 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.

2.4.3.2 Errors

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

2.4.4 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 (see Table 1 on page 28). The purpose of the Lock command is to lock the resources on a card so a master on the card cannot modify the resource. 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 32-bit read transfer with all byte lanes enabled (no data). All subsequent slave VMEbus transactions are coupled while the Universe II owns PCI LOCK_. The Universe II holds the PCI bus lock until the VMEbus lock command is terminated (by negating BBSY*).

The VMEbus Slave Channel has dedicated access to the PCI Master Interface during the locked transaction.

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.

2.4.4.1 Errors

If an error occurs on the PCI bus, a bus error will occur on the VMEbus because they are coupled. In the event a bus error occurs on the VMEbus once a LOCK_ has been established, the VMEbus master which locked the VMEbus must terminate the LOCK_ by negating BBSY*.

2.4.4.2 DMA Access

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

LOCK_ is negated on the PCI bus when AS* is negated on the VMEbus. LOCK_ is not negated when AS* is negated if LOCK_ was asserted by an ADOH/lock command

2.4.5 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 1 Control (VSI1_CTL)” on page 285), then every non-block slave read is mapped to a coupled PCI locked read. LOCK_ is 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.

RMW cycles are not supported with unaligned or D24 cycles.

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 are treated as independent transactions on the PCI bus (a read followed by a write). The write can be coupled or decoupled depending on the state of the PWEN bit in the accessed slave image.

There can be an adverse performance impact for reads that are processed through a RMW-capable slave image This can be increased if LOCK_ is currently owned by another PCI master.

When an external VMEbus Master begins a RMW cycle, at some point a read cycle appears 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 occurs on the PCI bus.

2.4.6 Register Accesses

See “Registers” on page 163 for a full description of register mapping and register access.

2.4.7 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)" on page 300 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)" on page 302 specifies the lowest address in the 4 Kbyte range that is decoded as a location monitor access. While the Universe II is has 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 is used, and hence which of four PCI interrupts to generate (see "Location Monitors" on page 122).

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 must be programmed to monitor the same 4 Kbytes of addresses on the VMEbus. If the Universe II accesses its own (enabled) location monitor, the same Universe II generates DTACK* on the VMEbus and 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*.

2.4.8 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.

ADOH, BLT and MBLT cycles must not be attempted when the LAS field of an image is programmed for PCI Configuration Space.

PCI Configuration cycles can only be generated when the VAS field in the appropriate VSIx_CTRL register is programmed for either A32, USER1, or USER2.

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 Bus Number field (BUS_NO[7:0]) in the "Master Control Register (MAST_CTL)" on page 271. 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.

2.4.8.1 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, the following steps must be completed:

1. Program the LAS field of VSIx_CTL for Configuration Space
2. Program the VSIx_BS, VSIx_BD registers to some suitable value
3. Program the VSIx_TO register to 0
4. 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
- VA[23:16] matches the BUS_NO in MAST_CTL register
- Other address bits are not important—they are not mapped to the PCI bus

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

VA[15:11][a]	PCI Address Line Asserted[b]
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

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

VA[15:11][a]	PCI Address Line Asserted[b]
01111	26
10000	27
10001	28
10010	29
10011	30
10100	31

a. The other values of VA[15:11] are not defined and must not be used.

b. Only one of AD[31:11] is asserted; the other address lines in AD[31:11] are negated.

ADOH, BLT and MBLT cycles must not be attempted when the LAS of an image is programmed to PCI Configuration space.

2.4.8.2 Generating Configuration Type 1 Cycles

The following steps are used to generate a configuration Type 1 cycle on the VMEbus:

1. Program LAS field of VSIx_CTL to Configuration Space
2. Program the VSIx_BS, VSIx_BD registers to some suitable value
3. Program the VSIx_TO register to 0
4. 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.

2.5 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
- Auto Slot ID (two methods)

2.5.1 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 *VME64 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 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 "Miscellaneous Control Register (MISC_CTL)" on page 273.

The Universe II monitors IACK*, instead of 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.

2.5.2 VMEbus Register Access at Power-up

The Universe II provides a VMEbus slave image that allows access to all Universe II registers. The base address for the slave image is programmed through the "VMEbus Register Access Image Base Address Register (VRAI_BS)" on page 304. At power-up, the Universe II can program the VRAI_BS and "VMEbus Register Access Image Control Register (VRAI_CTL)" on page 303 registers with information specifying the Universe II Control/Status (UCSR) register slave image (see "Power-Up Options" on page 135).

Register access at power-up is 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.

2.6 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 IDT SCV64 and the Universe II. Neither system identifies geographical addressing, only the relative position amongst the boards present in the system, for example, fourth board versus fourth slot.

Both the VME64 Auto Slot ID and the DY4 method of automatic slot identification are activated through a power-up option. Refer "Power-Up Options" on page 135 for more information.

Auto ID prevents the need for jumpers to uniquely identify cards in a system. The benefits of this feature are:

- Increases the speed of system level repairs in the field
- Reduces the possibility of incorrect configurations
- Reduces the number of unique spare cards that must be stocked

2.6.1 Auto Slot ID: VME64 Specified

The VME64 Auto Slot ID cycle, as described in the *VME64 Specification*, requires at power-up that the Auto ID slave takes the following actions:

- Generate IRQ2*
- Negate SYSFAIL*

When the Auto Slot ID slave responds to the Monarch's IACK cycle, the following actions are taken:

1. Enable accesses to its CR/CSR space
2. Provide a Status/ID to the Monarch indicating the interrupt is an Auto-ID request
3. Assert DTACK*
4. Release IRQ2*

The Universe II participates in the VME64 Auto Slot 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 "Miscellaneous Control Register (MISC_CTL)" on page 273. 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 STATID register. The default value is 0xFE.

The Universe II can be programmed so that it does not release SYSFAIL* until the SYSFAIL bit in the "VMEbus CSR Bit Clear Register (VCSR_CLR)" on page 329 is cleared by local logic.

SYSFAIL* is asserted if the SYSFAIL bit in the "VMEbus CSR Bit Set Register (VCSR_SET)" on page 330 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 Slot ID cycle proceeds.

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

2.6.1.1 Universe II and the Auto Slot ID Monarch

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

1. Masks IRQ2*
   - It will not service other interrupters at that interrupt level until current Auto Slot ID cycle is completed
2. Performs an access at 0x00 in CR/CSR space to get information about Auto Slot ID slave
3. Moves the CR/CSR base address to a new location
4. 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.

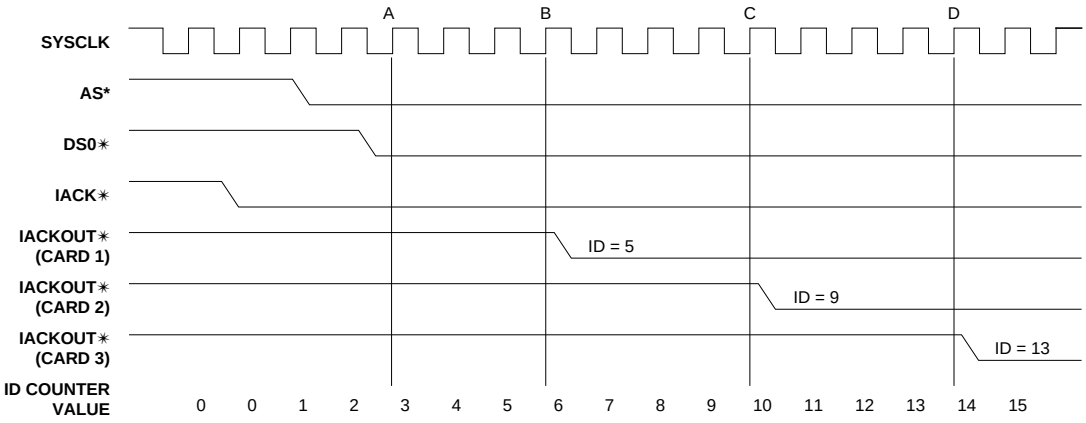
2.6.2 Auto-ID: A Proprietary IDT Method

The Universe II uses a proprietary Auto-ID scheme when enabled through a power-up option (see "Auto-ID" on page 137). The IDT 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 5). 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 5: 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 located in the “Miscellaneous Status Register (MISC_STAT)” on page 275.

2.6.3 System Controller Functions

When located in Slot 1 of the VMEbus system (see “First Slot Detector” on page 41), the Universe II assumes the role of SYSCON and sets the SYSCON status bit in the MISC_CTL register. In accordance with the *VME64 Specification*, as SYSCON the Universe II has the following functions:

- A system clock driver
- An arbitration module
- An IACK Daisy Chain Driver (DCD)
- A bus timer

2.6.3.1 System Clock Driver

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

2.6.3.2 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)
- Round Robin Arbitration Mode (RRS) (default setting)

These modes are selected with the VARB bit in the “Miscellaneous Status Register (MISC_STAT)” on page 275.

2.6.3.3 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 is asserted, the Arbitration Module asserts VBCLR_ until the owner releases the bus (VRBBSY_ is negated).

2.6.3.4 Round Robin Arbitration Mode (RRS)

This mode arbitrates all levels in a round robin mode, by 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 are required to service a queue of requests at one level.

2.6.3.5 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 can continue with other requesters. The timer is programmed using the VARBTO field in the MISC_CTL register, 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 can timeout in as little as 8 µs.

2.6.4 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* stays high for at least 30 ns as specified in rule 40 of the VME64 specification.

2.6.5 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 “Miscellaneous Control Register (MISC_CTL)” on page 273 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 VME Bus Time-out value has a 4us resolution. Therefore, the time-out may occur up to 4us earlier than the selected VME Bus Time-out period.

2.6.6 Bus Isolation Mode (BI-Mode)

BI-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
- Fault isolation in the event of a card failure
  - 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 are generated or responded to. No DMA activity occurs. Any access to other PCI images result in a Target-Retry. Access to other VMEbus images are ignored.

Entering BI-Mode has the following effects:

- The VMEbus Master Interface becomes inactive
  - PCI Target Channel coupled accesses are retried. The PCI Target Channel Posted Writes FIFO continues to accept transactions but eventually fills and no further posted writes are accepted. The DMA FIFO eventually empties or fills and no further DMA activity takes place on the PCI bus. The Universe II VMEbus Master does not service interrupts while in BI-Mode.
- The Universe II does not respond as a VMEbus slave
  - Except for accesses to the register image and CR/CSR image.
- The Universe II does not respond to any interrupt it had outstanding.
  - All VMEbus outputs from the Universe II are tri-stated, so that the Universe II are not 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 for the following reasons:

- If the BI-Mode power-up option is selected (See “Power-Up Options” on page 135 and Table 22 on page 135)
- When SYSRST* or RST_ is asserted any time after the Universe II has been powered-up in BI-Mode
- When VRIRQ_ [1] is asserted, provided that the ENGBI bit in the“Miscellaneous Control Register (MISC_CTL)” on page 273 has been set
- When the BI bit in the MISC_CTL register is set

Either of the following actions remove the Universe II from BI-Mode:

- Power-up the Universe II with the BI-Mode option off (see “BI-Mode” on page 137), or

- Clear the BI bit in the MISC_CTL register.
  - This is effective only if the source of the BI-Mode is no longer active. 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 does not work.

3. PCI Interface

Peripheral Component Interconnect (PCI) is a bus protocol that defines how devices communicate on a peripheral bus and with a host processor. If a device is referred to as PCI compliant it must be compliant with the *PCI Local Bus Specification (Revision 2.1)*. The Universe II PCI bus supports frequencies up to 33 MHz, and 32-bit or 64-bit transfers.

This chapter describes the Universe II's PCI Interface. This chapter discusses the following topics:

3.1 Overview

The Universe II PCI Bus Interface is a directly connected to the PCI bus. For information concerning the different types of PCI accesses available, see "PCI Bus Target Images" on page 70.

3.2 PCI Cycles

The PCI Bus Interface 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 "Endian Mapping" on page 359 and the *PCI 2.1 Specification*).

The Universe II has all the PCI signals described in the *PCI 2.1 Specification*) with the exception of SBO_ and SDONE. These pins are exception because 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 the following phases:

1. Request
2. Address
3. Data transfer
4. Cycle termination

3.2.1 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 138).

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 through the LD64EN bit, in the "DMA Transfer Control Register (DCTL)" on page 232. If the VMEbus slave image is programmed with a 64-bit PCI bus data width and Universe II is powered-up in a 64-bit PCI environment, the Universe II asserts REQ64_ during the address phase of the PCI transaction.

REQ64_ is asserted if LD64EN is set in a 64-bit PCI system independent of whether the Universe II has a full 64-bit transfer. This can 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 the LD64EN 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 never attempts a 64-bit cycle on the PCI bus if it is powered-up as a 32-bit device.

3.2.2 PCI Bus Request and Parking

The Universe II supports bus parking. If the Universe II requires the PCI bus it asserts 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.

Refer to the *PCI 2.1 Specification* for more information

3.2.3 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 II as VMEbus Slave" on page 32).

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

Table 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
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 “Memory Read Types” on page 51
1101	Dual Address Cycle	N/A
1110	Memory Read Line	See “Memory Read Types” on page 51
1111	Memory Write and Invalidate	See “Memory Read Types” on page 51

3.2.3.1 Memory Read Types

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 master, the Universe II can generate Memory Read Multiple but not Memory Read Line.

PCI targets must assert DEVSEL_ if they have decoded the access. During a Configuration cycle, the target is selected by its particular ID Select (IDSEL). If a target does not respond with DEVSEL_ within six clocks, a Master-abort is generated. The role of configuration cycles is described in the *PCI 2.1 Specification*.

3.2.4 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_ can 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 de-asserting 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 de-asserted (master-initiated termination) or if STOP_ is asserted (target-initiated). When both FRAME_ and IRDY_ are de-asserted (final data phase is complete), the bus is defined as idle.

3.2.5 Termination Phase

The PCI Bus Interface permits the following types of PCI terminations:

1. Master-abort: The PCI bus master negates FRAME_ when no target responds (DEVSEL_ not asserted) after six clock cycles.
2. 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 with data: The transaction is terminated after data is transferred. The Universe II de-asserts REQ_ for at least two clock cycles if it receives STOP_ from the PCI target.
    - Target-disconnect without data: The transaction is terminated before data is transferred. The Universe II de-asserts REQ_ for at least two clock cycles if it receives STOP_ from the PCI target.
3. 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.
4. 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.

3.2.6 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_[3:0] and AD[31:0]. 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.

When the Universe II is powered-up in a 64-bit PCI environment, it uses PAR64 in the same way as PAR, except for AD[63:32] and C/BE[7:4].

Universe II reports parity errors during all transactions with the PERR_ signal. 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.

3.3 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 following conditions are met:

- 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)
- There is a coupled cycle request

The DMA Channel makes an internal request for the PCI Master Interface when the following conditions are met:

- DMAFIFO has room for 128 bytes to be read from PCI
- DMAFIFO has queued 128 bytes to be written to PCI
- 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 the transaction is complete and the channels no longer need the PCI Master Interface. The VMEbus Slave Channel no longer needs the PCI Master Interface under the following conditions:

- An entire transaction (no greater in length than the programmed aligned burst size) is emptied from the RXFIFO
- The coupled cycle is complete

The DMA Channel no longer needs the PCI Master Interface when the following conditions are met:

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

Access from the VMEbus can be either coupled or decoupled. For a full description of the operation of these data paths, see “Universe II as VMEbus Slave” on page 32.

3.3.1 Command Types

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)
- 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 Image Programming” on page 67). When generating a memory transaction, the addressing is either 32-bit or 64-bit aligned, depending upon the PCI target. When generating an I/O transaction, the addressing is 32-bit aligned and all incoming transactions are coupled.

3.3.2 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 “Master Control Register (MAST_CTL)” on page 271). The PCI aligned burst size can be programmed at 32, 64 or 128 bytes. Burst transfers do not cross the programmed boundaries. For example, when programmed for 32-byte boundaries, a new burst begins at XXXX_XX20, XXXX_XX40, etc. If necessary, a new burst begins 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 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 master. 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 performing the following tasks:

- Emptying the RXFIFO
  - The TXFE bit in the “Miscellaneous Status Register (MISC_STAT)” on page 275 is clear
- Filling the RDFIFO receives a block read request from a VMEbus master to an appropriately programmed VMEbus slave image
- 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.

3.3.3 Termination

The Universe II performs a Master-Abort if the target does not respond within six 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 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 three 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 “Master Control Register (MAST_CTL)” on page 271. 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. The Universe II can issue a BERR* signal on the VMEbus. All VMEbus slave coupled transactions and decoupled transactions 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 “Error Handling” on page 125), the Universe II uses the “PCI Command Error Log Register (L_CMDERR)” on page 210 to log the command information for the transaction (CMDERR [3:0]) and the address of the errored transaction is latched in the “PCI Address Error Log (LAERR)” on page 211. 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 is 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 Generation and Handling” on page 109).

If an error occurs on the PCI bus, the Universe II does not translate the error condition into a BERR* on the VMEbus; the Universe II does not directly map the error. By taking no action, the Universe II forces the external VMEbus error timer to expire.

3.3.4 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].

When the Universe II is powered-up in a 64-bit PCI environment, it uses PAR64 in the same way as PAR, except for AD[63:32] and C/BE[7:4].

The PERESP bit in the “PCI Configuration Space Control and Status Register (PCI_CSR)” on page 172 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 (i.e. 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 monitors SERR_ and take appropriate action.

3.4 Universe II as PCI Target

The Universe II becomes PCI bus target when one of its nine programmed PCI target images, or one of its registers, is accessed by a PCI bus master. The
Universe II cannot access its own images or registers and master the PCI bus. Refer to “Registers” on page 163 for more information on register accesses.

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

3.4.1 Command Types

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.

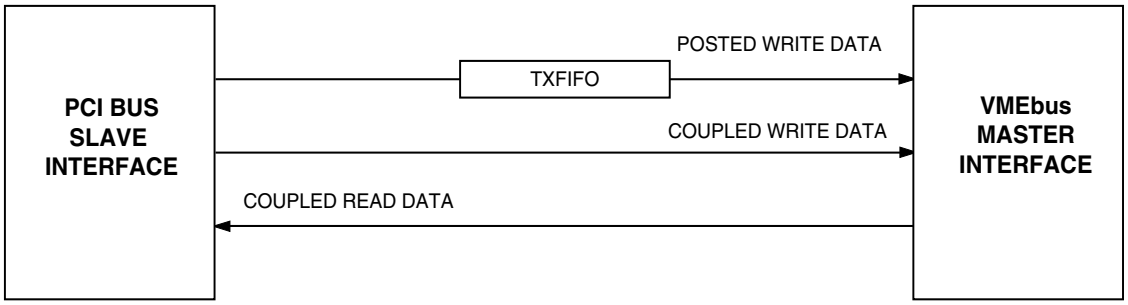
Address parity errors are reported if both PERESP and SERR_EN are set in the “PCI Configuration Space Control and Status Register (PCI_CSR)” on page 172. Address parity errors are reported by the Universe II by asserting the SERR_ signal and setting the S_SERR (Signalled SERR_) bit in the PCI_CS register. 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 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.

3.4.2 Data Transfer

Read transactions are always coupled, as opposed to VMEbus slave reads which can be pre-fetched (see “Universe II as VMEbus Slave” on page 32). Write transactions can be coupled or posted (see Figure 6 and “PCI Bus Target Images” on page 70). To ensure sequential consistency, coupled operations (reads or writes) are only processed once all previously posted write operations have completed (the TXFIFO is empty).

Figure 6: PCI Bus Target Channel Dataflow



The PCI bus and the VMEbus can have different data width capabilities. The maximum VMEbus data width is programmed into the PCI target image through the VDW bit in the “PCI Target Image 0 Control (LSI0_CTL)” on page 181. 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.

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-bit. The three-byte PCI data beat will be broken into three aligned VMEbus cycles: three single-byte cycles. If in the above example the PCI target image has a VDW set to 8-bit, then the three-byte PCI data beat will be broken into three single-byte VMEbus cycles.

3.4.3 Coupled Transfers

The PCI Target Channel supports *coupled* transfers. 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 where the termination of the cycle on the VMEbus is relayed directly to the PCI master in the normal manner (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 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, and the Universe II currently owns the VMEbus, the PCI Target Channel moves directly to the Coupled Data-Transfer Phase

3.4.3.1 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[1] 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 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.

For more information refer to “Data Transfer” on page 57.

3.4.3.2 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 moves directly to the Coupled Data-Transfer 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 “PCI Miscellaneous Register (LMISC)” on page 206. If this field is programmed to 0000, the Universe II does 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 releases 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.

1. PCI latency requirements (as described in revision 2.1 of the *PCI Specification*) require that only 16 clock cycles can elapse between the first and second data beat of a transaction. Since the Universe II cannot guarantee that data acknowledgment will be received from the VMEbus in time to meet these PCI latency requirements, the Universe II performs a target-disconnect after the first data beat of every coupled write transaction.

3.4.4 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 70) to 1 and setting the LAS bit to 0. Write transactions are relayed from the PCI bus to the VMEbus through a 64-entry deep TXFIFO. The TXFIFO allows each entry to contain 32 address bits (with extra bits provided for command information), or to a full 64-bit width. 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 (see “PCI Bus Target Images” on page 70). For this reason, any reprogramming of PCI bus target image attributes will only be reflected in TXFIFO entries queued after the reprogramming. Transactions queued before the re-programming are delivered to the VMEbus with the PCI bus target image attributes that were in use before the reprogramming.

Care must be taken before reprogramming target images. To ensure the FIFO is empty use one of the following options:

- Perform a coupled read. The coupled read does not complete until all posted-write data has been queued
- Read the MISC_STAT register until the TXFE bit has a value of 0

3.4.4.1 FIFO Entries

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

For 32-bit PCI transfers in the Universe II, the TXFIFO accepts a single burst of one address phase and 59 data phases when it is empty. For 64-bit PCI, the TXFIFO accepts 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 issues a disconnect if the address crosses 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 27). 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 70). For example, if the VMEbus data width is programmed to 16-bit, 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 by another master, and the FIFO does not empty, coupled transactions requested by the first PCI master in the PCI Target Channel does not proceed and are continually retried. This presents a potential starvation scenario.

This functionality is intended to support earlier versions of PCI-to-PCI bridges.

3.4.5 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 “Special Cycle PCI Bus Address Register (SCYC_ADDR)” on page 202, 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 (bits [31:2]) is used to 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 uses 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 7.

Figure 7: Register Fields for the Special Cycle Generator

Field	Register Bits	Description
32-bit address	ADDR in "Special Cycle PCI Bus Address Register (SCYC_ADDR)" on page 202	Specifies PCI bus target image address
PCI Address Space	LAS in "Special Cycle Control Register (SCYC_CTL)" on page 201	Specifies whether the address specified in the ADDR field lies in PCI memory or I/O space
Special cycle	SCYC[1:0] in "Special Cycle Control Register (SCYC_CTL)" on page 201	Disabled, RMW or ADOH
32-bit enable	EN [31:0] in "Special Cycle Swap/Compare Enable Register (SCYC_EN)" on page 203	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 "Special Cycle Compare Data Register (SCYC_CMP)" on page 204	Data which is compared to the VMEbus read data during a RMW cycle
32-bit swap	SWP [31:0] in "Special Cycle Swap Data Register (SCYC_SWP)" on page 205	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.

3.4.5.1 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) results 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 7). 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 equal 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.

RMW Constraints

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 "Special Cycle Control Register (SCYC_CTL)" on page 201) 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 only generates a RMW if it is accessed with an 8-bit, aligned 16-bit, or aligned 32-bit read cycle.
2. The Special Cycle Generator only generates a RMW if the size of the request is less than or equal to the programmed VMEbus Maximum Data width.
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 RMW criteria, the Universe II generates a Target-Abort. The
Universe II must be correctly programmed and accessed with correct byte-lane information.

3.4.5.2 VME Lock Cycles—Exclusive Access to VMEbus Resources

The VME Lock cycle is used, in combination with the VOWN bit in the "Master Control Register (MAST_CTL)" on page 271, 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. 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 8 on page 70).

However, after the VME Lock cycle is complete, there is no guarantee that the Universe II remains VMEbus master unless it has set the VOWN bit. If the Universe II loses VMEbus ownership, the VMEbus resouce does not remain locked.

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

1. 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
2. program the VOWN bit in the MAST_CTL register to a value of 1 (see "Using the VOWN bit" on page 64)

3. wait until the VOWN_ACK bit in the MAST_CTL register is a value of 1
4. generate an ADOH cycle with the Special Cycle Generator
5. perform transactions to be locked on the VMEbus
6. release the VMEbus by programming the VOWN bit in the MAST_CTL register to a value of 0
7. 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:

- All byte lane information is ignored for VME Lock cycles
- The Universe II generates a VME Lock cycle on the VMEbus if the PCI Target Image, which includes the special cycle, has posted writes disabled
- The Universe II Special Cycle Generator does not generate VME Lock cycles if the address space is not one of A16, A24 or A32 — it produces regular cycles instead

3.4.6 Using the VOWN bit

The Universe II provides a VMEbus ownership bit (VOWN bit in the “Master Control Register (MAST_CTL)” on page 271) 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 in the “PCI Interrupt Enable Register (LINT_EN)” on page 242. While the VMEbus is held using the VOWN bit, the Universe II sets the VOWN_ACK bit in the MAST_CTL register. The act of changing the VOWN_ACK bit from 0 to 1 generates and interrupt. The VMEbus Master Interface maintains bus tenure while the ownership bit is set, and only releases the VMEbus when the ownership bit is cleared.

3.4.6.1 Reasons for Using the VOWN Bit

If the VMEbus Master Interface is programmed for RWD, through the VREL bit in “Master Control Register (MAST_CTL)” on page 271, it may release the VMEbus when the PCI Target Channel has completed a transaction. If exclusive access to the VMEbus resource is required for multiple transactions, then the VMEbus ownership bit holds the bus until the exclusive access is no longer required.

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

3.4.7 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 a coupled cycle with FRAME_ asserted

   A Target-disconnect with data only occurs if FRAME_ is asserted.

   - After the first data phase of a PCI Memory command (with FRAME_ asserted) if AD[1:0] is not equal to 00 (refer to Revision 2.1 of the *PCI Specification*)
2. Target-Retry
   - 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 (for 64-bit PCI)
   - 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
   - Register Channel is locked by the VME Slave Channel if a register access (including a RMW access) is in progress or the registers have been locked by an ADOH access. If the registers are locked by the VME Slave Channel, register accesses by external PCI masters are retried
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 Configuration Space Control and Status Register (PCI_CSR)” on page 172

Whether to terminate a transaction or for retry purposes, the Universe II keeps STOP_ asserted until FRAME_ is de-asserted, independent of the logic levels of IRDY_ and TRDY_. If STOP_ is asserted while TRDY_ is de-asserted, it means that the Universe II does not transfer any more data to the master.

3.4.7.1 Error During Posted Write

If an error occurs during a posted write to the VMEbus, the Universe II uses the AMERR field in the “VMEbus AM Code Error Log (V_AMERR)” on page 307 to log the AM code of the transaction, and the state of the IACK* signal, through the IACK bit, to indicate whether the error occurred during an IACK cycle. The FIFO entries for the 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 because logs are valid if the V_STAT bit is set.

The address of the errored transaction is latched in the “VMEbus Address Error Log (VAERR)” on page 308. 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 Generation and Handling” on page 109).

4. Slave Image Programming

This chapter describes the Slave Image Programming functionality of the Universe II. This chapter discusses the following topics:

4.1 Overview

The Universe II 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 II's register space is described in "Registers" on page 163. This section describes the slave images used to map transactions between the PCI bus and VMEbus.

4.2 VME Slave Image Programming

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 4: 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

Table 5: 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 6: 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

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 is written to the PCI bus but no further data is accepted into this FIFO until the bit is set.

IDT recommends that the attributes in a slave image not be changed while data is enqueued in the Posted Writes FIFO. To ensure data is queued from the FIFO, check the RXFE status bit in the “Miscellaneous Status Register (MISC_STAT)” on page 275 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.

4.2.1 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 fourth of these images (VMEbus slave image 0 and 4) have a 4-Kbyte resolution while VMEbus slave images 1 to 3 and 5 to 7 have 64-Kbyte resolution (maximum image size of 4 Gbytes).

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

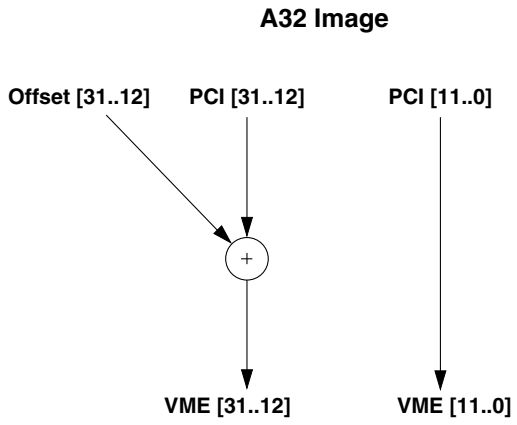
4.2.2 PCI Bus Fields

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 LAS 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 (see “VMEbus Read-Modify-Write Cycles (RMW Cycles)” on page 37). When the LLRMW bit is set, single cycle reads are 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.

Only accesses to PCI Memory Space are decoupled, accesses to I/O or Configuration Space are always coupled.

Figure 8: Address Translation Mechanism for VMEbus to PCI Bus Transfers



4.2.3 Control Fields

The control fields enable a VMEbus slave image (using the EN bit), as well as specify how reads and writes are 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 prefetches 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.

Prefetching is only possible in PCI Memory Space.

If the PWEN bit is set, incoming write data from the VMEbus is loaded into the RXFIFO (see “Posted Writes” on page 33). Note that posted write transactions can only be mapped to Memory space on the PCI bus. Setting the LAS 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 generates 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. If no 64-bit targets are expected on the PCI bus then performance can be improved by disabling LD64EN on the VMEbus slave images.

Universe II only performs 64-bit PCI transactions if the power-up option LCLSIZE bit in the “Miscellaneous Status Register (MISC_STAT)” on page 275 is set to 1. If the Universe II is set for a 32-bit PCI transaction (LCLSIZE bit set to 0) it does not perform 64-bit PCI transactions.

4.3 PCI Bus Target Images

The Universe II accepts accesses from the PCI bus with 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 Table 7, Table 8 and Table 9 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 is described in “Special PCI Target Image” on page 73.

Table 7: 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 8: 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 9: 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

IDT recommends that the attributes in a target image not be changed while data is enqueued in the Posted Writes FIFO. To ensure data is queued from the FIFO, check the TXFE status bit in the “Miscellaneous Status Register (MISC_STAT)” on page 275 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.

4.3.1 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 (PCI target images 0 and 4) have a 4 Kbyte resolution while PCI target images 1 to 3 and 5 to 7 have 64 Kbyte resolution. Typically, image 0 or image 4 would be used for an A16 image since they have the finest granularity.

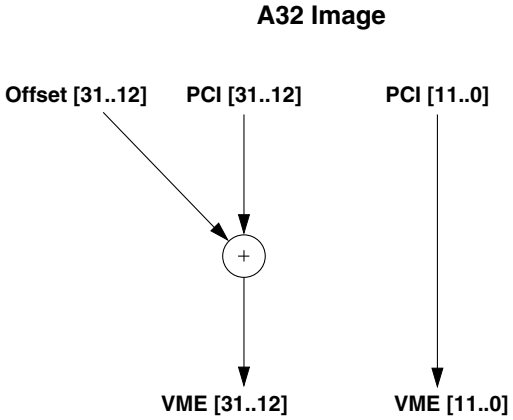
The address space of a VMEbus slave image must not overlap with the address space for the Universe II’s registers.

4.3.2 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. Figure 9 illustrates the translation process.

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



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

The Universe II provides support for user defined AM codes. The “User AM Codes Register (USER_AM)” on page 276 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 “PCI Target Image 0 Control (LSI0_CTL)” on page 181 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.

4.3.2.1 Transfers

Transfers appear on the VMEbus as 16-bit transfers when the Universe II is programmed in the following manner:

- PWEN= 1
- VDW=16-bit, 32-bit or 64-bit
- VCT= 0

External PCI master begins a burst 32-bit write with A2=0 and BE_=0011, followed by a transfer with BE_=1100.

These criteria optimize performance of 32-bit PCI systems which regularly perform 16-bit transfers. A series of 16-bit transfers is also performed if 64-bit posted write is received with BE_=11000011.

4.3.3 Control Fields

The control fields 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 performs posted writes when that particular PCI target image is accessed. Posted write transactions are only decoded within PCI Memory space. Accesses from I/O spaces results in coupled cycles independent of the setting of the PWEN bit.

4.4 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 (SLSI)” on page 208 is divided into four 16 Mbyte regions numbered 0 to 3 (see Figure 10 on page 74). 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 tables below.

Table 10: PCI Bus Fields for the Special PCI Target Image

Field	Register Bits	Description
Base	BS[5:0]	64 Mbyte aligned base address for the image
Address space	LAS	Places image in Memory or I/O

Table 11: VMEbus Fields for the Special PCI Bus Target Image

Field	Register Bits	Description
Maximum data width	VDW	Separately sets each region for 16 or 32 bits
Mode	SUPER	Separately sets each region as supervisor or non-privileged
Type	PGM	Separately sets each region as program or data

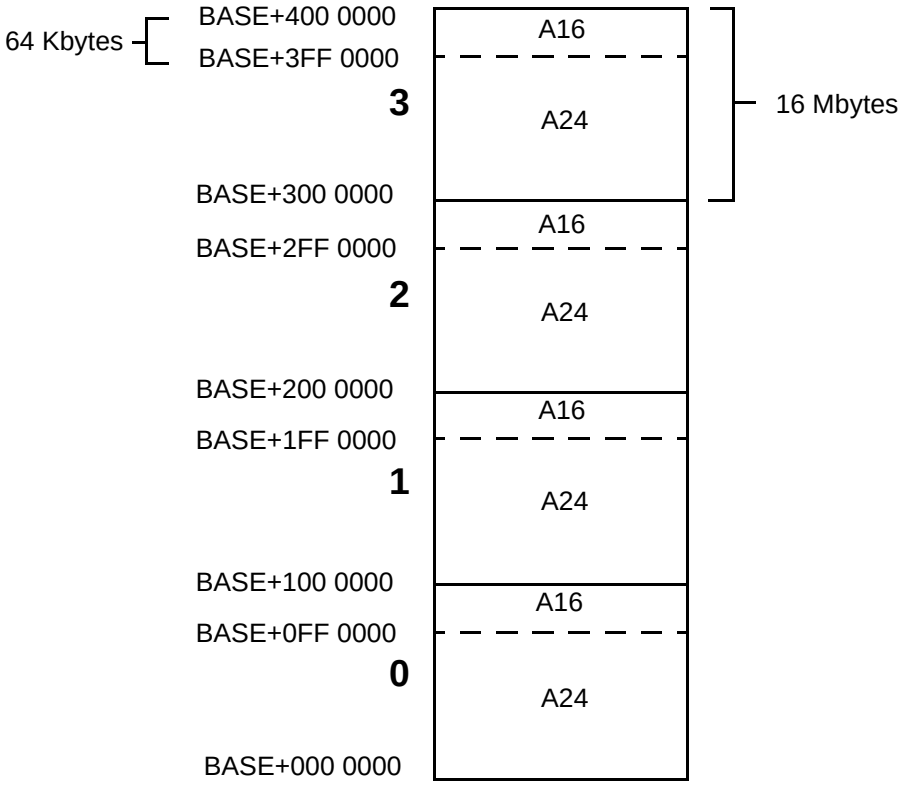
Table 12: Control Fields for the Special PCI Bus Target Image

Field	Register Bits	Description
Image enable	EN	Enable bit for the image
Posted write	PWEN	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 70), which are typically programmed to access A32 space.

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 10: Memory Mapping in the Special PCI Target Image



5. Registers Overview

The Universe II Control and Status Registers (UCSR) occupy 4 Kbytes of internal memory. This chapter discusses the following topics:

5.1 Overview

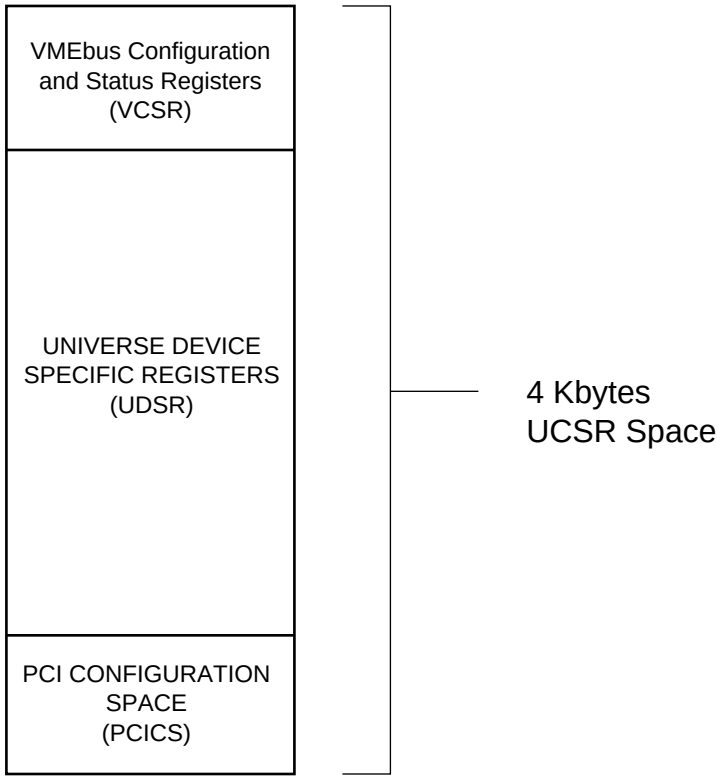
The Universe II Control and Status Registers (UCSR) occupy 4 Kbytes of internal memory. This 4 Kbytes is logically divided into the following three groups (see Figure 11):

- PCI Configuration Space (PCICS)
- Universe II Device Specific Registers (UDSR)
- 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 in the following sections.

Figure 11: Universe II Control and Status Register Space



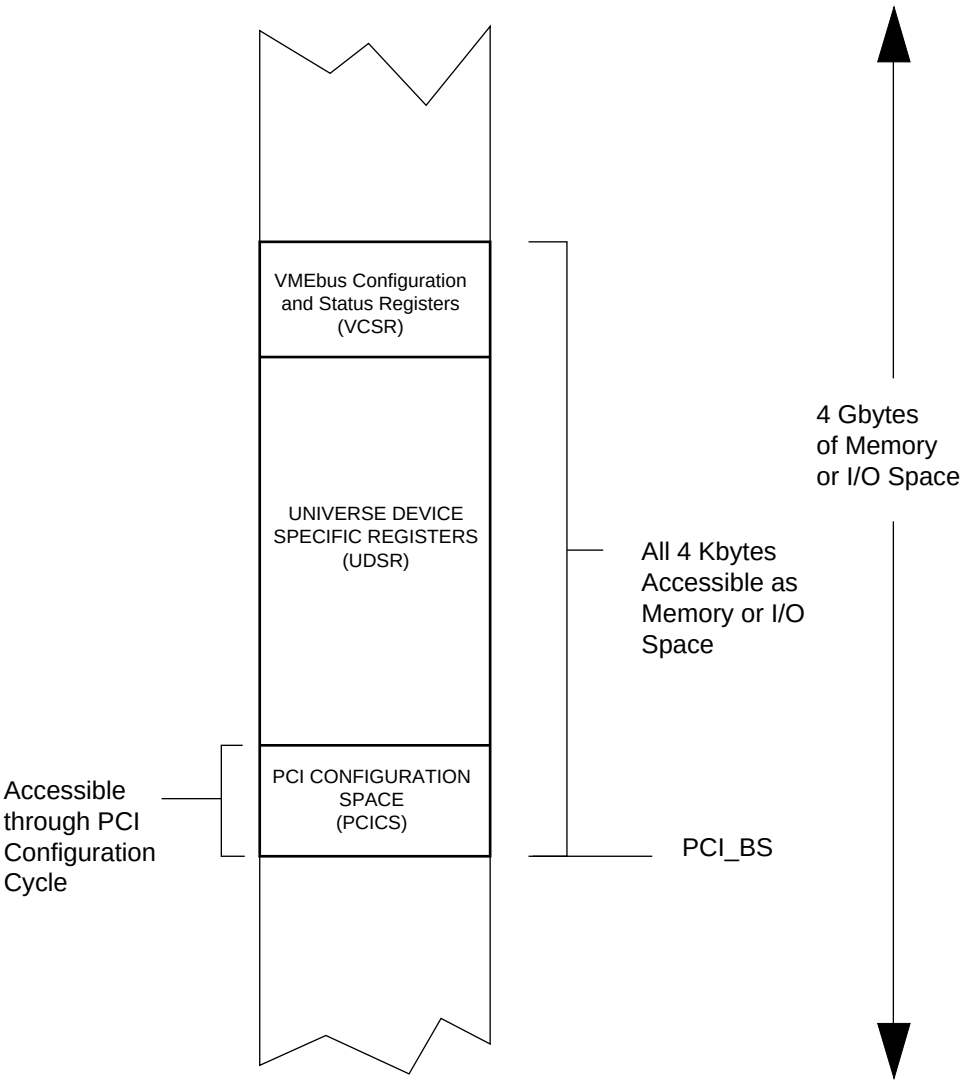
5.2 Register Access from the PCI Bus

There are different mechanisms to access the UCSR space from the PCI bus: Configuration space, PCI Memory or I/O space.

5.2.1 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 through 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 area 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 12 on page 77). Only the PCI configuration registers are accessible through PCI Configuration cycles.

Figure 12: PCI Bus Access to UCSR as Memory or I/O Space



5.2.2 Memory or I/O Access

Two 4-Kbyte ranges of addresses in PCI Memory space and/or PCI I/O space can be dedicated to the Universe II registers. There is one 4-Kbyte range in PCI Memory space and one 4-Kbyte range in PCI I/O space.

The Universe II has the following two programmable registers: “PCI Configuration Base Address Register (PCI_BS0)” on page 178 and “PCI Configuration Base Address 1 Register (PCI_BS1)” on page 179. These register 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 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 and do not accept burst writes.

5.2.2.1 Conditions of Target-Retry

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

- While UCSR space is being accessed by a VMEbus master, PCI masters are retried.
- If a VMEbus master is performing a RMW access to the UCSRs then PCI attempts to access the USCR space results 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 is retried until BBSY* is negated.

5.2.3 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 does 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*.

5.3 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 can 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.

5.3.1 VMEbus Register Access Image (VRAI)

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 13, and Figure 13). All registers are accessed as address offsets from the VRAI base address programmed in the “VMEbus Register Access Image Base Address Register (VRAI_BS)” on page 304. The image can be enabled or disabled using the EN bit in the“VMEbus Register Access Image Control Register (VRAI_CTL)” on page 303.

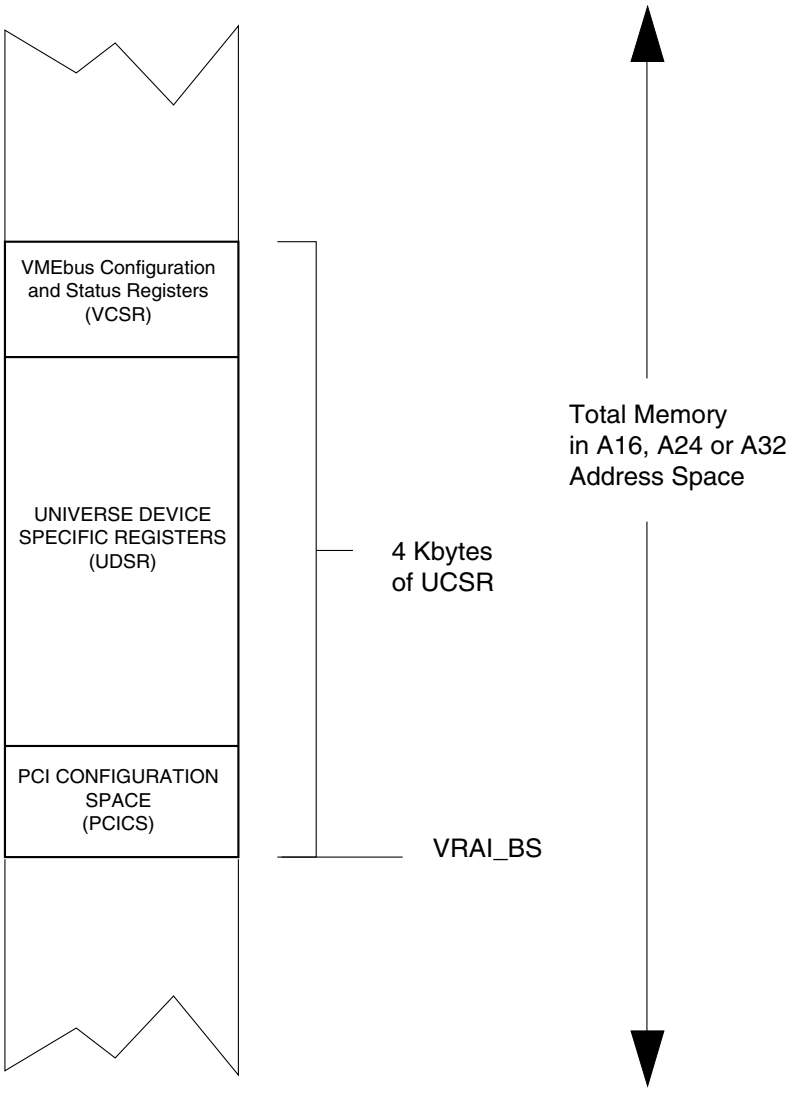
The VMEbus register access image is defined by Table 13.

Table 13: Programming the VMEbus Register Access Image

Field	Register Bits	Description
Address space	VAS in the VRAI_CTL	One of A16, A24, A32
Base address	BS[31:12] in the VRAI_BSi	Lowest address in the 4Kbyte slave image
Slave image enable	EN in the VRAI_CTL	Enables VMEbus register access image
Mode	SUPER in the VRAI_CTL	Supervisor and/or Non-Privileged
Type	PGM in the VRAI_CTL	Program and/or Data

Note that the VRAI base address can be configured as a power-up option (see “Resets, Clocks and Power-up Options” on page 129).

Figure 13: UCSR Access from the VMEbus Register Access Image



5.3.2 CR/CSR Accesses

The *VME64 Specification* assigns a total of 16 Mbytes of CR/CSR space for the VMEbus system. The CR/CSR image is enabled with the EN bit in the “VMEbus CSR Control Register (VCSR_CTL)” on page 305. 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 14). The base address of the CR/CSR space allocated to the Universe II’s slot is programmed in the “VMEbus CSR Base Address Register (VCSR_BS)” on page 331.

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 which is programmed in the “VMEbus CSR Translation Offset (VCSR_TO)” on page 306.

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 34 on page 164). When accessing the UCSR in CR/CSR space, the first register has 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 34 on page 164.

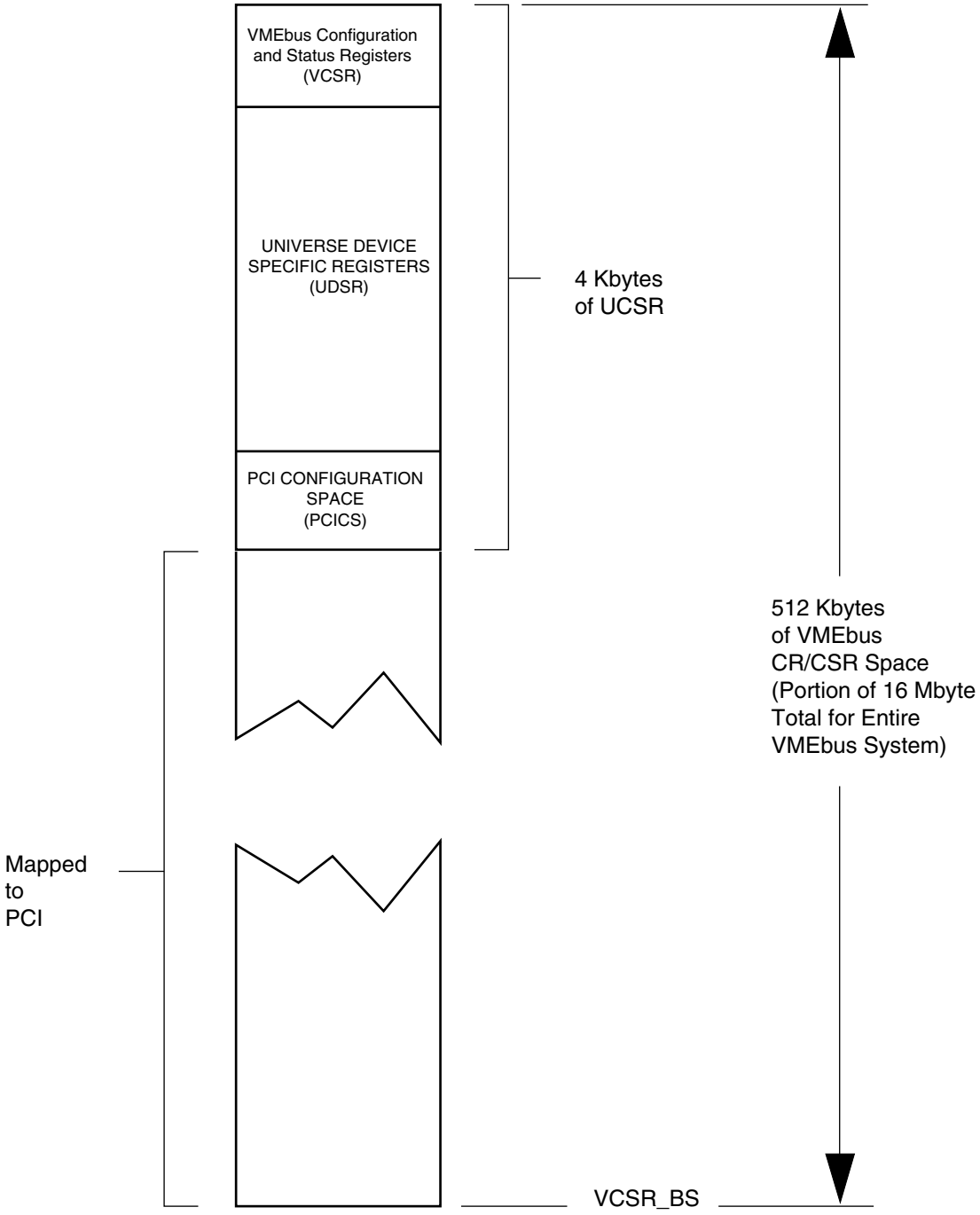
5.3.3 RMW and ADOH Register Access Cycles

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 ADOH 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 14: UCSR Access in VMEbus CR/CSR Space



5.4 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 “Mailbox 0 Register (MBOX0)” on page 265 to “Mailbox 3 Register (MBOX3)” on page 268). 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 Generation and Handling” on page 109 and “Mailbox Register Access Interrupts” on page 122.

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.

5.5 Semaphores

The Universe II has two general-purpose semaphore registers each containing four semaphores (“Semaphore 0 Register (SEMA0)” on page 269 and “Semaphore 1 Register (SEMA1)” on page 270). To gain ownership of a semaphore, a process writes a logic 1 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 shares 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 (“Special Cycle Generator” on page 61). 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.

Each of the four semaphores in a semaphore register are intended to be accessed with 8-bit transfers.

6. DMA Controller

Direct memory access (DMA) allows a transaction to occur between two devices without involving the host processor (for example, a read transaction between a peripheral device and host processor memory). Because less time is required to complete transactions, applications that contain one or more DMA channels support faster read and write transfers than applications that support only host-assisted transactions.

This chapter discusses the following topics:

6.1 Overview

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 can be linked to another command packet, so 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.

6.2 DMA Registers

The DMA registers reside in a register block starting at offset 0x200. The registers describe the following information for a single DMA transfer:

- Source: where to transfer data from
- Destination: where to transfer data to
- Size: how much data to transfer
- Attributes: 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 are programmed directly. 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 Mode” on page 96).

6.2.1 Source and Destination Addresses

The source and destination addresses for the DMA reside in two registers: the “DMA PCI Bus Address Register (DLA)” on page 235), and the “DMA VMEbus Address Register (DVA)” on page 236. The determination of which is the source address, and which is the destination is made by the L2V bit in the “DMA Transfer Control Register (DCTL)” on page 232. When set, the DMA transfers data from the PCI to the VMEbus. The 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 can 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 “DMA Transfer Control Register (DCTL)” on page 232. The sub-spaces are programmed in the PGM and SUPER fields of the same register.

Although the PCI and VMEbus addresses may be programmed to any byte aligned address, they must be 8-byte aligned to each other (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 does not start. It sets the protocol error bit (P_ERR) in the “DMA General Control/Status Register (DGCS)” on page 238 and,if enabled to, generates an interrupt. When this occurs linked-list operations 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 the DMA is stopped, halted, or at the completion of processing a command packet. If read during DMA activity, they 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 must not be used, but the DTBC is valid (see “DMA Error Handling” on page 105 for details). At the end of a successful linked-list transfer, the DVA and DLA registers point to the next address at the end of the transfer block, and the DTBC register is 0.

6.2.2 Non-incrementing DMA Mode

The VMEbus Non-Incrementing Mode (Non-Inc Mode) enables the DMA Controller to perform transfers to or from a fixed VMEbus address. This means that the specified VMEbus address is not incremented during DMA reads or writes. This applies to both Direct and Linked List modes of DMA operation. For more information on these two types of DMA operation, refer to “Direct Mode Operation” on page 93 and “Linked-list Mode” on page 96.

Unlike incrementing DMA operation, in Non-Inc Mode the DMA Controller can only perform 8-,16- or 32-bit single cycle transfers on the VMEbus. This means that BLT and MBLT transfers cannot be performed when operating in Non-Inc Mode.

6.2.2.1 Using Non-Inc Mode

The VMEbus Non-Inc Mode is enabled by writing a 1 to the NO_VINC bit in the “DMA Transfer Control Register (DCTL)” on page 232.

In order to set-up and initiate DMA operation, the same steps which are described in the DMA Controller Section in the Universe II User Manual must be followed for Non-Inc Mode.

The steps in setting-up and initiating DMA operation are as follows:

1. Program the tenure and interrupt requirements in the DGCS register (offset 0x220).
2. Program the source and destination addresses in the DLA and DVA registers.
3. Set the GO bit in the DGCS register.

6.2.2.2 Issues with Non-Inc Mode

The VMEbus Address

In Non-Inc Mode, the DVA register (offset 0x210) does not necessarily contain the fixed VMEbus address. This register must not be read during a DMA Non-Inc Mode transfer. Once a DMA transfer has been stopped — by setting the STOP bit of the DGCS register — the Non-Inc Mode transfer cannot be restarted by simply writing a 1 to the GO bit of the DGCS register. The DVA register must be reprogrammed with the required address before setting the DGCS GO bit.

The VON Counter

When the VON counter in the DGCS register reaches its programmed limit, the VMEbus Master Interface of the Universe II stops transferring data until the VOFF timer expires. If the device is operating in Non-Inc Mode, the VON counter has different limits than those indicated in the DMA Controller section.

The different settings are detailed in Table 14.

Table 14: VON Settings for Non-Inc Mode

VON	VMEbus Aligned DMA Transfer Count
001	128 bytes
010	256 bytes
011	512 bytes
100	1024 bytes
101	2048 bytes
110	4096 bytes
111	8192 bytes

P_ERR Flag Behavior

When the GO bit is set in Non-Inc Mode, the P_ERR flag of the “DMA General Control/Status Register (DGCS)” on page 238 is 1 when the following conditions are true:

- VCT bit of the “DMA Transfer Control Register (DCTL)” on page 232 has a value of 1
- VDW field of the DCTL register has a value of 01 and bit 0 of the DVA register is a value of 0
- VDW field of the DCTL register has a value of 10 and bits 1 and 0 of the DVA register are non-zero
- VDW field of the DCTL register has a value of 11.

Single Cycle Transfers

The Universe II performs 8-,16- or 32-bit single cycle transfers on the VMEbus. BLT and MBLT transfers cannot be performed when operating in Non-Inc Mode.

6.2.2.3 Non-Inc Mode Performance

The transfer performance of DMA in Non-Inc Mode has been simulated at 14 MB/s for 32-bit writes and at 8 MB/s for 32-bit reads. This performance was determined using ideal slave responses; lower performance can be expected in actual systems.

6.2.3 Transfer Size

The DMA can be programmed through the “DMA Transfer Byte Count Register (DTBC)” on page 234 to transfer any number of bytes from 1 byte to 16 MBytes. There are no alignment requirements to the source or destination addresses. If the width of the data turnovers (8- through 64-bit on VMEbus and 32- or 64-bit on PCI) do not align to the length of the transfer or the source/destination addresses, the DMA inserts 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 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 (DMA is indicated as active by the ACT bit in the DGCS register). At the end of a transfer it contains 0. However, if stopped by the user (through 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 105).

Starting the DMA while DTBC is 0 results in one of two situations. If the CHAIN bit in the “DMA General Control/Status Register (DGCS)” on page 238 is not set, the DMA does not start; it performs 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“DMA Command Packet Pointer (DCPP)” on page 237, 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).

6.2.4 Transfer Data Width

The VMEbus and PCI bus data widths are determined by three fields in the “DMA Transfer Control Register (DCTL)” on page 232. 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)
- D8BLT
- 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 can 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 performs a D08 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 also performs 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 can 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 "DMA Transfer Control Register (DCTL)" on page 232 If the Universe II has powered-up on a 32-bit bus (see "Power-Up Options" on page 135), this bit has 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 2.1 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.

6.2.5 DMA Command Packet Pointer

The "DMA Command Packet Pointer (DCPP)" on page 237 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 0 when the GO bit is set) the DMA reads the 32-byte command packet from PCI memory and executes the transfer it describes.

6.2.6 DMA Control and Status

"DMA General Control/Status Register (DGCS)" on page 238 contains a number of fields that control initiation and operation of the DMA as well as actions to be taken on completion.

6.2.6.1 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 describe 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 Configuration Space Control and Status Register (PCI_CSR)" on page 172, 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 "DMA General Control/Status Register (DGCS)" on page 238 and, if enabled, an interrupt is generated.

If the DMA has been terminated (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 only has an effect if all status bits have been cleared. These bits include STOP, HALT, DONE, LERR, VERR, and P_ERR and are located in the "DMA General Control/Status Register (DGCS)" on page 238. These bits are all cleared by writing 1 to them, either before or while setting the GO bit.

The GO bit always returns a 0 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.

6.2.6.2 DMA VMEbus Ownership

Two fields in the DGCS register determine how the DMA shares 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

VON affects how much data the DMA transfers 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 stops 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 performs an early release of BBSY* when the VON counter reaches its programmed limit. VON may be disabled by setting the field to zero. When the VON bit is set to 0, the DMA continues 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 101 for details on the VMEbus request and release conditions for the DMA.

VOFF

VOFF affects how long the DMA waits before re-requesting the bus after the VON limit has been reached. By setting VOFF to 0, the DMA immediately re-requests 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 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 (for example, the DMAFIFO has become 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 101 for details on the VMEbus request and release conditions for the DMA.

See “DMA Channel Interactions with Other Channels” on page 105 for information on other mechanisms which can delay the DMA Channel from acquiring the VMEbus or the PCI bus.

6.2.6.3 DMA Completion and Termination

Normally, the DMA continues processing its transfers and command packets until either it completes all requests, 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.

STOP

When the STOP_REQ bit in the DGCS register is set, 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 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 re-starting reads, then the request remains pending until the VOFF timer has expired and the bus has been granted.

HALT

HALT provides a mechanism to interrupt the DMA at command packet boundaries during a linked-list mode 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 be cleared.

Regardless of how the DMA stops—whether normal, bus error or user interrupted—the DMA indicates in the DGCS register why it stopped. The STOP and HALT bits are 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 86). Before the DMA can be restarted, each of these status bits must be cleared.

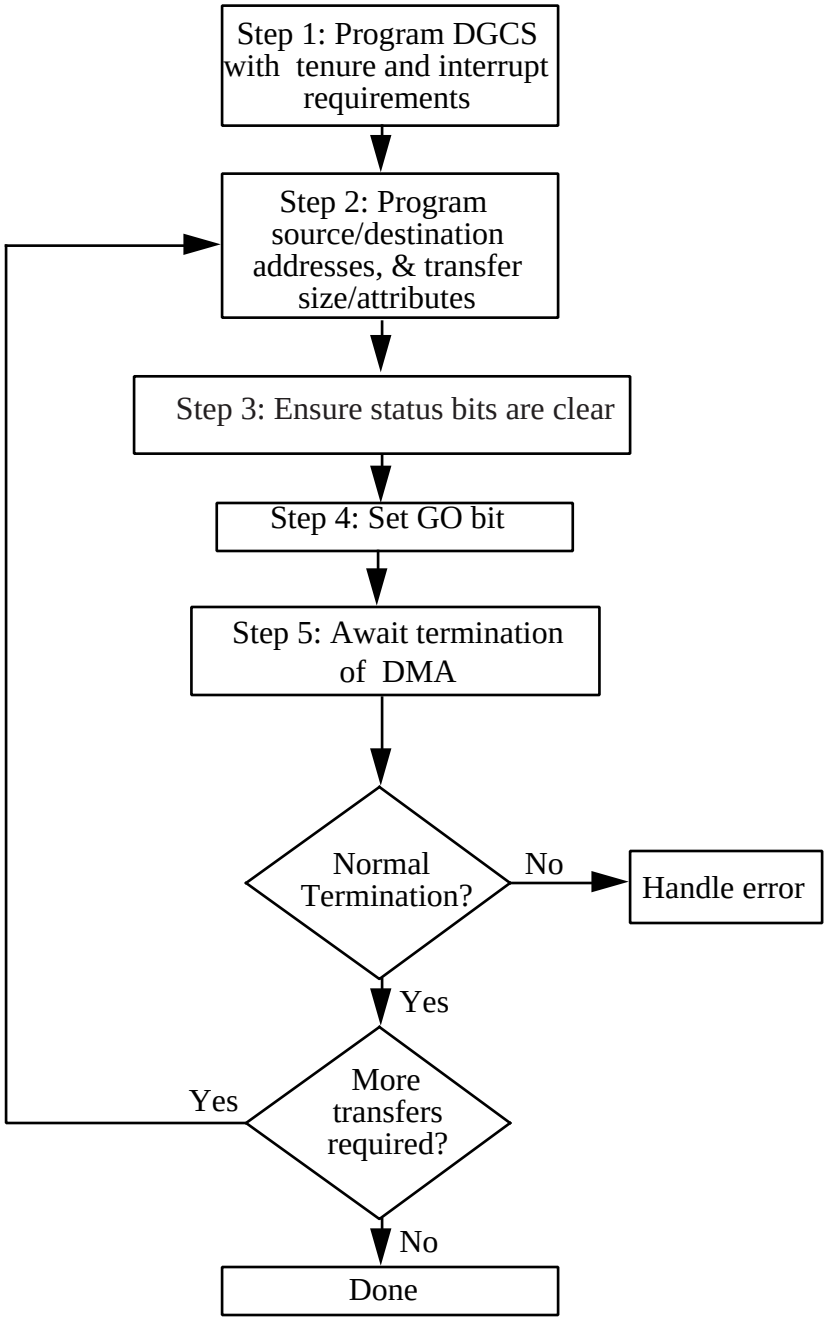
When the DMA terminates, an interrupt may be generated to VMEbus or PCI bus. The user has control over which DMA termination conditions cause the interrupt through the INT_STOP, INT_HALT, INT_DONE, INT_LERR, INT_VERR, and INT_P_ERR bits in the DGCS register.

6.3 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 15 describes the steps involved in operating the DMA in direct mode.

Figure 15: Direct Mode DMA transfers



In Step 1, the “DMA General Control/Status Register (DGCS)” on page 238is 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 111 and “VMEbus Interrupt Generation” on page 113 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 can 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, the DMA waits for termination of the DMA transfers.

— The DMA continues with the transfers until it:

  – Completes all transfers

  – Terminates early with the STOP_REQ bit

  – Encounters an error on the PCI bus or VMEbus

Each of these conditions 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 is 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 ensures 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 stops 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 has 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 must 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 105). If the STOP bit was set, the software must take whatever actions were programmed 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.

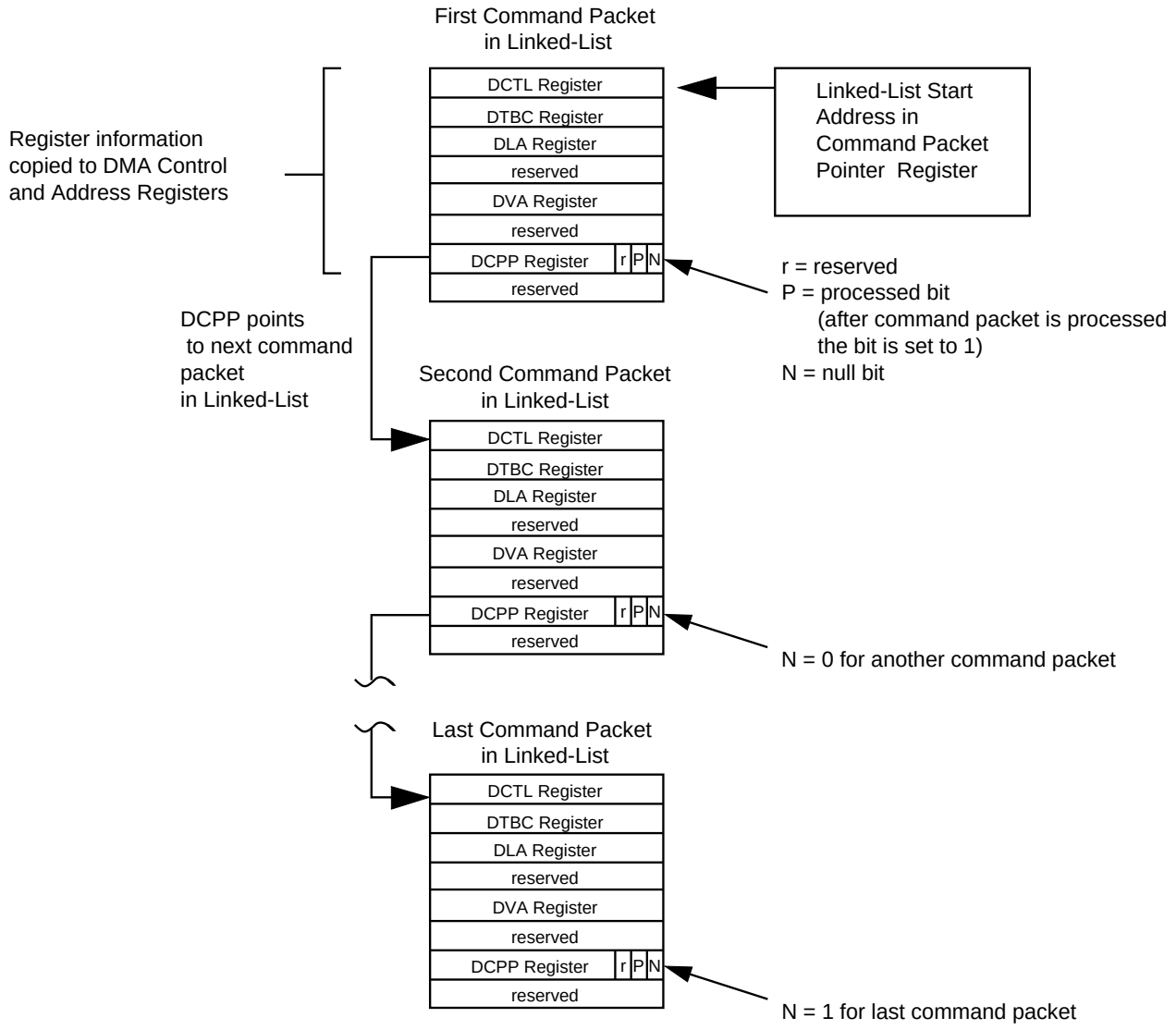
If the DONE bit was set, it indicates that the DMA completed its requested transfer successfully, and if more transfers are required, the software can proceed to Step 2 to start a new transfer.

6.4 Linked-list Mode

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 16 below), 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. The PROCESSED bit must be initially set for 0. 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.

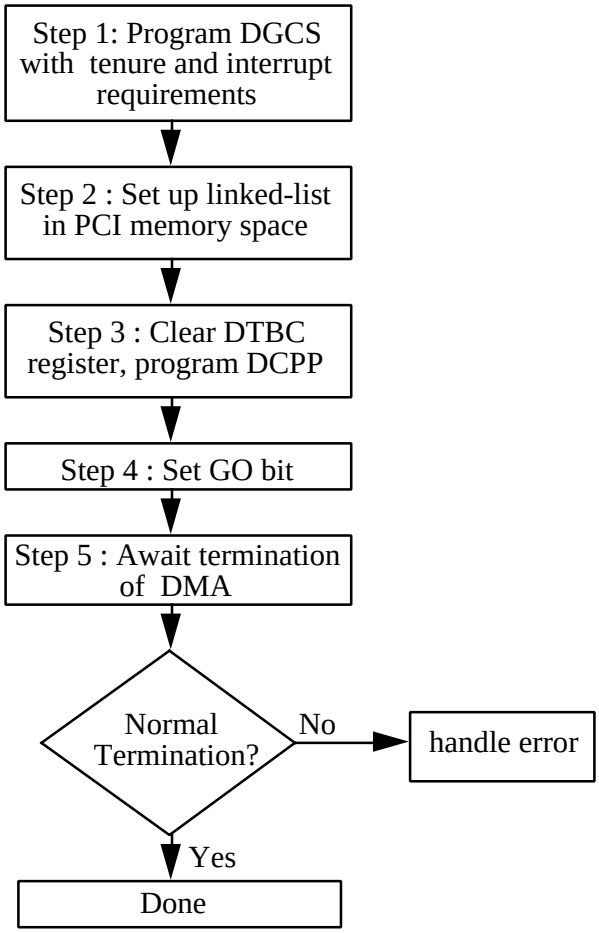
Figure 16: Command Packet Structure and Linked List Operation



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

Figure 17: DMA 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 111 and “VMEbus Interrupt Generation” on page 113).

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.

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 cannot read the first command packet but instead performs 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 first performs the transfers defined by the current contents of the DCTL, DTBC, DVA and DLA registers. Once that is complete it then starts 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 16). The DMA terminate when one of the following conditions are met:

  – Processes a command packet with the NULL bit set indicating the last packet of the list
  – Stops with the STOP_REQ bit in the DGCS register
  – Halts with the HALT_REQ bit in the DGCS register
  – Encounters an error on either the PCI bus or VMEbus

Each of these conditions 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 is 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 ensures that the DMA has not been hung up by a busy VMEbus, or other such system issues.

Linked-list mode can be halted by setting the HALT_REQ bit in the “DMA Transfer Control Register (DCTL)” on page 232. 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 µs after the HALT bit is set: therefore after a DMA halt the user should wait at least 1 µs before checking the status of the PROCESSED bit.

The DMA can be restarted by clearing the HALT status bit and setting the GO bit 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 (that is, the Universe II processes the current command packet (see Figure 16)).

In contrast to a halt, the DMA can also be immediately terminated through the STOP_REQ bit. This stops 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 causes 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 can 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 performs the transfers in the order set in the linked list. For more details on updating the linked list see “Linked-list Updating” on page 100.

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.

6.4.1 Linked-list Updating

The Universe II provides a mechanism which enables 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 “DMA Linked List Update Enable Register (D_LLUE)” on page 241. 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. 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 encounters 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 is accomplished by the following steps:

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 is in one of the following conditions:

1. It can be active and working its way through the linked list. In this case, no further steps are required.
2. The DMA can 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 points 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 have not been created ahead of time, the DCPP register can not be programmed to any valid packet, and needs programming to the newly programmed packet.
3. The DMA has encountered an error. In this circumstance, see “DMA Error Handling” on page 105 for how to handle DMA errors.

Operation can 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 is 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 has no effect for case 1 since the DMA is still active, and restarts 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 is not be sufficient to restart the DMA—the error bits in the DGCS register also has to be cleared before operation can continue.

6.5 FIFO Operation and Bus Ownership

The DMA uses a 256-byte, 64-bit FIFO (DMAFIFO). 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.

6.5.1 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 “Master Control Register (MAST_CTL)” on page 271). 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 can fill frequently during PCI to VMEbus transfers, though the depth of the FIFO helps to minimize this situation.

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.

The DMA can prefetch extra read data from the external PCI target. This means that the DMA must only be used with memory on the PCI bus which has no adverse side-effects when prefetched. The Universe II prefetches 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 “DMA Transfer Byte Count Register (DTBC)” on page 234 are 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 fetches the full eight bytes, and nothing more. Programming the DTBC to less than eight bytes still results 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 25 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 one of the following conditions are met:

- DMAFIFO is empty
- DMA block is complete
- DMA is stopped
- a linked list is halted
- DMA encounters an error
- 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 “DMA General Control/Status Register (DGCS)” on page 238. With VON enabled, the DMA relinquishes ownership of the Master Interface at defined address boundaries. See “DMA VMEbus Ownership” on page 91.

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 91.

The DMA Channel unpacks the 64-bit data queued in the DMAFIFO to whatever the programmed transfer width is on the VMEbus (D16, D32, or D64). The VMEbus Master Interface delivers the data in the DMAFIFO according to the VMEbus cycle type programmed into the “DMA Transfer Control Register (DCTL)” on page 232. The DMA provides data to the VMEbus until one of the following conditions are met:

- DMA FIFO empties
- DMA VMEbus Tenure Byte Count expires.
  - — Set in the VON in the “DMA General Control/Status Register (DGCS)” on page 238

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). The 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 27).

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-bit. 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 can 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 52).

6.5.2 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 requests the VMEbus (through the VMEbus Master Interface) and begins reading data from the VMEbus. The Universe II maintains VMEbus ownership until one of the following conditions are met:

- DMAFIFO is full
- DMA block is complete
- DMA is stopped
- a linked list is halted
- DMA encounters an error
- 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 “DMA General Control/Status Register (DGCS)” on page 238. With VON enabled, the DMA will relinquish ownership of the Master Interface at defined address boundaries (see “DMA VMEbus Ownership” on page 91).

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 91.

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 (three entries remaining) the DMA requests that the VMEbus Master Interface complete its pending operations and stop. The pending read operations 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 can 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 57).

6.6 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 through 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 can be individually enabled, and are listed in Table 15. Setting the enable bit enables the corresponding interrupt source.

Table 15: 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 the corresponding DMA bit in the “PCI Interrupt Status Register (LINT_STAT)” on page 244 and “VMEbus Interrupt Status Register (VINT_STAT)” on page 251 are set — regardless of whether the LINT_EN or VINT_EN enable bits have been set. Each one must be cleared independently. Clearing either the LINT_STAT or VINT_STAT registers does not clear the other. See “Interrupt Generation and Handling” on page 109.

6.7 DMA Channel Interactions with Other Channels

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

The Universe II does not apply *PCI 2.1 Specification* transaction ordering requirements to the DMA Controller. 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 does not perform transfers on the PCI bus until the Universe II is unlocked (see “VMEbus Lock Commands (ADOH Cycles)” on page 36). When an external VMEbus Master begins a RMW cycle, at some point a read cycle appears 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 occurs on the PCI bus (see “VMEbus Read-Modify-Write Cycles (RMW Cycles)” on page 37).

If the PCI Target Channel locks the VMEbus using VOWN, no DMA transfers takes place on the VMEbus (see “Using the VOWN bit” on page 64).

6.8 DMA Error Handling

This section describes how the Universe II responds to errors involving the DMA, and how the user can recover from them. 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.

6.8.1 DMA Software Response to Error

While the DMA is operating normally, the ACT bit in the “DMA General Control/Status Register (DGCS)” on page 238. Once the DMA has terminated, it clears this bit, and sets 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 DMA is interrupted, either the STOP or HALT bits are set. If an error has occurred, one of the remaining three bits, LERR, VERR, or P_ERR, is 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 104).

- LERR is set if the DMA encounters an error on the PCI bus (either a Master-Abort or Target-Abort)
  - — Bits in the “PCI Configuration Space Control and Status Register (PCI_CSR)” on page 172 indicate which of these conditions caused the error.

- VERR is set if the DMA encounters a bus error on the VMEbus. This is 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 86).

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.

6.8.2 DMA Hardware Response to Error

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 104).

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).

6.8.2.1 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.

6.8.3 Resuming DMA Transfers

When a DMA error occurs (on the source or destination bus), the status bits must be read in order to 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 (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 can 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).
3. If the difference between the value contained in the DTBC register and the original value is less than 512 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 512 bytes (the FIFO depth of the Universe II), add 512 bytes 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 90).

7. Interrupt Generation and Handling

An interrupt is a signal informing a program that an event (for example, an error) has occurred. When a program receives an interrupt signal, it temporarily suspends normal processing and diverts the execution of instructions to a sub-routine handled by an interrupt controller. The controller communicates with the host processor and the device that initiated the interrupt to determine how to handle the interrupt.

Interrupt signals can come from a variety of sources. Interrupt signals generated by devices (for example, a printer) indicate an event has occurred and are called hardware interrupts. Interrupt signals generated by programs are called software interrupts.

This chapter discussed the interrupt generation and handling functionality of the Universe II. This chapter discusses the following topics:

- “Interrupt Generation” on page 111
- “Interrupt Handling” on page 116

7.1 Overview

The Universe II has two types of interrupt capability: it is a generator of interrupts and an interrupt handler.

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

- PCI LINT_[7:0]
- VMEbus IRQ*[7:1]
- 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. The VMEbus sources, VIRQ[7:1], are not mapped to the PCI bus interrupts. However, the VIRQ[7:1] bits are status bits which indicate whether or not a STATUS/ID vector has been acquired. This indication can be used to generate an interrupt on the PCI bus. Some internal sources (for example, error conditions or DMA activity) can be mapped to either bus.

Figure 18: Universe Interrupt Circuitry

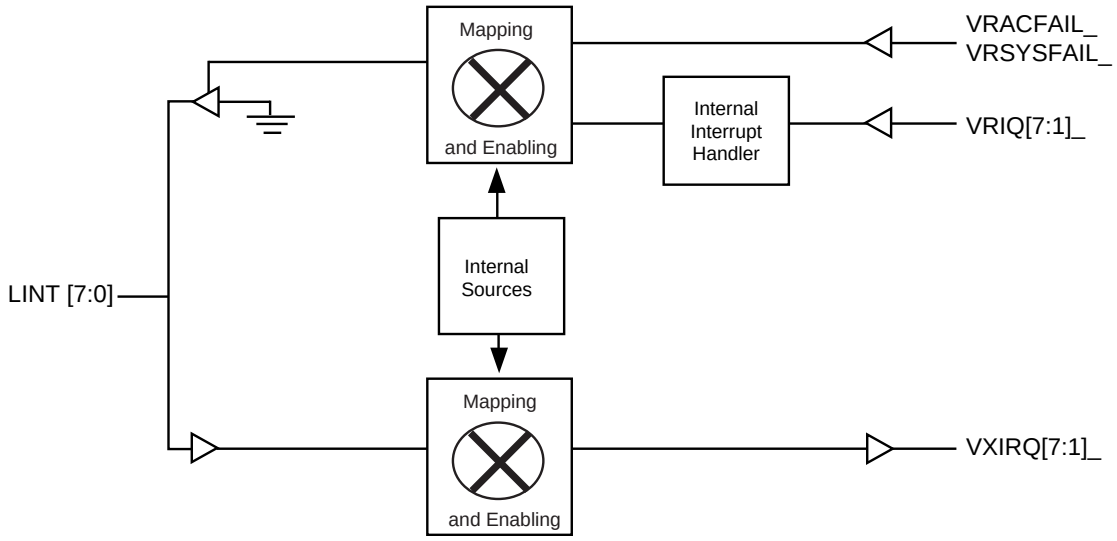


Figure 18 illustrates the circuitry inside the Universe II Interrupt Channel. The PCI hardware interrupts are listed on the left, and the VMEbus interrupt inputs and outputs are on the right. Internal interrupts are also illustrated. The figure shows that the interrupt sources may be mapped and enabled. 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.)

7.2 Interrupt Generation

The Universe II has the ability to generate interrupts on both the PCI bus and VMEbus.

7.2.1 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, all interrupt sources can be routed to a single PCI interrupt output.

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 63)
  - Software interrupt
  - Mailbox access
  - Location monitor access
  - VMEbus IACK cycle performed in response to a software interrupt

Each source can be individually enabled in the “PCI Interrupt Enable Register (LINT_EN)” on page 242 and mapped to a single LINT_ signal through the “PCI Interrupt Map 0 Register (LINT_MAP0)” on page 246, “PCI Interrupt Map 1 Register (LINT_MAP1)” on page 247, and “PCI Interrupt Map 2 Register (LINT_MAP2)” on page 263. 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 “PCI Interrupt Status Register (LINT_STAT)” on page 244. See Table 17 on page 114 for a list of the enable, mapping and status bits for PCI interrupt sources.

Table 16: Source, Enabling, Mapping, and Status of PCI Interrupt Output

Interrupt Source	Enable Bit in LINT_EN Register	Mapping Field in LINT_MAPx Register	Status Bit in LINT_STAT Register
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 occurred during a posted write	VERR	VERR	VERR
PCI Target-Abort or Master-Abort occurred during a posted write	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 “PCI Interrupt Status Register (LINT_STAT)” on page 244 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 PCI clock at their inputs.

All other sources of PCI interrupts are edge-sensitive. 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 116).

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 1 to the appropriate status bit.

7.2.2 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 "VMEbus Interrupt Enable Register (VINT_EN)" on page 248 and mapped to a particular VMEbus Interrupt level using the "VME Interrupt Map 0 Register (VINT_MAP0)" on page 253, "VME Interrupt Map 1 Register (VINT_MAP1)" on page 254, and "VME Interrupt Map 2 Register (VINT_MAP2)" on page 264. 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"VMEbus Interrupt Status Register (VINT_STAT)" on page 251, 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 "Interrupt STATUS/ID Out Register (STATID)" on page 255, unless it is configured as SYSCON in which case it does not monitor IACKIN*. See Table 17 for a list of the enable, mapping and status bits for VMEbus interrupt sources.

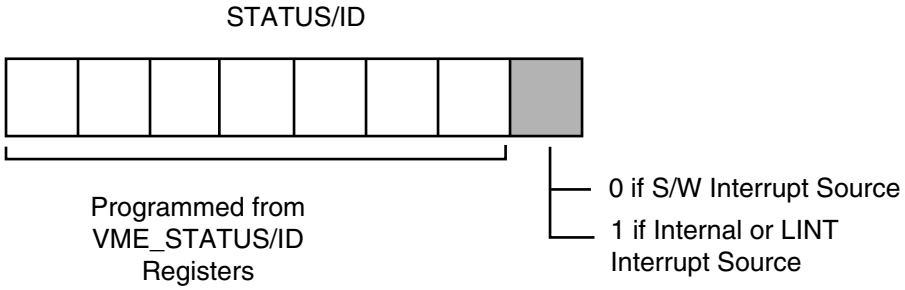
Table 17: Source, Enabling, Mapping, and Status of VMEbus Interrupt Outputs

Interrupt Source	Enable Bit in the "VMEbus Interrupt Enable Register (VINT_EN)" on page 248	Mapping Field in VINT_MAPx	Status Bit in the "VMEbus Interrupt Status Register (VINT_STAT)" on page 251
VMEbus Software Interrupt	SW_INT7-1	N/A[a]	SW_INT7-1
VMEbus Error	VERR	VERR in the "VME Interrupt Map 1 Register (VINT_MAP1)" on page 254	VERR
PCI Target-Abort or Master-Abort	LERR	LERR in the "VME Interrupt Map 1 Register (VINT_MAP1)" on page 254	LERR
DMA Event	DMA	DMA in the "VME Interrupt Map 1 Register (VINT_MAP1)" on page 254	DMA
Mailbox Register	MBOX3-0	MBOX3-0 in the "VME Interrupt Map 2 Register (VINT_MAP2)" on page 264	MBOX3-0
PCI bus Interrupt Input	LINT7-0	LINT7-0 in the "VME Interrupt Map 0 Register (VINT_MAP0)" on page 253	LINT7-0
VMEbus Software Interrupt (mappable)	SW_INT	SW_INT in the "VME Interrupt Map 1 Register (VINT_MAP1)" on page 254	SW_INT

a. This set of software interrupts cannot be mapped. That is, setting the SW_INT1 bit triggers VXIRQ1, setting the SW_INT2 bit triggers VXIRQ2, etc.

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 19: STATUS/ID Provided by Universe II



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 1 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. 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.

7.3 Interrupt Handling

The Universe II can handle interrupts from both the PCI bus and the VMEbus.

7.3.1 PCI Interrupt Handling

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 “VMEbus Interrupt Enable Register (VINT_EN)” on page 248, 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 clears the VMEbus interrupt by first clearing the source of the interrupt on the PCI bus, and then clearing the VMEbus interrupt (by writing a 1 to the appropriate bit in the “VMEbus Interrupt Status Register (VINT_STAT)” on page 251.

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.

7.3.2 VMEbus Interrupt Handling

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 “PCI Interrupt Enable Register (LINT_EN)” on page 242.

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 “Master Control Register (MAST_CTL)” on page 271 (see “VMEbus Requester” on page 25). 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. If an interrupt at a higher priority is asserted on the VMEbus before BBSY* is asserted the Universe II performs an interrupt acknowledge for the first interrupt it detected. Upon completion of that IACK cycle, the Universe II then performs 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, “VIRQ1 STATUS/ID Register (V1_STATID)” on page 256 through “VIRQ7 STATUS/ID Register (V7_STATID)” on page 262, 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 is not run on that level until the level has been cleared by writing a 1 to the corresponding status bit in the “PCI Interrupt Status Register (LINT_STAT)” on page 244. If other interrupts (at different levels) are pending while the interrupt is waiting to be cleared, 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 “PCI Interrupt Map 0 Register (LINT_MAP0)” on page 246. The interrupt is cleared and the VMEbus interrupt level is re-armed by clearing the correct bit in the “PCI Interrupt Status Register (LINT_STAT)” on page 244 register.

7.3.2.1 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 “Error Handling” on page 125). If the error occurs during an IACK cycle, the IACK_ bit is set in the “VMEbus AM Code Error Log (V_AMERR)” on page 307. 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 “Interrupt STATUS/ID Out Register (STATID)” on page 255, and generates an interrupt on the appropriate LINT_ pin (as mapped in the “PCI Interrupt Map 0 Register (LINT_MAP0)” on page 246). The PCI resource, upon receiving the PCI interrupt, is expected to read the“Interrupt STATUS/ID Out Register (STATID)” on page 255, 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 can generate two PCI interrupts: one through the VMEbus master bus error interrupt and another through the standard PCI interrupt translation. If an error occur during acquisition of a STATUS/ID, the “PCI Interrupt Status Register (LINT_STAT)” on page 244 shows that both VIRQx, and VERR are active.

7.3.3 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. Table 18 shows to which bus each interrupt source can be routed.

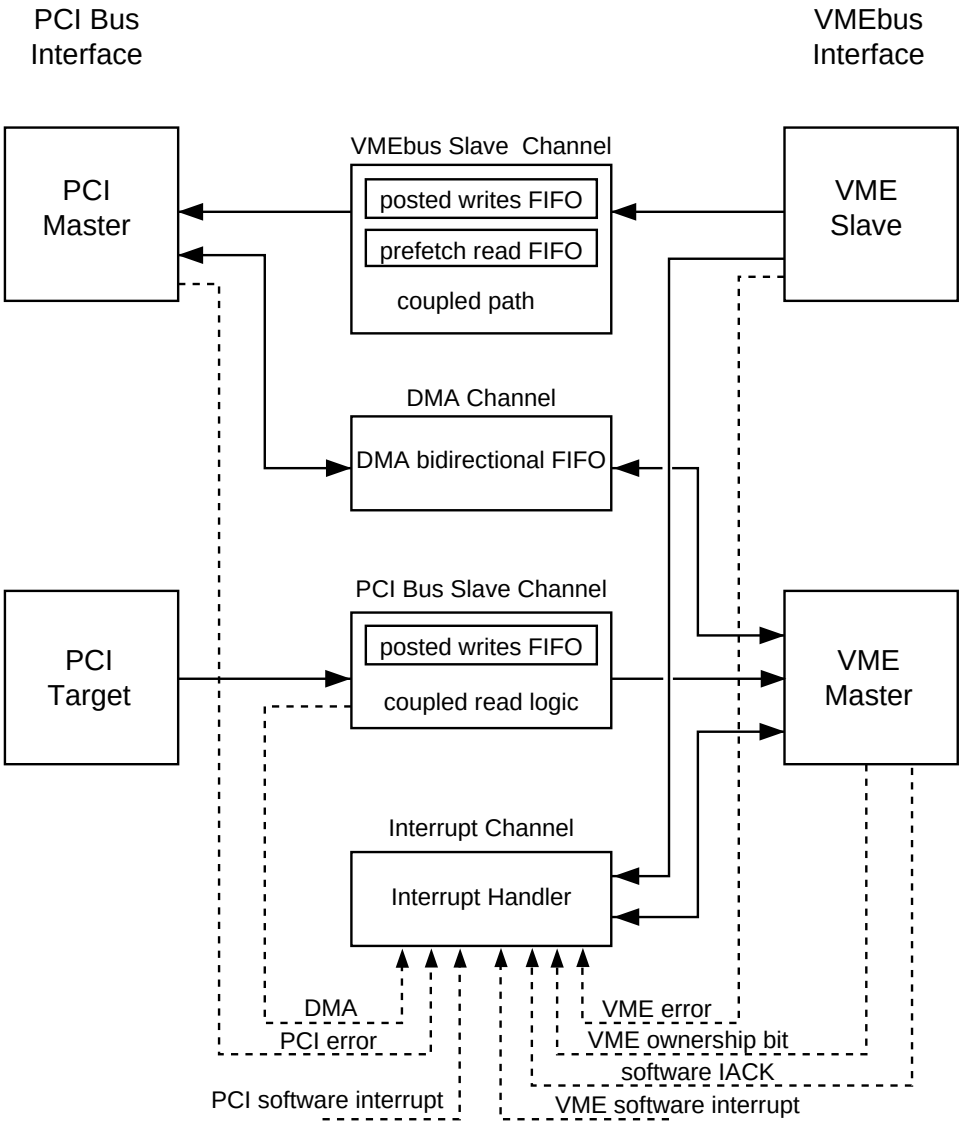
Some sources can be mapped to both buses, but mapping interrupts to a single bus is recommended.

Table 18: 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 20 shows the sources of interrupts, and the interfaces from which they originate.

Figure 20: Sources of Internal Interrupts



7.3.3.1 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 1 to the respective enable bits.

Interrupting the VMEbus Through Software

The following methods trigger software interrupts on the VMEbus:

1. The first method for interrupting the VMEbus through software involves writing 1 to one of the SW_INT7-1 bits in the VINT_EN register while the mask bit is 0.[1] This causes an interrupt to be generated on the corresponding IRQ7-1 line. For example, 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 1 to the SW_INT bit in the"VMEbus Interrupt Enable Register (VINT_EN)" on page 248 when this bit is 0 triggers one interrupt on the VMEbus on the level programmed in the "VME Interrupt Map 1 Register (VINT_MAP1)" on page 254.

   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.

This method is provided for compatibility with the original Universe device.

With both methods, the mask bit (SW_INTx or SW_INT) in the "VMEbus Interrupt Enable Register (VINT_EN)" on page 248 must be 0 in order for writing 1 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 "PCI Interrupt Enable Register (LINT_EN)" on page 242 (see "Software IACK Interrupt" on page 121).

Interrupting the PCI bus Through Software

On the PCI bus, there is only one method of directly triggering a software interrupt. This method is the same as the second method described in "Interrupting the VMEbus Through Software" on page 120. Causing a 0 to 1 transition in the SW_INT in the LINT_EN 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 "PCI Interrupt Map 1 Register (LINT_MAP1)" on page 247 determines which interrupt line is asserted on the PCI interface.

1. The term "enable" is more meaningful with respect to the other fields in this register, i.e., excluding the software interrupts. Writing to the software interrupt fields of this register does not enable an interrupt, it triggers an interrupt.

Termination of Software Interrupts

Any software interrupt can be cleared by clearing the respective bit in the “VMEbus Interrupt Enable Register (VINT_EN)” on page 248 or “PCI Interrupt Enable Register (LINT_EN)” on page 242. However, this method is not recommend for VME bus software interrupts because it can result in a false interrupts on that bus. These false interrupts are caused because the Universe II does not respond to the interrupt handler’s IACK cycle, and the handler is 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 and 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 “PCI Interrupt Status Register (LINT_STAT)” on page 244 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.

7.3.3.2 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 is generated upon completion of an IACK cycle that had been initiated by the Universe II’s software interrupt if the following occurs:

- The SW_IACK bit is set in the “PCI Interrupt Status Register (LINT_STAT)” on page 244
- The SW_IACK field in the “PCI Interrupt Map 1 Register (LINT_MAP1)” on page 247 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 device and acknowledged.

Like other interrupt sources, this interrupt source can be independently enabled through the “PCI Interrupt Enable Register (LINT_EN)” on page 242 and mapped to a particular LINT_ pin using the LINT_MAP1 register. A status bit in the LINT_STAT register indicates when the interrupt source is active, and is used to clear the interrupt once it has been serviced.

7.3.3.3 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 (“Master Control Register (MAST_CTL)” on page 271). 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 63). The interrupt is cleared by writing a one to the matching bit in the “PCI Interrupt Status Register (LINT_STAT)” on page 244.

7.3.3.4 DMA Interrupt

The DMA module provides the following possible interrupt sources:

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

All of these interrupt sources are ORed to a single DMA interrupt output line. When an interrupt comes from the DMA module, software must read the DMA status bits in the “PCI Interrupt Status Register (LINT_STAT)” on page 244 and the “VMEbus Interrupt Status Register (VINT_STAT)” on page 251 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 104).

7.3.3.5 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 written to (see “Mailbox Registers” on page 83). The user may enable or disable an interrupt response to the access of any mailbox register. Each register access may be individually mapped to a specific interrupt on the PCI bus through the “PCI Interrupt Map 2 Register (LINT_MAP2)” on page 263 and/or the VMEbus through the “VME Interrupt Map 2 Register (VINT_MAP2)” on page 264. The status of the PCI interrupt and the VMEbus are recorded in the “PCI Interrupt Status Register (LINT_STAT)” on page 244 and “VMEbus Interrupt Status Register (VINT_STAT)” on page 251.

7.3.3.6 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 38).

In order for an incoming VMEbus transaction to activate the location monitor of the Universe II the following criteria must be met:

- Location monitor must be enabled

- Access must be within 4 kbytes of the location monitor base address (see “Location Monitor Base Address Register (LM_BS)” on page 302)
- 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, and
- VA[4:3] = 11 selects Location Monitor 4.

An interrupt response to the access of any location monitor can be enabled or disabled with bits in the “PCI Interrupt Enable Register (LINT_EN)” on page 242. Access to each location monitor can be individually mapped to a specific interrupt on the PCI bus through the “PCI Interrupt Map 2 Register (LINT_MAP2)” on page 263—not to the VMEbus bus. The status of the PCI interrupt is logged in the LM bit of the “PCI Interrupt Status Register (LINT_STAT)” on page 244.

7.3.3.7 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.

7.3.4 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 42.

8. Error Handling

Errors occur in a system as a result of parity, bus, or internal problems. In order to handle errors so that they have minimum effects on an application, devices have a logic module called an error handler. The error handler logs data about the error then communicates the information to another device (for example, a host processor) that is capable of resolving the error condition.

This chapter discusses the following topics:

8.1 Overview

There are different conditions under which bus errors can 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 can 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.

8.2 Errors on 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 Configuration Space Control and Status Register (PCI_CSR)" on page 172 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 in the PCI_CSR register. No information is logged in either direction, nor is an interrupt generated.

8.3 Errors on Decoupled Transactions

During decoupled transactions, there is a possibility that an error in the transaction can occur. The following sections detail the decoupled transactions supported by Universe II and the types of error handling supported for these transactions.

8.3.1 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 or target by the Universe II, the Universe II can 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.

8.3.1.1 Error Logs

If the error occurs during a posted write to the VMEbus, the Universe II uses the “VMEbus AM Code Error Log (V_AMERR)” on page 307 to log the AM code of the transaction in the AMERR field. 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 12.3.108). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR interrupts are enabled (see “Interrupt Generation and Handling” on page 109). The remaining entries of the transaction are removed from the FIFO.

If the error occurs during a posted write to the PCI bus, the Universe II uses the “PCI Command Error Log Register (L_CMDERR)” on page 210 to log the command information for the transaction (CMDERR [3:0]). The address of the errored transaction is latched in the “PCI Address Error Log (LAERR)” on page 211. An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “Interrupt Generation and Handling” on page 109).

Under either of the 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. if 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).

8.3.2 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 “VME Slave Image Programming” on page 67). 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 “Master Control Register (MAST_CTL)” on page 271). 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.

8.3.3 DMA Errors

The Universe II’s response to a bus error during a transfer controlled by the DMA Channel is described in “DMA Error Handling” on page 105.

8.3.4 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 Configuration Space Control and Status Register (PCI_CSR)” on page 172 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 (that is, 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_CSR 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_CSR 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 for one clock cycle and setting the S_SERR (Signalled SERR_) bit in the PCI_CSR register. Assertion of SERR_ can be disabled by clearing the SERR_EN bit in the PCI_CSR 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 monitors SERR_ and takes appropriate action.

9. Resets, Clocks and Power-up Options

This chapter highlights utility functions in the Universe II. This chapter discusses the following topics:

9.1 Overview

The Universe II has many programmable reset options and power-up options that impact the functionality of the device.

9.2 Resets

The Universe II provides a number of pins and registers for reset support. Pin support is summarized in Table 19.

Table 19: Hardware Reset Mechanisms

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 re-configures 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 re-configures 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 133

The Universe II is only reset through hardware. Software can 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, does not reset the Universe II itself unless LRST_ is looped back to RST_. As described in “Reset Implementation Cautions” on page 133, there are potential loopback configurations resulting in permanent reset.

Table 20: Software Reset Mechanism

Register	Name	Type	Function
“Master Control Register (MAST_CTL)” on page 271	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
“VMEbus CSR Bit Set Register (VCSR_SET)” on page 330	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

Table 20: Software Reset Mechanism

Register	Name	Type	Function
“VMEbus CSR Bit Clear Register (VCSR_CLR)” on page 329	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

9.2.1 Universe II Reset Circuitry

Table 21 on page 132 and Figure 21 on page 133 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_ must be asserted.

PWRRST_ resets all aspects of Universe II listed in column 1 of Table 21. Table 21 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 Specification* (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_.

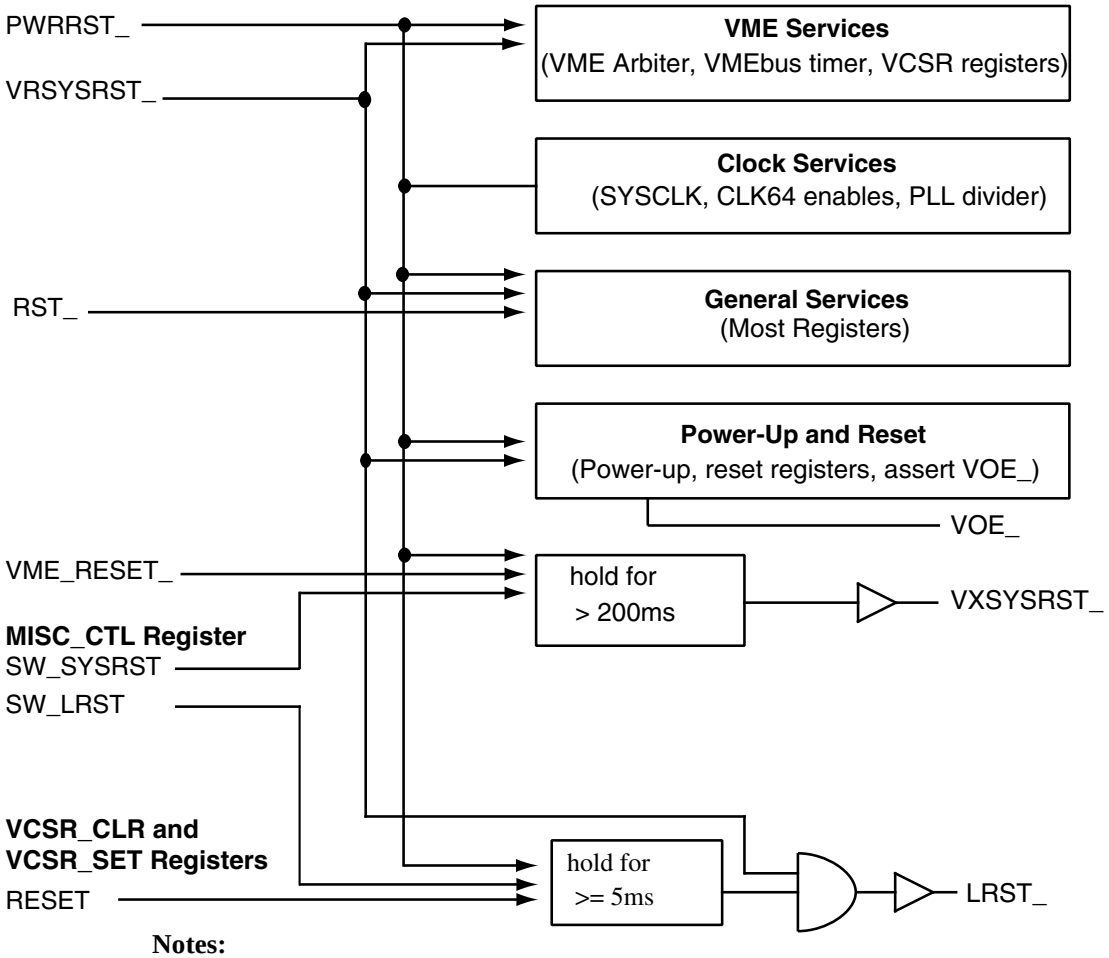
Table 21: Functions Affected by Reset Initiators

Effect of Reset[a]	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	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 VRSYSRST_, or SW_LRST bit in MISC_CTL register, or RESET bit in VCSR_SET register[b]

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. LRST_ can be cleared by writing 1 to the RESET bit in the CSR_CLR register.

Figure 21: Reset Circuitry



Notes:

1. On PWRRST_, options are loaded from pins. On SYSRST and RST_, options are loaded from values that were latched at the previous PWRRST_.
2. Refer to “Registers” on page 163 to find the effects of various reset events.

9.2.2 Reset Implementation Cautions

To prevent the Universe II from resetting the PCI bus, the LRST_ output can be left unconnected. Otherwise, LRST_ must be grouped with other PCI reset generators to assert the RST_ signal so that the following conditions are met:

RST_ = LRST_ & reset_source1 & reset_source2 &...

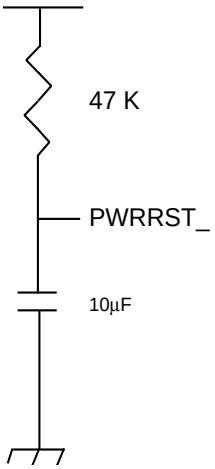
If the Universe II is the only initiator of PCI reset, LRST_ can be directly connected to RST_.

Assertion of VME_RESET_ causes the Universe II to assert VXSYSRST_.

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 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 21). It must be held asserted for over 100 milliseconds after power is stable. Typically this can be achieved through a resistor/capacitor combination (see Figure 22)or under voltage sensing circuits.

Figure 22: Resistor-Capacitor Circuit Ensuring Power-Up Reset Duration



The Universe II supports the VMEbus CSR Bit Clear and Bit Set registers. The “VMEbus CSR Bit Set Register (VCSR_SET)” on page 330 register allows the user to assert LRST_ or SYSFAIL by writing to the RESET or SYSFAIL bits. LRST_ or SYSFAIL remains asserted until the corresponding bit is cleared in the“VMEbus CSR Bit Clear Register (VCSR_CLR)” on page 329. The FAIL bit in each of these registers is a status bit and is set by the software to indicate board failure.

9.3 Power-Up Options

The Universe II can 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.

Table 22: 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	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	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 "Registers" on page 163	VA[1]
PCI Bus Size[b]	MISC_STAT	LCLSIZE	32-bit	REQ64_
PCI CSR Master Enable	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 PCI Bus Size is loaded on any RST_ event (*PCI 2.1 Specification*).

The majority of the Universe II power-up options are loaded from the VMEbus address and data lines after any PWRRST_ (see Table 22 on page 135). The PCI bus width (a power-up option required by the *PCI 2.1 Specification*) 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 Ω is required on the signal. See “PCI Bus Width” on page 138 and the *VMEbus Specification.*

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_ are reloaded into the Universe II (except for PCI bus width and VMEbus SYSCON enabling, which are loaded from their pins).

Table 22 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 136.

9.3.1 Power-up Option Descriptions

This section describes each of the groups of power-up options that were listed in Table 22.

9.3.1.1 VMEbus Register Access Image

The Universe II has several VMEbus slave images, each of which can provide a different mapping of VMEbus cycles to PCI cycles. All VMEbus slave images are configurable through a set of VMEbus slave image registers: VSIx_CTL, VSIx_BS, VSIx_BD, and VSIx_TO.

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 access image address space and the upper five bits of its base address; all other bits are 0 (see Table 23). 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 23 shows how the upper bits in the VRAI base address are programmed for A16, A24, and A32 VMEbus register access images.

Table 23: 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

9.3.1.2 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 “Registers” on page 163).

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 128 Mbyte 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 or configured through a register access before accessing configuration ROM.

9.3.1.3 Auto-ID

There are two Auto-ID mechanisms provided by the Universe II. One is the *VME64 Specification* version which relies upon use of the CR/CSR space for configuration of the VMEbus system, and a IDT 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 42).

Because VME64 Auto-ID relies upon SYSFAIL to operate correctly, this power-up option overrides the SYSFAIL power-up option described in “SYSFAIL* Assertion” on page 138.

9.3.1.4 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 can be enabled as a power-up option (see “Bus Isolation Mode (BI-Mode)” on page 46). When the Universe II has been powered-up in BI-Mode, then any subsequent SYSRST* or RST_ restores the Universe II to BI-Mode,

9.3.1.5 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.

9.3.1.6 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 is used when extensive on-board diagnostics need to be performed before release of SYSFAIL*. After completion of diagnostics, SYSFAIL* can be released through software or through initiation of the VME64 Auto-ID sequence (if enabled, see “Auto Slot ID: VME64 Specified” on page 42).

9.3.1.7 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 possesses its other default conditions: the translation offset is 0, posted writes are disabled, and only 32-bit (maximum) non-block VMEbus cycles in the non-privileged data space are generated.

This option is typically used to access permits the use of Boot ROM on another card in the VMEbus system.

9.3.1.8 PCI Register Access

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

9.3.1.9 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 the *PCI 2.1 Specification*). 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_ can be left un-terminated.

9.3.1.10 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 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 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 is constant until the next power-up sequence. See "Memory or I/O Access" on page 77.

9.3.2 Power-up Option Implementation

In order to implement power-up requirements for the Universe II weak pull-up resistors are required.

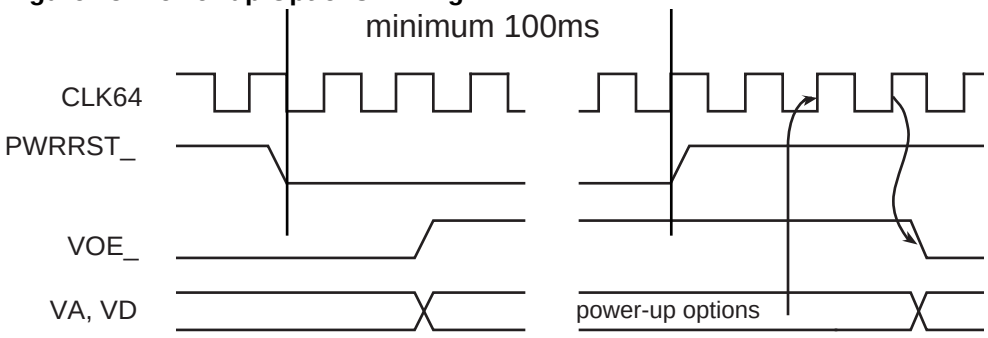
9.3.2.1 Pull-up Requirements

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 negates VOE_, putting these signals into a high-impedance state. While VOE_ is negated the pull-ups (or internal pull-downs) bring the option pins (on A[31:1] and D[31:27]) to their appropriate state.

The internal pull-downs are very weak. The leakage current on many transceivers can be sufficient to override these pull-downs. To ensure proper operation designers must ensure power-up option pins 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 23: 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.

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 is not prepared to receive a slave access until 70 ns after release of SYSRST*.

9.3.3 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 24 shows how these pins must be terminated.

Table 24: Manufacturing Pin Requirements for Normal Operating Mode

Pin Name	Pin Value
tmode[2]	V_{SS} (or pulled-down if board tests are performed, see “Auxiliary Test Modes” on page 141)
tmode[1]	
tmode[0]	
pll_testsel	V_{SS}
enid	V_{SS}
pll_testout	No connect
VCOCTL	V_{SS}

9.4 Test Modes

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

9.4.1 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 25 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 25: Test Mode Operation

Operation Mode	TMODE[2:0]	PLL_TESTSEL
Normal Mode	000	0
Accelerate	001	0
PLL Test	010	1
Scan Mode	011	1
NAND Tree Simulation	100	1
RAM Test	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.

9.4.2 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:

- five-pin JTAG interface (TCK, TDI, TDO, TMS, and TRST_)
- JTAG TAP controller
- three-bit instruction register

- boundary scan register
- bypass register
- 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.

9.5 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 50-50 duty cycle with a maximum rise time of 5 ns.

Using a different clock frequency is not recommended. It will alter various internal timers and change VME timing.

10. Signals and Pinout

This chapter discusses the following topics:

10.1 Overview

The following detailed description of the Universe II signals is organized according to these functional groups:

- VMEbus Signals
- PCI Signals

10.2 VMEbus Signals

Table 26: 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.	
VAS_DIR	**Output**
VMEbus Address Strobe Direction Control – 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.	
VBCLR_	**Output**
VMEbus Bus Clear – 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.	
VBGI[3:0]_	**Input**
VMEbus Bus Grant Inputs – 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.	

Table 26: VMEbus Signals *(Continued)*

VBGO[3:0]_	**Output**
VMEbus Bus Grant Outputs – Only one output is asserted at any time, according to the level at which the VMEbus is being granted.	
VD[31:0]_	**Bidirectional**
VMEbus Data Lines – 31 through 0	
VD_DIR	**Output**
VMEbus Data Transceiver Direction Control – 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.	
VDS[1:0]_	**Bidirectional**
VMEbus Data Strobes – 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.	
VDS_DIR	**Output**
VMEbus Data Strobe Direction Control – 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.	
VDTACK_	**Bidirectional**
VMEbus Data Transfer Acknowledge – 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*.	
VIACK_	**Bidirectional**
VMEbus Interrupt Acknowledge – Indicates that the cycle just beginning is an interrupt acknowledge cycle.	
VIACKI_	**Input**
VMEbus Interrupt Acknowledge In – 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_.	
VIACKO_	**Output**
VMEbus Interrupt Acknowledge Out– Generated by the Universe II if it receives VIACKI_ and is not currently generating an interrupt at the level being acknowledged.	
VLWORD_	**Bidirectional**
VMEbus Longword Data Transfer Size Indicator – 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.	

Table 26: VMEbus Signals *(Continued)*

VOE_	**Output**
VMEbus Transceiver Output Enable – 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). VOE_ is negated during some VMEbus Slave Channel read operations.	
VRACFAIL_	**Input**
VMEbus ACFAIL Input signal – Warns the VMEbus system of imminent power failure. This gives the modules in the system time to shut down in an orderly fashion before power-down. ACFAIL is mapped to a PCI interrupt.	
VRBBSY_	**Input**
VMEbus Receive Bus Busy – Allows the Universe II to monitor whether the VMEbus is owned by another VMEbus master	
VRBERR_	**Input**
VMEbus Receive Bus Error – A low level signal indicates that the addressed slave has not responded, or is signalling an error.	
VRBR[3:0]_	**Input**
VMEbus Receive Bus Request Lines – 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.	
VRIRQ[7:1]_	**Input**
VMEbus Receive Interrupts 7 through 1 – These interrupts can be mapped to any of the Universe II's PCI interrupt outputs. VRIRQ7-1_ are individually maskable, but cannot be read.	
VRSYSFAIL_	**Input**
VMEbus Receive SYSFAIL – Asserted by a VMEbus system to indicate some system failure. VRSYSFAIL_ is mapped to a PCI interrupt.	
VRSYSRST_	**Input**
VMEbus Receive System Reset – Causes assertion of LRST_ on the local bus and resets the Universe II.	
VSLAVE_DIR	**Output**
VMEbus Slave Direction Control – Transceiver control that allows 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.	
VSYSCLK	**Bidirectional**
VMEbus System Clock – Generated by the Universe II when it is the Syscon and monitored during DY4 Auto ID sequence	

Table 26: VMEbus Signals *(Continued)*

VSCON_DIR	**Output**
Syscon Direction Control – 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.	
VWRITE_	**Bidirectional**
VMEbus Write signal – Indicates the direction of data transfer.	
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*.	

10.3 PCI Bus Signals

Table 27: PCI Bus Signals

ACK64_	**Bidirectional**
Acknowledge 64-bit Transfer – It indicates slave can perform a 64-bit transfer when driven by the PCI slave (target).	
AD [31:0]	**Bidirectional**
PCI Address/Data Bus – Address and data are multiplexed over these pins providing a 32-bit address and 32-bit data bus.	

Table 27: PCI Bus Signals

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 – This signal 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 – This signal 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 – This signal is used as a chip select during configuration read and write transactions	
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 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. When the Universe II samples LOCK_, it can be excluded from a particular PCI device.	
LRST_	**Output**

Table 27: PCI Bus Signals

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_	**Output**
System Error – Reports address parity errors or any other system error.	
STOP_	**Bidirectional**
Stop – 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 terminates the transaction if it receives STOP_ from the PCI slave.	

Table 27: PCI Bus Signals

TCK	**Input**
JTAG Test Clock Input – Used to clock the Universe II's TAP controller. Tie to any logic level if JTAG is not used in the system.	
TDI	**Input**
JTAG Test Data Input – 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.	
TDO	**Output**
JTAG Test Data Output – Used to serially shift test data and test instructions out of the Universe II	
TMODE [2:0]	**Input**
Test Mode Enable – Used for chip testing, tie to ground for normal operation.	
TMS	**Input**
JTAG Test Mode Select – 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.	
TRDY_	**Bidirectional**
Target Ready – 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.	
TRST_	**Input**
JTAG Test Reset – Provides asynchronous initialization of the TAP controller in the Universe II. Tie to ground if JTAG is not used in the system.	
VCOCTL	**Input**
Manufacturing testing – Tie to ground for normal operation	
VME_RESET_	**Input**
VMEbus Reset Input — Generates a VME bus system reset.	

10.4 Pin-out

10.4.1 313-pin Plastic BGA Package (PBGA)

	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**	1
2		vd[18]		vd[14]		vd[13]		vd[6]		vd[2]		vscon_ DIR		vxbbsy		ad[61]		ad[59]		ad[24]		PLL_ testout		VDD		2
3	vd[23]		vd[21]		vd[20]		vd[12]		vd[3]		int_[5]		vbclr_		vrbr_[2]		VDD		ad[57]		lclk		VDD		vcoctl	3
4		vd[26]		VDD		vd[15]		vd[11]		vd[4]		int_[4]		VSS		pwrrst_		ad[25]		ad[56]		perr_		ad[22]		4
5	vd[30]		vd[24]		VDD		vd[17]		vd[8]		VDD		ad[63]		ad[60]		ad[26]		vrbr_[0]		int_[1]		**AVDD**		par64	5
6		vd[27]		vd[25]		vd[16]		vd[10]		vd[7]		vrbbsy_		ad[30]		VSS		int_[3]		ad[23]		ad[55]		VSS		6
7	vrberr_		vd[28]		viack_		**VDD**		vd[0]		vrbr_[3]		vsysclk		VDD		ad[28]		**VDD**		serr_		devsel_		ad[20]	7
8		vam_DIR		vwrite_		vd[29]		**VDD**		vd[1]		int_[6]		par		ad[29]		**VDD**		ad[21]		ad[54]		trdy_		8
9	vam[5]		VDD		vd[31]		vam[2]		**VDD**		int_[2]		**VSS**		ad[62]		**VDD**		ad[18]		ad[53]		VSS		VDD	9
10		vam[3]		vam[1]		vd_DIR		vam[4]		**VSS**		vxsysfail		ad[31]		**VDD**		ad[51]		VSS		ad[52]		ad[19]		10
11	vds_[1]		tms		vam[0]		VDD		vds_DIR		**VSS**		**VSS**		**VSS**		VSS		ack64_		VDD		ad[50]		ad[16]	11
12		voe_		vxberr		vds_[0]		tck		vas_DIR		**VSS**		**VSS**		cbe[6]		ad[48]		ad[49]		ad[17]		cbe[7]		12
13	tdi		tdo		trst_		va_DIR		**VSS**		**VSS**		**VSS**		**VSS**		**VSS**		cbe[3]		tmode[0]		vrsysfail_		VSS	13
14		vas_		va[5]		va[3]		va[1]		vlword_		**VSS**		**VSS**		cbe[2]		cbe[1]		cbe[0]		VSS		cbe[5]		14
15	va[2]		vslave_DIR		VDD		vdtack_		va[4]		**VSS**		**VSS**		**VSS**		ad[15]		VDD		ad[14]		irdy_		cbe[4]	15
16		va[8]		va[10]		va[13]		va[7]		**VSS**		vbgi_[1]		ad[0]		**VSS**		vrirq_[5]		VSS		idsel		ad[47]		16
17	VDD		va[9]		va[14]		va[6]		**VDD**		ad[32]		**VSS**		vrirq_[7]		**VDD**		frame_		ad[13]		VDD		gnt_	17
18		va[12]		va[17]		va[16]		**VDD**		vxirq[3]		vbgo_[2]		vracfail_		ad[3]		**VDD**		ad[45]		stop_		req64_		18
19	va[11]		va[18]		va[23]		**VDD**		vxirq[1]		VDD		ad[33]		ad[2]		ad[35]		**VDD**		rst_		ad[12]		ad[46]	19
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]		20
21	va[15]		vrsysrst_		va[28]		va[27]		vrirq_[4]		viacko_		vbgi_[0]		VDD		ad[38]		ad[41]		tmode[1]		ad[10]		enid	21
22		va[20]		VDD		va[29]		vrirq_[1]		req_		vxirq[6]		ad[1]		ad[4]		vme_reset_		ad[40]		ad[8]		ad[42]		22
23	vxsysrst		clk64		va[25]		vxbr[3]		VDD		vxirq[5]		vbgi_[3]		ad[34]		ad[36]		tmode[2]		lock_		VDD		viacki_	23
24		VDD		va[30]		va[31]		vxbr[1]		vxirq[2]		vbgo_[3]		vxbr[2]		VSS		VSS		VDD		ad[7]		VDD		24
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]	25

11. Electrical Characteristics

This chapter discusses the following topics:

11.1 DC Characteristics

11.1.1 Non-PCI Characteristics

Table 28 specifies the required DC characteristics of all non-PCI signals pins.

Table 28: Non-PCI Electrical Characteristics

Symbols	Parameters	Test conditions	Tested at 0°C to 70°C	
			Min	Max
VIH_TTL	Voltage Input high	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	2.2 V	V_{DD} + 0.3V
VIH_CMOS	Voltage Input high	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	0.7 Vdd	V_{DD} + 0.3V
VIL_TTL	Voltage Input low	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	–0.3 V	0.8V
VIL_CMOS	Voltage Input low	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	–0.3 V	0.3V_{DD}
VT+_TTL	Voltage Input high (Schmitt trigger)	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	-	2.4 V
VT+_CMOS	Voltage Input high (Schmitt trigger)	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	-	0.7V_{DD}
VT -_ttl	Voltage Input low (Schmitt trigger)	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	0.8V	-
VT -_CMOS	Voltage Input low (Schmitt trigger)	V_{OUT} = 0.1V or V_{DD} – 0.1V; [I_{OUT}] = 20 µA	0.25V_{DD}	-
I_{IN}	Input leakage current	With no pull-up or pull-down resistance (Vin = Vss or V_{DD})	-5.0µA	5.0µA

Table 28: Non-PCI Electrical Characteristics

Symbols	Parameters	Test conditions	Tested at 0°C to 70°C	
			Min	Max
I_{IH}	Input leakage current high	Inputs with pull-down resistance (Vin = Vdd)	10µA	180µA
I_{IL}	Input leakage current low	Inputs with pull-up resistance (Vin = Vss)	-180µA	-10µA
I_{OZ}	Tristate output leakage	Vout = Vdd or Vss	-10.0µA	10.0µA

11.1.2 PCI Characteristics

Table 29 specifies the required AC and DC characteristics of all PCI Universe II signal pins.

Table 29: AC/DC PCI Electrical Characteristics

Symbols	Parameters	Condition	Min	Max	Units
VIL	Voltage Input low		-0.5	0.8	V
VIH	Voltage Input high		2.0	Vdd + 0.5	V
I_{IN}	Input leakage current	Vin = 2.7V or 0.5V	-10	10	µA
I_{IL}	Input leakage current low (Pin with pull-up)	Vin = 0.5V	-70	-10	µA
VOL	Voltage output low	Iout = 3mA, 6mA	-	0.4	V
VOH	Voltage output high	Iout = -2mA	2.4	-	V
IOH (AC)[a]	Switching current high	0 < Vout[2]1.4	-44	-	mA
		1.4 < Vout < 2.4	-44 + (Vout -1.4) / 0.024	-	mA
		3.1 < Vout < Vdd	-	Equation A	mA
	(Test point)	Vout = 3.1V	-	-142	mA
IOL (AC)[b]	Switching current high	0 < Vout[3]1.4	95	-	mA
		0.6Vdd < Vout < 0.1Vdd	Vout / 0.023	-	mA
		0.71 < Vout < 0	-	Equation B	mA
	(Test point)	Vout = 0.71V	-	206	mA

Table 29: AC/DC PCI Electrical Characteristics

Symbols	Parameters	Condition	Min	Max	Units
ICL	Low clamp current	-5 Vin^2-1	-25 + (Vin +10) / 0.015	-	mA
SLEWR	Output rise slew rate	0.4V to 2.4V load	1	5	V/ns
SLEWR	Output fall slew rate	2.4V to 0.4V load	1	5	V/ns

a. Equation A: Ioh = 11.9 * (Vout - 5.25) * (Vout + 2.45) for Vdd > Vout > 3.1V

b. Equation B: Iol = 78.5 * Vout * (4.4 - Vout) for 0V < Vout < 0.71V

11.1.3 Pin List and DC Characteristics for all Signals

Table 30 specifies the required DC characteristics of all Universe II signal pins

Table 30: Pin List and DC Characteristics for Universe II Signals

Pin Name	Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description
ack64#	W11	I/O	TTL	3S	6	–2	PCI Acknowledge 64 Bit Transfer
AD [63:0]	Listed in: "313 Pin PBGA Package" on page 333	I/O	TTL	3S	6	–2	PCI Address/Data Pins
C/BE# [0]	Y14	I/O	TTL	3S	6	–2	PCI Command and Byte Enables
C/BE# [1]	V14						
C/BE# [2]	T14						
C/BE# [3]	W13						
C/BE# [4]	AE15						
C/BE# [5]	AD14						
C/BE# [6]	T12						
C/BE# [7]	AD12						
clk64	C23	I	TTL	–	–	–	VME Clock 64 MHz—60-40 duty, 5 ns rise time
devsel#	AC7	I/O	–	3S	6	–2	PCI Device Select
enid	AE21	I	CMOS	–	–	–	Enable IDD Tests

Table 30: Pin List and DC Characteristics for Universe II Signals *(Continued)*

Pin Name	Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description
frame#	W17	I/O	TTL	3S	6	–2	PCI Cycle Frame
gnt#	AE17	I	TTL	–	–	–	PCI Grant
idsel	AB16	I	TTL	–	–	–	PCI Initialization Device Select
lint# [0]	K20	I/O	TTL	OD	12	–12	PCI Interrupt
lint# [1]	AA5	I/O	TTL	OD	4	–4	
lint# [2]	L9						
lint# [3]	V6						
lint# [4]	M4						
lint# [5]	L3						
lint# [6]	M8						
lint# [7]	L1						
irdy#	AC15	I/O	TTL	3S	6	–2	PCI Initiator Ready
lclk	AA3	I	TTL	–	–	–	PCI Clock Signal
lock#	AA23	I/O	TTL	3S	6	–2	PCI Lock
lrst#	R1	O	–	3S	6	–2	PCI Reset Output
par	P8	I/O	TTL	3S	6	–2	PCI parity
par64	AE5	I/O	TTL	3S	6	–2	PCI Parity Upper DWORD
perr#	AB4	I/O	TTL	3S	6	–2	PCI Parity Error
pll_testout	AB2	For factory testing					
pll_testsel	AC1	For factory testing					
pwrrst#	T4	I	TTL/Schm	–	–	–	Power–up Reset
req#	K22	O	–	3S	6	–2	PCI Request
req64#	AD18	I/O	TTL	3S	6	–2	PCI Request 64 Bit Transfer
rst#	AA19	I	TTL	–	–	–	PCI Reset
serr#	AA7	O	TTL	OD	12	–12	PCI System Error
stop#	AB18	I/O	TTL	3S	6	–2	PCI Stop

Table 30: Pin List and DC Characteristics for Universe II Signals *(Continued)*

Pin Name	Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description
tck	H12	I	TTL	–	–	–	JTAG Test Clock Input
tdi	A13	I	TTL (PU)	–	–	–	JTAG Test Data Input
tdo	C13	O	–	3S	–	–	JTAG Test Data OUTput
tmode [0]	AA13	I	TTL	–	–	–	Test Mode Enable
tmode [1]	AA21						
tmode [2]	W23						
tms	C11	I	TTL (PU)	–	–	–	JTAG Test Mode Select
trdy#	AD8	I/O	TTL	3S	6	–2	PCI Target Ready
trst#	E13	I	TTL (PU)	–	–	–	JTAG Test Reset
VA [31:1]	Listed in: “313 Pin PBGA Package” on page 333	I/O	TTL (PD)	3S	3	–3	VMEbus Address Pins
vam [0]	E11	I/O	TTL	3S	3	–3	VMEbus Address Modifier Signals
vam [1]	D10						
vam [2]	G9						
vam [3]	B10						
vam [4]	H10						
vam [5]	A9						
vam_dir	B8	O	–	3S	6	–6	VMEbus AM Signal Direction Control
vas#	B14	I/O	TTL/Schm (PU)	3S	3	–3	VMEbus Address Strobe
vas_dir	K12	O	–	3S	6	–6	VMEbus AS Direction Control
va_dir	G13	O	–	3S	12	–12	VMEbus Address Direction Control
vbclr#	N3	O	–	3S	3	–3	VMEbus BCLR* Signal

Table 30: Pin List and DC Characteristics for Universe II Signals *(Continued)*

Pin Name	Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description
vbgi# [0]	N21	I	TT	–	–	–	VMEbus Bus Grant In
vbgi# [1]	M16						
vbgi# [2]	N25						
vbgi# [3]	N23		TT (PD)				
vbgo# [0]	M20	O	–	3S	12	–12	VMEbus Bus Grant Out
vbgo# [1]	L25						
vbgo# [2]	M18						
vbgo# [3]	M24						
vcoctl	AE3	I	–	–	–	–	Factory testing
VD [31:0]	Listed in: “313 Pin PBGA Package” on page 333	I/O	TTL	3S	3	–3	VMEbus Data Pins
vd_dir	F10	O	–	3S	12	–12	VMEbus Data Direction Control
vds# [0]	F12	I/O	TTL (PU)	3S	3	–3	VMEbus Data Strobes
vds# [1]	A11						
vds_dir	J11	O	–	3S	6	–6	VMEbus Data Strobe Direction Control
vdtack#	G15	I/O	TTL/Schm (PU)	3S	3	–3	VMEbus DTACK* Signal
viack#	E7	I/O	TTL	3S	3	–3	VMEbus IACK* Signal
viacki#	AE23	I	TTL	–	–	–	VMEbus IACKIN* Signal
viacko#	L21	O	–	3S	12	–12	VMEbus IACKOUT* Signal
vlword#	K14	I/O	TTL (PD)	3S	3	–3	VMEbus LWORD* Signal
VME_RESET#	V22	I	TTL				VMEbus Reset Input
voe#	B12	O	–	3S	24	–24	VMEbus Transceiver Output Enable
vracfail#	P18	I	TTL/Schm	–	–	–	VMEbus ACFAIL* Signal

Table 30: Pin List and DC Characteristics for Universe II Signals *(Continued)*

Pin Name	Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description
vrbbsy#	M6	I	TTL/Schm	–	–	–	VMEbus Received BBSY* Signal
vrberr#	A7	I	TTL/Schm	–	–	–	VMEbus Receive Bus Error
vrbr# [0]	W5	I	TTL/Schm	–	–	–	VMEbus Receive Bus Request
vrbr# [1]	U1						
vrbr# [2]	R3						
vrbr# [3]	L7						
vrirq# [1]	H22	I	TTL/Schm	–	–	–	VMEbus Receive Interrupts
vrirq# [2]	H20						
vrirq# [3]	E25						
vrirq# [4]	J21						
vrirq# [5]	V16						
vrirq# [6]	P20						
vrirq# [7]	R17						
vrsysfail#	AC13	I	TTL	–	–	–	VMEbus Receive SYSFAIL Signal
vrsysrst#	C21	I	TTL/Schm	–	–	–	VMEbus Receive SYSRESET* Signal
vscon_dir	M2	O	–	3S	6	–6	SYSCON signals direction control
vslave_dir	C15	O	–	3S	6	–6	DTACK/BERR direction control
vsysclk	N7	I/O	TTL	3S	3	–3	VMEbus SYSCLK Signal
vwrite#	D8	I/O	TTL	3S	3	–3	VMEbus Write
vxbbsy	P2	O	–	3S	3	–3	VMEbus Transmit BBSY* Signal
vxberr	D12	O	–	3S	3	–3	VMEbus Transmit Bus Error (BERR*)

Table 30: Pin List and DC Characteristics for Universe II Signals *(Continued)*

Pin Name	Pin Number	Type	Input Type	Output Type	I_{OL} (mA)	I_{OH} (mA)	Signal Description
vxbr [0]	G25	O	–	3S	3	–3	VMEbus Transmit Bus Request
vxbr [1]	H24						
vxbr [2]	P24						
vxbr [3]	G23						
vxirq [1]	J19	O	–	3S	3	–3	VMEbus Transmit Interrupts
vxirq [2]	K24						
vxirq [3]	K18						
vxirq [4]	J25						
vxirq [5]	L23						
vxirq [6]	M22						
vxirq [7]	R25						
vxsysfail	M10	O	–	3S	3	–3	VMEbus Transmit SYSFAIL Signal
vxsysrst	A23	O	–	3S	3	–3	VMEbus Transmit SYSRESET* Signal

11.2 Operating Conditions

The following table specifies recommended operating condition for the Universe II.

Table 31: Operating Conditions

Symbols	Parameters	Min	Max	Frequency Operation (Mhz)
Vdd	DC Supply Voltage (5V ± 10%)	4.5V	5.5V	
Ta (Commercial)	Ambient Temperature	0ºC	+70ºC	25 - 33
Ta (Industrial)	Ambient Temperature	-40ºC	+85ºC	25 - 33
Ta (Extended)	Ambient Temperature	-55ºC	+125ºC	25

11.2.1 Absolute Maximum Ratings

Table 32: Absolute Maximum Ratings

Parameter	Range
DC Supply Voltage (VSS to VDD)	-0.3 to 7.0 V
DC Input Voltage (V_{IN})	-0.5 to vdd + 0.5V
DC Current Drain per Pin, Any Single Input or Output	± 50ma
DC Current Drain per Pin, Any Paralleled Outputs	± 100ma
DC Current Drain V_{DD} and V_{SS} Pins	± 75ma
Storage Temperature, (T_{STG})	-40 ºC to 125 ºC

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.

11.3 Power Dissipation

Table 33: Power Dissipation

Parameter	Rating
IDLE	
Power Dissipation	1.50 W
Typical	
Power Dissipation (32-bit PCI)	2.00 W
Power Dissipation (64-bit PCI)	2.20 W
Maximum	
Power Dissipation (32-bit PCI)	2.70W
Power Dissipation (64-bit PCI)	3.20W

11.4 Power Sequencing

When designing with the Universe II device, care must be taken when powering the device to ensure proper operation. During power-up, no signals must be applied to any Universe II signal pins prior to stable power being applied to the device.

In a mixed 3.3V and 5V design, IDT recommends that 5V power be stable prior to other devices coming out reset. If other devices come out of reset before the 5V power is stable, make certain that no signals are driven to the Universe II signal pins - including possible signals from the VME backplane.

12. Registers

This appendix discusses the following topics:

12.1 Overview

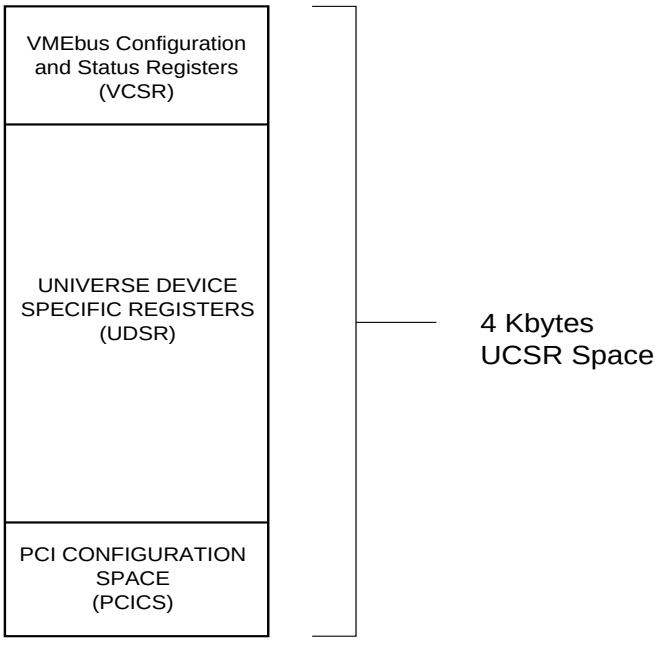
The Universe II Control and Status Registers facilitate host system configuration and allow the user to control Universe II operational characteristics. The registers are divided into three groups:

- PCI Configuration Space
- VMEbus Configuration and Status Registers
- Universe II Device Specific Status Registers

Universe II registers have little-endian byte-ordering.

Figure 24 below summarizes the supported register access mechanisms.

Figure 24: UCSR Access Mechanisms



Bits listed as reserved must be programmed with a value of 0. Reserved bits always read a value of zero.

12.2 Register Map

Table 34 lists the Universe II registers by address offset.

Table 34: Universe II Register Map

Offset	Register	Name
0x000	“PCI Configuration Space ID Register (PCI_ID)” on page 171	PCI_ID
0x004	“PCI Configuration Space Control and Status Register (PCI_CSR)” on page 172	PCI_CSR
0x008	“PCI Configuration Class Register (PCI_CLASS)” on page 176	PCI_CLASS
0x00C	“PCI Configuration Miscellaneous 0 Register (PCI_MISC0)” on page 177	PCI_MISC0
0x010	“PCI Configuration Base Address Register (PCI_BS0)” on page 178	PCI_BS0
0x014	“PCI Configuration Base Address 1 Register (PCI_BS1)” on page 179	PCI_BS1
0x018-0x024	PCI Unimplemented	
0x028	PCI Reserved	
0x02C	PCI Reserved	
0x030	PCI Unimplemented	
0x034	PCI Reserved	
0x038	PCI Reserved	
0x03C	“PCI Configuration Miscellaneous 1 Register (PCI_MISC1)” on page 180	PCI_MISC1
0x040-0x0FF	PCI Unimplemented	
0x100	“PCI Target Image 0 Control (LSI0_CTL)” on page 181	LSI0_CTL
0x104	“PCI Target Image 0 Base Address Register (LSI0_BS)” on page 183	LSI0_BS
0x108	“PCI Target Image 0 Bound Address Register (LSI0_BD)” on page 184	LSI0_BD
0x10C	“PCI Target Image 0 Translation Offset (LSI0_TO)” on page 185	LSI0_TO
0x110	Reserved	
0x114	“PCI Target Image 1 Control (LSI1_CTL)” on page 186	LSI1_CTL
0x118	“PCI Target Image 1 Base Address Register (LSI1_BS)” on page 188	LSI1_BS
0x11C	“PCI Target Image 1 Bound Address Register (LSI1_BD)” on page 189	LSI1_BD

Table 34: Universe II Register Map *(Continued)*

Offset	Register	Name
0x120	“PCI Target Image 1 Translation Offset (LSI1_TO)” on page 190	LSI1_TO
0x124	Reserved	
0x128	“PCI Target Image 2 Control (LSI2_CTL)” on page 191	LSI2_CTL
0x12C	“PCI Target Image 2 Base Address Register (LSI2_BS)” on page 193	LSI2_BS
0x130	“PCI Target Image 2 Bound Address Register (LSI2_BD)” on page 194	LSI2_BD
0x134	“PCI Target Image 2 Translation Offset (LSI2_TO)” on page 195	LSI2_TO
0x138	Reserved	
0x13C	“PCI Target Image 3 Control (LSI3_CTL)” on page 196	LSI3_CTL
0x140	“PCI Target Image 3 Base Address Register (LSI3_BS)” on page 198	LSI3_BS
0x144	“PCI Target Image 3 Bound Address Register (LSI3_BD)” on page 199	LSI3_BD
0x148	“PCI Target Image 3 Translation Offset (LSI3_TO)” on page 200	LSI3_TO
0x14C-0x16C	Reserved	
0x170	“Special Cycle Control Register (SCYC_CTL)” on page 201	SCYC_CTL
0x174	“Special Cycle PCI Bus Address Register (SCYC_ADDR)” on page 202	SCYC_ADDR
0x178	“Special Cycle Swap/Compare Enable Register (SCYC_EN)” on page 203	SCYC_EN
0x17C	“Special Cycle Compare Data Register (SCYC_CMP)” on page 204	SCYC_CMP
0x180	“Special Cycle Swap Data Register (SCYC_SWP)” on page 205	SCYC_SWP
0x184	“PCI Miscellaneous Register (LMISC)” on page 206	LMISC
0x188	“Special PCI Target Image (SLSI)” on page 208	SLSI
0x18C	“PCI Command Error Log Register (L_CMDERR)” on page 210	L_CMDERR
0x190	“PCI Address Error Log (LAERR)” on page 211	LAERR
0x194-0x19C	Reserved	
0x1A0	“PCI Target Image 4 Control Register (LSI4_CTL)” on page 212	LSI4_CTL
0x1A4	“PCI Target Image 4 Base Address Register (LSI4_BS)” on page 214	LSI4_BS
0x1A8	“PCI Target Image 4 Bound Address Register (LSI4_BD)” on page 215	LSI4_BD
0x1AC	“PCI Target Image 4 Translation Offset (LSI4_TO)” on page 216	LSI4_TO
0x1B0	Reserved	

Table 34: Universe II Register Map *(Continued)*

Offset	Register	Name
0x1B4	“PCI Target Image 5 Control Register (LSI5_CTL)” on page 217	LSI5_CTL
0x1B8	“PCI Target Image 5 Base Address Register (LSI5_BS)” on page 219	LSI5_BS
0x1BC	“PCI Target Image 5 Bound Address Register (LSI5_BD)” on page 220	LSI5_BD
0x1C0	“PCI Target Image 5 Translation Offset (LSI5_TO)” on page 221	LSI5_TO
0x1C4	Reserved	
0x1C8	“PCI Target Image 6 Control Register (LSI6_CTL)” on page 222	LSI6_CTL
0x1CC	“PCI Target Image 6 Base Address Register (LSI6_BS)” on page 224	LSI6_BS
0x1D0	“PCI Target Image 6 Bound Address Register (LSI6_BD)” on page 225	LSI6_BD
0x1D4	“PCI Target Image 6 Translation Offset (LSI6_TO)” on page 226	LSI6_TO
0x1D8	Reserved	
0x1DC	“PCI Target Image 7 Control Register (LSI7_CTL)” on page 227	LSI7_CTL
0x1E0	“PCI Target Image 7 Base Address Register (LSI7_BS)” on page 229	LSI7_BS
0x1E4	“PCI Target Image 7 Bound Address Register (LSI7_BD)” on page 230	LSI7_BD
0x1E8	“PCI Target Image 7 Translation Offset (LSI7_TO)” on page 231	LSI7_TO
0x1EC-0x1FC	Reserved	
0x200	“DMA Transfer Control Register (DCTL)” on page 232	DCTL
0x204	“DMA Transfer Byte Count Register (DTBC)” on page 234	DTBC
0x208	“DMA PCI Bus Address Register (DLA)” on page 235	DLA
0x20C	Reserved	
0x210	“DMA VMEbus Address Register (DVA)” on page 236	DVA
0x214	Reserved	
0x218	“DMA Command Packet Pointer (DCPP)” on page 237	DCPP
0x21C	Reserved	
0x220	“DMA General Control/Status Register (DGCS)” on page 238	DGCS
0x224	“DMA Linked List Update Enable Register (D_LLUE)” on page 241	D_LLUE
0x2280x-2FC	Reserved	
0x300	“PCI Interrupt Enable Register (LINT_EN)” on page 242	LINT_EN

Table 34: Universe II Register Map *(Continued)*

Offset	Register	Name
0x304	“PCI Interrupt Status Register (LINT_STAT)” on page 244	LINT_STAT
0x308	“PCI Interrupt Map 0 Register (LINT_MAP0)” on page 246	LINT_MAP0
0x30C	“PCI Interrupt Map 1 Register (LINT_MAP1)” on page 247	LINT_MAP1
0x310	“VMEbus Interrupt Enable Register (VINT_EN)” on page 248	VINT_EN
0x314	“VMEbus Interrupt Status Register (VINT_STAT)” on page 251	VINT_STAT
0x318	“VME Interrupt Map 0 Register (VINT_MAP0)” on page 253	VINT_MAP0
0x31C	“VME Interrupt Map 1 Register (VINT_MAP1)” on page 254	VINT_MAP1
0x320	“Interrupt STATUS/ID Out Register (STATID)” on page 255	STATID
0x324	“VIRQ1 STATUS/ID Register (V1_STATID)” on page 256	V1_STATID
0x328	“VIRQ2 STATUS/ID Register (V2_STATID)” on page 257	V2_STATID
0x32C	“VIRQ3 STATUS/ID Register (V3_STATID)” on page 258	V3_STATID
0x330	“VIRQ4 STATUS/ID Register (V4_STATID)” on page 259	V4_STATID
0x334	“VIRQ5 STATUS/ID Register (V5_STATID)” on page 260	V5_STATID
0x338	“VIRQ6 STATUS/ID Register (V6_STATID)” on page 261	V6_STATID
0x33C	“VIRQ7 STATUS/ID Register (V7_STATID)” on page 262	V7_STATID
0x340	“PCI Interrupt Map 2 Register (LINT_MAP2)” on page 263	LINT_MAP2
0x344	“VME Interrupt Map 2 Register (VINT_MAP2)” on page 264	VINT_MAP2
0x348	“Mailbox 0 Register (MBOX0)” on page 265	MBOX0
0x34C	“Mailbox 1 Register (MBOX1)” on page 266	MBOX1
0x350	“Mailbox 2 Register (MBOX2)” on page 267	MBOX2
0x354	“Mailbox 3 Register (MBOX3)” on page 268	MBOX3
0x358	“Semaphore 0 Register (SEMA0)” on page 269	SEMA0
0x35C	“Semaphore 1 Register (SEMA1)” on page 270	SEMA1
0x360-0x3FC	Reserved	
0x400	“Master Control Register (MAST_CTL)” on page 271	MAST_CTL
0x404	“Miscellaneous Control Register (MISC_CTL)” on page 273	MISC_CTL
0x408	“Miscellaneous Status Register (MISC_STAT)” on page 275	MISC_STAT

Table 34: Universe II Register Map *(Continued)*

Offset	Register	Name
0x40C	“User AM Codes Register (USER_AM)” on page 276	USER_AM
0x410-0x4F8	Reserved	
0x4FC	“Universe II Specific Register (U2SPEC)” on page 277	U2SPEC
0x500-0xEFC	Reserved	
0xF00	“VMEbus Slave Image 0 Control Register (VSI0_CTL)” on page 279	VSI0_CTL
0xF04	“VMEbus Slave Image 0 Base Address Register (VSI0_BS)” on page 282	VSI0_BS
0xF08	“VMEbus Slave Image 0 Bound Address Register (VSI0_BD)” on page 283	VSI0_BD
0xF0C	“VMEbus Slave Image 0 Translation Offset (VSI0_TO)” on page 284	VSI0_TO
0xF10	Reserved	
0xF14	“VMEbus Slave Image 1 Control (VSI1_CTL)” on page 285	VSI1_CTL
0xF18	“VMEbus Slave Image 1 Base Address Register (VSI1_BS)” on page 287	VSI1_BS
0xF1C	“VMEbus Slave Image 1 Bound Address Register (VSI1_BD)” on page 288	VSI1_BD
0xF20	“VMEbus Slave Image 1 Translation Offset (VSI1_TO)” on page 289	VSI1_TO
0xF24	Reserved	
0xF28	“VMEbus Slave Image 2 Control (VSI2_CTL)” on page 290	VSI2_CTL
0xF2C	“VMEbus Slave Image 2 Base Address Register (VSI2_BS)” on page 292	VSI2_BS
0xF30	“VMEbus Slave Image 2 Bound Address Register (VSI2_BD)” on page 293	VSI2_BD
0xF34	“VMEbus Slave Image 2 Translation Offset (VSI2_TO)” on page 294	VSI2_TO
0xF38	Reserved	
0xF3C	“VMEbus Slave Image 3 Control (VSI3_CTL)” on page 295	VSI3_CTL
0xF40	“VMEbus Slave Image 3 Base Address Register (VSI3_BS)” on page 297	VSI3_BS
0xF44	“VMEbus Slave Image 3 Bound Address Register (VSI3_BD)” on page 298	VSI3_BD
0xF48	“VMEbus Slave Image 3 Translation Offset (VSI3_TO)” on page 299	VSI3_TO

Table 34: Universe II Register Map *(Continued)*

Offset	Register	Name
0xF4C-0xF60	Reserved	
0xF64	“Location Monitor Control Register (LM_CTL)” on page 300	LM_CTL
0xF68	“Location Monitor Base Address Register (LM_BS)” on page 302	LM_BS
0xF6C	Reserved	
0xF70	“VMEbus Register Access Image Control Register (VRAI_CTL)” on page 303	VRAI_CTL
0xF74	“VMEbus Register Access Image Base Address Register (VRAI_BS)” on page 304	VRAI_BS
0xF78-0xF7C	Reserved	
0xF80	“VMEbus CSR Control Register (VCSR_CTL)” on page 305	VCSR_CTL
0xF84	“VMEbus CSR Translation Offset (VCSR_TO)” on page 306	VCSR_TO
0xF88	“VMEbus AM Code Error Log (V_AMERR)” on page 307	V_AMERR
0xF8C	“VMEbus Address Error Log (VAERR)” on page 308	VAERR
0xF90	“VMEbus Slave Image 4 Control (VSI4_CTL)” on page 309r	VSI4_CTL
0xF94	“VMEbus Slave Image 4 Base Address Register (VSI4_BS)” on page 311	VSI4_BS
0xF98	“VMEbus Slave Image 4 Bound Address Register (VSI4_BD)” on page 312	VSI4_BD
0xF9C	“VMEbus Slave Image 4 Translation Offset (VSI4_TO)” on page 313	VSI4_TO
0xFA0	Reserved	
0xFA4	“VMEbus Slave Image 5 Control (VSI5_CTL)” on page 314	VSI5_CTL
0xFA8	“VMEbus Slave Image 5 Base Address Register (VSI5_BS)” on page 316	VSI5_BS
0xFAC	“VMEbus Slave Image 5 Bound Address Register (VSI5_BD)” on page 317	VSI5_BD
0xFB0	“VMEbus Slave Image 5 Translation Offset (VSI5_TO)” on page 318	VSI5_TO
0xFB4	Reserved	
0xFB8	“VMEbus Slave Image 6 Control (VSI6_CTL)” on page 319	VSI6_CTL
0xFBC	“VMEbus Slave Image 6 Base Address Register (VSI6_BS)” on page 321	VSI6_BS
0xFC0	“VMEbus Slave Image 6 Bound Address Register (VSI6_BD)” on page 322	VSI6_BD

Table 34: Universe II Register Map *(Continued)*

Offset	Register	Name
0xFC4	“VMEbus Slave Image 6 Translation Offset (VSI6_TO)” on page 323	VSI6_TO
0xFC8	Reserved	
0xFCC	“VMEbus Slave Image 7 Control (VSI7_CTL)” on page 324	VSI7_CTL
0xFD0	“VMEbus Slave Image 7 Base Address Register (VSI7_BS)” on page 326	VSI7_BS
0xFD4	“VMEbus Slave Image 7 Bound Address Register (VSI7_BD)” on page 327	VSI7_BD
0xFD8	“VMEbus Slave Image 7 Translation Offset (VSI7_TO)” on page 328	VSI7_TO
0xFDC-0xFEC	Reserved	
0xFF0	VME CR/CSR Reserved	
0xFF4	“VMEbus CSR Bit Clear Register (VCSR_CLR)” on page 329	VCSR_CLR
0xFF8	“VMEbus CSR Bit Set Register (VCSR_SET)” on page 330	VCSR_SET
0xFFC	“VMEbus CSR Base Address Register (VCSR_BS)” on page 331	VCSR_BS

12.3 Register Description

The following tables describe the Universe II registers.

12.3.1 PCI Configuration Space ID Register (PCI_ID)

Register name: PCI_VID	Register offset: 0x00

Bits	7	6	5	4	3	2	1	0
31:24	DID							
23:16	DID							
15:08	VID							
07:00	VID							

Bits	Name	Description	Type	Reset by	Reset value
31:16	DID[15:0]	Device ID IDT allocated device identifier	R	all	0x10E3
15:00	VID[15:0]	Vendor ID PCI SIG allocated vendor identifier Note: IDT acquired Tundra Semiconductor.	R	all	0x10E3

12.3.2 PCI Configuration Space Control and Status Register (PCI_CSR)

Register name: PCI_CSR	Register offset: 0x004

Bits	7	6	5	4	3	2	1	0
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

Bits	Name	Description	Type	Reset by	Reset value
31	D_PE	Detected 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. 0=No parity error 1=Parity error	R/Write 1 to Clear	All	0x0
30	S_SERR	Signalled SERR_ 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. 0=SERR_ not asserted 1=SERR_ asserted.	R/Write 1 to Clear	All	0x0
29	R_MA	Received Master-Abort The Universe II PCI master interface sets this bit when a transaction it initiated had to be terminated with a Master-Abort. 0=Master did not generate Master-Abort 1=Master generated Master-Abort	R/Write 1 to Clear	All	0x0
28	R_TA	Received Target-Abort The Universe II PCI master interface sets this bit when a transaction it initiated was terminated with a Target-Abort. 0=Master did not detect Target-Abort 1=Master detected Target-Abort.	R/Write 1 to Clear	All	0x0
27	S_TA	Signalled Target-Abort 0=Target did not terminate transaction with Target-Abort 1=Target terminated transaction with Target-Abort.	R/Write 1 to Clear	All	0x0

Bits	Name	Description	Type	Reset by	Reset value
26:25	DEVSEL	Device Select Timing The Universe II is a medium speed device	R	All	0x1
24	DP_D	Master Data Parity Error The Universe II PCI master interface sets this bit if the Parity Error Response bit is set, if it is the master of transaction in which it asserts PERR_, or the addressed target asserts PERR_. 0=Master did not detect/generate data parity error, 1=Master detected/generated data parity error.	R/Write 1 to Clear	All	0x0
23	TFBBC	Target Fast Back to Back Capable Universe II cannot accept Back to Back cycles from a different agent.	R	All	0x0
22:10	Reserved	Reserved	R	All	0x0
09	MFBBC	Master Fast Back-to-Back Enable The Universe II master never generates fast back to back transactions. 0=No fast back-to-back transactions	R	All	0x0
08	SERR_EN	SERR_ Enable Setting this and PERESP allows the Universe II PCI target interface to report address parity errors with SERR_. 0=Disable SERR_ driver 1=Enable SERR_ driver Note: The Universe II only refuses PCI addresses with parity errors when both the PERESP and SERR_EN bits are programmed to a value of 1.	R/W	All	0x0
07	WAIT	Wait Cycle Control 0=No address/data stepping	R	All	0x0
06	PERESP	Parity Error Response 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. 0=Disable 1=Enable Note: The Universe II only refuses PCI addresses with parity errors when both the PERESP and SERR_EN bits are programmed to a value of 1.	R/W	All	0x0

Bits	Name	Description	Type	Reset by	Reset value
05	VGAPS	VGA Palette Snoop The Universe II treats palette accesses like all other accesses. 0=Disable	R	All	0x0
04	MWI_EN	Memory Write and Invalidate Enable The Universe II PCI master interface never generates a Memory Write and Invalidate command. 0=Disable	R	All	0x0
03	SC	Special Cycles 0=Disable The Universe II PCI target interface never responds to special cycles.	R	All	0x0
02	BM	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.	R/W	PWR VME	This bit is 1 after reset if the VMEbus Address [14] equals 1 during power-on reset. This bit is reloaded by “all”.
01	MS	Target Memory Enable 0=Disable 1=Enable	R/W	PWR VME	This bit is 1 after reset if the VMEbus Address [13:12] equals 10 during power-on reset. This bit is reloaded by “all”

Bits	Name	Description	Type	Reset by	Reset value
00	IOS	Target IO Enable 0=Disable 1=Enable	R/W	PWR VME	This bit is 1 after reset if the VMEbus Address [13:12] equals 10 during power-on reset. This bit is reloaded by “all”

12.3.3 PCI Configuration Class Register (PCI_CLASS)

Register name: PCI_CLASS	Register offset: 0x08

Bits	7	6	5	4	3	2	1	0
31:24	BASE							
23:16	SUB							
15:08	PROG							
07:00	RID							

Bits	Name	Description	Type	Reset by	Reset value
31:24	BASE [7:0]	Base Class Code The Universe II is defined as a PCI bridge device	R	All	0x06
23:16	SUB [7:0]	Sub Class Code The Universe II sub-class is “other bridge device”	R	All	0x80
15:08	PROG [7:0]	Programming Interface The Universe II does not have a standardized register-level programming interface	R	All	0x00
07:00	RID [7:0]	Revision ID	R	All	0x02

12.3.4 PCI Configuration Miscellaneous 0 Register (PCI_MISC0)

The Universe II is not a multi-function device.

Register name: PCI_MISC0	Register offset: 0x0C

Bits	7	6	5	4	3	2	1	0
31:24	BISTC	SBIST	Reserved		CCODE			
23:16	MFUNCT	LAYOUT						
15:08	LTIMER					Reserved		
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31	BISTC	The Universe II is not BIST Capable	R	All	0x0
30	SBIST	Start BIST The Universe II is not BIST capable	R	All	0x0
29:28	Reserved	Reserved	R	All	0x0
27:24	CCODE	Completion Code The Universe II is not BIST capable	R	All	0x0
23	MFUNCT	Multifunction Device The Universe II is not a multi-function device. 0=No 1=Yes	R	All	0x0
22:16	LAYOUT	Configuration Space Layout	R	All	0x0
15:11	LTIMER	Latency Timer The latency timer has a resolution of eight clocks. When the Universe II latency timer is programmed for eight clock periods and FRAME_, IRDY_ and TRDY_ are asserted while GNT_ is not asserted, FRAME_ is negated on the next clock edge.	R/W	All	0x0
10:00	Reserved	Reserved	R	All	0x0

12.3.5 PCI Configuration Base Address Register (PCI_BS0)

This register specifies the 4 Kbyte aligned base address of the 4 Kbyte Universe II register space on PCI.

Register name: PCI_BS0	Register offset: 0x10

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	BS					Reserved		
07:00	Reserved							SPACE

Bits	Name	Description	Type	Reset by	Reset value
31:12	BS[31:12]	Base Address	R/W	All	0x00
11:02	Reserved	Reserved	R	All	0x00
00	SPACE	PCI Bus Address Space 0=Memory 1=I/O	R	All	Undefined

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 135). If mapped into Memory space, the user is free to locate the 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.

12.3.6 PCI Configuration Base Address 1 Register (PCI_BS1)

This register specifies the 4 KByte aligned base address of the 4 KByte Universe II register space in PCI

Register name: PCI_BS1	Register offset: 0x14

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	BS				Reserved			
07:00	Reserved							SPACE

Bits	Name	Description	Type	Reset by	Reset value
31:12	BS[31:12]	Base Address	R/W	All	0x00
11:02	Reserved	Reserved	R	All	0x00
00	SPACE	PCI Bus Address Space 0=Memory 1=I/O	R	All	Undefined

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 135). 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.

12.3.7 PCI Configuration Miscellaneous 1 Register (PCI_MISC1)

Register name: PCI_MISC1	Register offset: 0x3C

Bits	7	6	5	4	3	2	1	0
31:24	MAX_LAT [7:0]							
23:16	MIN_GNT [7:0]							
15:08	INT_PIN [7:0]							
07:00	INT_LINE [7:0]							

Bits	Name	Description	Type	Reset by	Reset value
31:24	MAX_LAT [7:0]	Maximum Latency: This device has no special latency requirements	R	All	0x00
23:16	MIN_GNT [7:0]	Minimum Grant 250 ns increments 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.	R	All	0b11
15:08	INT_PIN [7:0]	Interrupt Pin Universe II pin INT_ [0] has a PCI compliant I/O buffer	R	All	0b01
07:00	INT_LINE [7:0]	Interrupt Line Used by some PCI systems to record interrupt routing information	R/W	All	0x00

12.3.8 PCI Target Image 0 Control (LSI0_CTL)

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.

Register name: LSI0_CTL	**Register offset: 0x100**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31	EN	Image Enable 0=Disable 1=Enable	R/W	All	Undefined
30	PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0x00
29:24	Reserved	Reserved	R	All	0x00
23:22	VDW	VMEbus Maximum Data width 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	0b10
21:19	Reserved	Reserved	R	All	0x00

Bits	Name	Description	Type	Reset by	Reset value
18:16	VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	Undefined
15	Reserved	Reserved	R	All	0x00
14	PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0x00
13	Reserved	Reserved	R	All	0x00
12	SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0x00
11:09	Reserved	Reserved	R	All	0x00
08	VCT	VMEbus Cycle Type 0=No BLTs on VMEbus 1=Single BLTs on VMEbus	R/W	All	0x00
07:01	Reserved	Reserved	R	All	0x00
00	LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	Undefined

12.3.9 PCI Target Image 0 Base Address Register (LSI0_BS)

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 4 Kbyte resolution. PCI Target Images 1, 2, 3, 5, 6, and 7 have a 64 Kbyte resolution.

Register name: LSI0_BS	Register offset: 0x104

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	BS					Reserved		
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:28	BS[31:28]	Base Address	R/W	All	Undefined
27:12	BS[27:12]	Base Address	R/W	All	0x00
11:00	Reserved	Reserved	R	All	0x00

12.3.10 PCI Target Image 0 Bound Address Register (LSI0_BD)

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.

Register name: LSI0_BD	**Register offset: 0x108**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	BD							
23:16	BD							
15:08	BD					Reserved		
07:00	Reserved							

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31:28	BD[31:28]	Bound Address	R/W	All	Undefined
27:12	BD[27:12]	Bound Address	R/W	All	0x00
11:00	Reserved	Reserved	R	All	0x00

12.3.11 PCI Target Image 0 Translation Offset (LSI0_TO)

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.

Register name: LSI0_TO	**Register offset: 0x10C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	TO							
23:16	TO							
15:08	TO				Reserved			
07:00	Reserved							

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31:12	TO[31:12]	Translation Offset	R/W	All	0x00
11:00	Reserved	Reserved	R	All	0x00

12.3.12 PCI Target Image 1 Control (LSI1_CTL)

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.

Register name: LSI1_CTL	**Register offset: 0x114**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31	EN	Image Enable 0=Disable 1=Enable	R/W	All	Undefined
30	PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0x00
29:24	Reserved	Reserved	R	All	0x00
23:22	VDW	VMEbus Maximum Data width 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	0b10
21:19	Reserved	Reserved	R	All	0x00

Bits	Name	Description	Type	Reset by	Reset value
18:16	VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	Undefined
15	Reserved	Reserved	R	All	0x00
14	PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0x00
13	Reserved	Reserved	R	All	0x00
12	SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0x00
11:09	Reserved	Reserved	R	All	0x00
08	VCT	VMEbus Cycle Type 0=No BLTs on VMEbus 1=Single BLTs on VMEbus	R/W	All	0x00
07:01	Reserved	Reserved	R	All	0x00
00	LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	Undefined

12.3.13 PCI Target Image 1 Base Address Register (LSI1_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: LSI1_BS	Register offset: 0x118

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	BS[31:16]	Base Address	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.14 PCI Target Image 1 Bound Address Register (LSI1_BD)

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.

Register name: LSI1_BD	**Register offset: 0x11C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	BD							
23:16	BD							
15:08	Reserved							
07:00	Reserved							

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31:16	BD[31:16]	Bound Address	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.15 PCI Target Image 1 Translation Offset (LSI1_TO)

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.

Register name: LSI1_TO	Register offset: 0x120

Bits	7	6	5	4	3	2	1	0
31:24	TO							
23:16	TO							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	TO[31:16]	Translation Offset	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.16 PCI Target Image 2 Control (LSI2_CTL)

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.

Register name: LSI2_CTL	**Register offset: 0x13C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31	EN	Image Enable 0=Disable 1=Enable	R/W	All	Undefined
30	PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0x00
29:24	Reserved	Reserved	R	All	0x00
23:22	VDW	VMEbus Maximum Data width 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	0b10
21:19	Reserved	Reserved	R	All	0x00

Bits	Name	Description	Type	Reset by	Reset value
18:16	VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	Undefined
15	Reserved	Reserved	R	All	0x00
14	PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0x00
13	Reserved	Reserved	R	All	0x00
12	SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0x00
11:09	Reserved	Reserved	R	All	0x00
08	VCT	VMEbus Cycle Type 0=No BLTs on VMEbus 1=Single BLTs on VMEbus	R/W	All	0x00
07:01	Reserved	Reserved	R	All	0x00
00	LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	Undefined

12.3.17 PCI Target Image 2 Base Address Register (LSI2_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: LSI12_BS	Register offset: 0x12C

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	BS[31:16]	Base Address	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.18 PCI Target Image 2 Bound Address Register (LSI2_BD)

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.

Register name: LSI2_BD	Register offset: 0x130

Bits	7	6	5	4	3	2	1	0
31:24	BD							
23:16	BD							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	BD[31:16]	Bound Address	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.19 PCI Target Image 2 Translation Offset (LSI2_TO)

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.

Register name: LSI2_TO	Register offset: 0x134

Bits	7	6	5	4	3	2	1	0
31:24	TO							
23:16	TO							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	TO[31:16]	Translation Offset	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.20 PCI Target Image 3 Control (LSI3_CTL)

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.

Register name: LSI3_CTL	**Register offset: 0x13C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31	EN	Image Enable 0=Disable 1=Enable	R/W	All	Undefined
30	PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0x00
29:24	Reserved	Reserved	R	All	0x00
23:22	VDW	VMEbus Maximum Data width 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	10
21:19	Reserved	Reserved	R	All	0x00

Bits	Name	Description	Type	Reset by	Reset value
18:16	VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	Undefined
15	Reserved	Reserved	R	All	0x00
14	PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0x00
13	Reserved	Reserved	R	All	0x00
12	SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0x00
11:09	Reserved	Reserved	R	All	0x00
08	VCT	VMEbus Cycle Type 0=No BLTs on VMEbus 1=Single BLTs on VMEbus	R/W	All	0x00
07:01	Reserved	Reserved	R	All	0x00
00	LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	Undefined

12.3.21 PCI Target Image 3 Base Address Register (LSI3_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: LSI3_BS	Register offset: 0x140

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	BS[31:16]	Base Address	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.22 PCI Target Image 3 Bound Address Register (LSI3_BD)

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.

Register name: LSI3_BD	Register offset: 0x144

Bits	7	6	5	4	3	2	1	0
31:24	BD							
23:16	BD							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	BD[31:16]	Bound Address	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.23 PCI Target Image 3 Translation Offset (LSI3_TO)

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.

Register name: LSI3_TO	Register offset: 0x148

Bits	7	6	5	4	3	2	1	0
31:24	TO							
23:16	TO							
15:08	Reserved							
07:00	Reserved							

Bits	Name	Description	Type	Reset by	Reset value
31:16	TO[31:16]	Translation Offset	R/W	All	0x00
15:00	Reserved	Reserved	R	All	0x00

12.3.24 Special Cycle Control Register (SCYC_CTL)

The special cycle generator generates 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.

Register Name: SCYC_CTL	**Register offset: 0x170**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							
07:00	Reserved					LAS	SCYC	

Bits	**Name**	**Description**	**Type**	**Reset by**	**Reset value**
31:03	Reserved	Reserved	R	All	0x00
02	LAS	PCI Bus Address Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space For a RMW cycle only	R/W	All	0x00
01:00	SCYC	Special Cycle 00=Disable 01=RMW 10=ADOH 11=Reserved	R/W	All	0x00

12.3.25 Special Cycle PCI Bus Address Register (SCYC_ADDR)

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_.

Register name: SCYC_ADDR	Register offset: 0x174

Bits	7	6	5	4	3	2	1	0
31:24	ADDR							
23:16	ADDR							
15:08	ADDR							
07:00	ADDR						Reserved	

Bits	Name	Description	Type	Reset by	Reset value
31:02	ADDR	Address	R/W	All	0x00
01:00	Reserved	Reserved	R/W	All	0x00

12.3.26 Special Cycle Swap/Compare Enable Register (SCYC_EN)

The bits enabled in this register determine the bits that are involved in the compare and swap operations for VME RMW cycles.

Register name: SCYC_EN	Register offset: 0x178

Bits	7	6	5	4	3	2	1	0
31:24	EN							
23:16	EN							
15:08	EN							
07:00	EN							

Bits	Name	Description	Type	Reset by	Reset value
31:00	EN	Bit Enable 0=Disable 1=Enable	R/W	All	0x00

12.3.27 Special Cycle Compare Data Register (SCYC_CMP)

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.

Register name: SCYC_CMP	Register offset: 0x17C

Bits	7	6	5	4	3	2	1	0
31:24	CMP							
23:16	CMP							
15:08	CMP							
07:00	CMP							

Bits	Name	Description	Type	Reset by	Reset value
31:00	CMP	The data returned from the VMEbus is compared with the contents of this register.	R/W	All	0x00

12.3.28 Special Cycle Swap Data Register (SCYC_SWP)

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.

Register name: SCYC_SWP	Register offset: 0x180

Bits	7	6	5	4	3	2	1	0
31:24	SWP							
23:16	SWP							
15:08	SWP							
07:00	SWP							

Bits	Name	Description	Type	Reset by	Reset value
31:00	SWP	Swap data	R/W	All	0x00

12.3.29 PCI Miscellaneous Register (LMISC)

This register can only be set at configuration or after disabling all PCI Target Images.

Register name: LMISC	Register offset: 0x184

Bits	7	6	5	4	3	2	1	0
31:24	CRT[3:0]				Reserved	CWT		
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset By	Reset State
CRT[3:0]	CRT This field is provided for backward compatibility with the Universe I. It has no effect on the operation of the Universe II.	R/W	all	000
CWT [2:0]	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=256 PCI Clocks 110=512 PCI Clocks others= Reserved	R/W	all	000

Bits	Name	Description	Type	Reset by	Reset value
31:28	CRT	This field is provided for backward compatibility with the Universe I. It has no effect on the operation of the Universe II.	R/W	All	0x00
27	Reserved	Reserved	R	All	0x00

Bits	Name	Description	Type	Reset by	Reset value
26:24	CWT	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=256 PCI Clocks 110=512 PCI Clocks others= Reserved	R/W	All	0x00
23:00	Reserved	Reserved	R	All	0x00

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 behavior is unpredictable if CWT is changed during coupled cycle activity.

12.3.30 Special PCI Target Image (SLSI)

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.

Register name: SLSI	**Register offset: 0x188**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWN	Reserved					
23:16	VDW				Reserved			
15:08	PGM				SUPER			
07:00	BS						Reserved	LAS

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	All	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0
VDW	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	R/W	All	0

Name	Description	Type	Reset by	Reset value
PGM	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	R/W	All	0
SUPER	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	R/W	All	0
BS	Base Address Specifies a 64 MByte aligned base address for this 64 MByte image.	R/W	All	0
LAS	PCI Bus Address Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	0

12.3.31 PCI Command Error Log Register (L_CMDERR)

The Universe II PCI Master Interface is responsible for logging errors under the following conditions:

- Posted write transaction results in a target abort,
- Posted write transaction results in a master abort, or
- Maximum retry counter expires during retry of posted write transaction.

This register logs the command information.

Register name: L_CMDERR	**Register offset: 0x18C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	CMDERR				M_ERR	Reserved		
23:16	L_STAT	Reserved						
15:08	Reserved							
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
CMDERR	PCI Command Error Log	R	All	0111
M_ERR	Multiple Error Occurred 0=Single error 1=At least one error has occurred since the logs were frozen.	R	All	0
L_STAT	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	R/W	All	0

12.3.32 PCI Address Error Log (LAERR)

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.

Register name: LAERR	**Register offset: 0x190**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	LAERR							
23:16	LAERR							
15:08	LAERR							
07:00	LAERR						Reserved	

Name	**Description**	**Type**	**Reset by**	**Reset value**
LAERR	PCI Address Error Log	R	All	0

12.3.33 PCI Target Image 4 Control Register (LSI4_CTL)

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.

Register name: LSI4_CTL	**Register offset: 0x1A0**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	All	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0
VDW	VMEbus Maximum Datawidth 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	10
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	0

Name	Description	Type	Reset by	Reset value
PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0
SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0
VCT	VMEbus Cycle Type 0=no BLTs on VMEbus 1=BLTs on VMEbus	R/W	All	0
LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	0

12.3.34 PCI Target Image 4 Base Address Register (LSI4_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: LSI4_BS	Register offset: 0x1A4

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	BS					Reserved		
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
BS	Base Address	R/W	All	0

12.3.35 PCI Target Image 4 Bound Address Register (LSI4_BD)

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.

Register name: LSI4_BD	**Register offset: 0x1A8**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	BD							
23:16	BD							
15:08	BD				Reserved			
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
BD	Bound Address	R/W	All	0

12.3.36 PCI Target Image 4 Translation Offset (LSI4_TO)

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.

Register name: LSI4_TO	Register offset: 0x1AC

Bits	7	6	5	4	3	2	1	0
31:24	TO							
23:16	TO							
15:08	TO					Reserved		
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
TO	Translation offset	R/W	All	0

12.3.37 PCI Target Image 5 Control Register (LSI5_CTL)

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.

Register name: LSI5_CTL	**Register offset: 0x1B4**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disabl 1=Enable	R/W	All	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0
VDW	VMEbus Maximum Datawidth 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	10
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	0

Name	Description	Type	Reset by	Reset value
PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0
SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0
VCT	VMEbus Cycle Type 0=no BLTs on VMEbus 1=BLTs on VMEbus	R/W	All	0
LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	0

12.3.38 PCI Target Image 5 Base Address Register (LSI5_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: LSI5_BS	Register offset: 0x1B8

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS	Base Address	R/W	All	0

12.3.39 PCI Target Image 5 Bound Address Register (LSI5_BD)

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.

Register name: LSI5_BD	Register offset: 0x1BC

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD	Bound Address	R/W	All	0

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.

12.3.40 PCI Target Image 5 Translation Offset (LSI5_TO)

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.

Register name: LSI5_TO	Register offset: 0x1C0

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO	Translation offset	R/W	All	0

12.3.41 PCI Target Image 6 Control Register (LSI6_CTL)

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.

Register name: LSI6_CTL	**Register offset: 0x1C8**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23:16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	All	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0
VDW	VMEbus Maximum Datawidth 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	10
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=CR/CSR 110=User1 111=User2	R/W	All	0

Name	Description	Type	Reset by	Reset value
PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0
SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0
VCT	VMEbus Cycle Type 0=no BLTs on VMEbus 1=BLTs on VMEbus	R/W	All	0
LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	0

12.3.42 PCI Target Image 6 Base Address Register (LSI6_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: LSI6_BS	Register offset: 0x1CC

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS	Base Address	R/W	All	0

12.3.43 PCI Target Image 6 Bound Address Register (LSI6_BD)

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.

Register name: LSI6_BD	Register offset: 0x1D0

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD	Bound Address	R/W	All	0

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.

12.3.44 PCI Target Image 6 Translation Offset (LSI6_TO)

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.

Register name: LSI6_TO	Register offset: 0x1D4

Bits	Function
31:24	TO
23:16	TO
15;08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO	Translation offset	R/W	All	0

12.3.45 PCI Target Image 7 Control Register (LSI7_CTL)

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.

Register name: LSI7_CTL	**Register offset: 0x1DC**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	Reserved					
23;16	VDW		Reserved			VAS		
15:08	Reserved	PGM	Reserved	SUPER	Reserved			VCT
07:00	Reserved							LAS

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	All	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	All	0
VDW	VMEbus Maximum Datawidth 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	10
VAS	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2	R/W	All	0
PGM	Program/Data AM Code 0=Data 1=Program	R/W	All	0
SUPER	Supervisor/User AM Code 0=Non-Privileged 1=Supervisor	R/W	All	0

Name	Description	Type	Reset by	Reset value
VCT	VMEbus Cycle Type 0=no BLTs on VMEbus 1=BLTs on VMEbus	R/W	All	0
LAS	PCI Bus Memory Space 0=PCI Bus Memory Space 1=PCI Bus I/O Space	R/W	All	0

12.3.46 PCI Target Image 7 Base Address Register (LSI7_BS)

The base address specifies the lowest address in the address range that is decoded.

Register name: LSI7_BS	Register offset: 0x1E0

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS	Base Address	R/W	All	0

12.3.47 PCI Target Image 7 Bound Address Register (LSI7_BD)

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.

Register name: LSI7_BD	Register offset: 0x1E4

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD	Bound Address	R/W	All	0

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.

12.3.48 PCI Target Image 7 Translation Offset (LSI7_TO)

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.

Register name: LSI7_TO	Register offset: 0x1E8

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO	Translation offset	R/W	All	0

12.3.49 DMA Transfer Control Register (DCTL)

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.

Register name: DCTL	**Register offset: 0x200**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	L2V	Reserved						
23:16	VDW		Reserved			VAS		
15:08	PGM		SUPER		Reserved		NO_ VINC	VCT
07:00	LD64EN	Reserved						

Name	**Description**	**Type**	**Reset by**	**Reset value**
L2V	Direction 0=Transfer from VMEbus to PCI Bus 1=Transfer from PCI Bus to VMEbus	R/W	All	0
VDW	VMEbus Maximum Datawidth 00=8-bit data width 01=16 bit data width 10=32-bit data width 11=64-bit data width	R/W	All	0
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	All	0
PGM	Program/Data AM Code 00=Data 01=Program others= Reserved	R/W	All	0

Name	Description	Type	Reset by	Reset value
SUPER	Supervisor/User AM Code 00=Non-Privileged 01=Supervisor others= Reserved	R/W	All	0
NO_VINC	VMEbus Non-Incrementing Mode (Non-Inc Mode) 0=disabled 1=enabled	R/W	All	0
VCT	VMEbus Cycle Type 0=no BLTs on VMEbus 1=BLTs on VMEbus	R/W	All	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	All	1

12.3.50 DMA Transfer Byte Count Register (DTBC)

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.

Register name: DTBC	Register offset: 0x204

Bits	Function
31:24	Reserved
23:16	DTBC
15:08	DTBC
07:00	DTBC

Name	Description	Type	Reset by	Reset value
DTBC	DMA Transfer Byte Count	R/W	All	0

In direct mode the user must re-program 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.

12.3.51 DMA PCI Bus Address Register (DLA)

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.

Register name: DLA	Register offset: 0x208

Bits	Function
31:24	LA
23:16	LA
15:08	LA
07:00	LA

Name	Description	Type	Reset by	Reset value
LA[31:3]	PCI Bus Address	R/W	All	0
LA[2:0]	PCI Bus Address Address bits [2:0] must be programmed the same as those in the DVA.	R/W	All	0

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.

12.3.52 DMA VMEbus Address Register (DVA)

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.

Register name: DVA	Register offset: 0x210

Bits	Function
31:24	VA
23:16	VA
15:08	VA
07:00	VA

Name	Description	Type	Reset by	Reset value
VA	VMEbus Address Address bits [2:0] must be programmed the same as those in the DLA.	R/W	All	0

After a bus error, a Target-Abort, or a Master-Abort, the value in the DVA register must not be used to reprogram the DMA because it has no usable information. Some offset from its original value must be used.

12.3.53 DMA Command Packet Pointer (DCPP)

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.

Register name: DCPP	**Register offset: 0x218**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	DCPP							
23:16	DCPP							
15:08	DCPP							
07:00	DCPP			Reserved				

Name	**Description**	**Type**	**Reset by**	**Reset value**
DCPP [31:5]	DMA Command Packet Pointer	R/W	All	0

12.3.54 DMA General Control/Status Register (DGCS)

Register name: DGCS	Register offset: 0x220

Bits	7	6	5	4	3	2	1	0
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

Name	Description	Type	Reset by	Reset value
GO	DMA Go Bit 0=No effect 1=Enable DMA Transfers	W/Read 0 always	All	0
STOP_ REQ	DMA Stop Request 0=No effect 1=Stop DMA transfer when all buffered data has been written	W/Read 0 always	All	0
HALT_ REQ	DMA Halt Request 0=No effect 1=Halt the DMA transfer at the completion of the current command packet	W/Read 0 always	All	0
CHAIN	DMA Chaining 0=DMA Direct Mode 1=DMA Linked List mode	R/W	All	0
VON [2:0]	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	R/W	All	0

Name	Description	Type	Reset by	Reset value
VOFF [3:0]	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.	R/W	All	0
ACT	DMA Active Status Bit 0=Not Active 1=Active	R	All	0
STOP	DMA Stopped Status Bit 0=Not Stopped 1=Stopped	R/Write 1 to Clear	All	0
HALT	DMA Halted Status Bit 0=Not Halted 1=Halted	R/Write 1 to Clear	All	0
DONE	DMA Done Status Bit 0=Not Complete 1=Complete	R/Write 1 to Clear	All	0
LERR	DMA PCI Bus Error Status Bit 0=No Error 1=Error	R/Write 1 to Clear	All	0
VERR	DMA VMEbus Error Status Bit 0=No Error 1=Error	R/Write 1 to Clear	All	0
P_ERR	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	R/Write 1 to Clear	All	0

Name	Description	Type	Reset by	Reset value
INT_STOP	Interrupt when Stopped 0=Disable 1=Enable	R/W	All	0
INT_HALT	Interrupt when Halted 0=Disable 1=Enable	R/W	All	0
INT_ DONE	Interrupt when Done 0=Disable 1=Enable	R/W	All	0
INT_LERR	Interrupt on LERR 0=Disable 1=Enable	R/W	All	0
INT_VERR	Interrupt on VERR 0=Disable 1=Enable	R/W	All	0
INT_P_ ERR	Interrupt on Master Enable Error 0=Disable 1=Enable	R/W	All	0

STOP, HALT, DONE, LERR, VERR, and P_ERR must be cleared before the GO bit is enabled.

12.3.55 DMA Linked List Update Enable Register (D_LLUE)

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 100).

Register name: D_LLUE	Register offset: 0x224

Bits	7	6	5	4	3	2	1	0
31:24	UPDATE	Reserved						
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
UPDATE	DMA Linked List Update Enable 0=PCI Resource not Updating Linked List 1=PCI Resource Updating Linked List	R/W	All	0

12.3.56 PCI Interrupt Enable Register (LINT_EN)

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.

Register name: LINT_EN	**Register offset: 0x300**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
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

Name	**Description**	**Type**	**Reset by**	**Reset value**
LM3	Location Monitor 3 0=LM3 Interrupt Disabled 1=LM3 Interrupt Enabled	R/W	All	0
LM2	Location Monitor 2 0=LM2 Interrupt Disabled 1=LM2 Interrupt Enabled	R/W	All	0
LM1	Location Monitor 1 0=LM1 Interrupt Disabled 1=LM1 Interrupt Enabled	R/W	All	0
LM0	Location Monitor 0 0=LM0 Interrupt Disabled 1=LM0 Interrupt Enabled	R/W	All	0
MBOX3	Mailbox 3 0=MBOX3 Interrupt Disabled 1=MBOX3 Interrupt Enabled	R/W	All	0
MBOX2	Mailbox 2 0=MBOX2 Interrupt Disabled 1=MBOX2 Interrupt Enabled	R/W	All	0

Name	Description	Type	Reset by	Reset value
MBOX1	Mailbox 1 0=MBOX1 Interrupt Disabled 1=MBOX1 Interrupt Enabled	R/W	All	0
MBOX0	Mailbox 0 0=MBOX0 Interrupt Disabled 1=MBOX0 Interrupt Enabled	R/W	All	0
ACFAIL	ACFAIL Interrupt 0=ACFAIL Interrupt Disabled 1=ACFAIL Interrupt Enabled	R/W	All	0
SYSFAIL	SYSFAIL Interrupt 0=SYSFAIL Interrupt Disabled 1=SYSFAIL Interrupt Enabled	R/W	All	0
SW_INT	Local Software Interrupt 0=PCI Software Interrupt Disabled 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.	R/W	All	0
SW_IACK	“VME Software IACK” 0 =“VME Software IACK” Interrupt Disabled 1 =“VME Software IACK” Interrupt Enabled	R/W	All	0
VERR	PCI VERR Interrupt 0 =PCI VERR Interrupt Disabled 1=PCI VERR Interrupt Enabled	R/W	All	0
LERR	PCI LERR Interrupt 0 =PCI LERR Interrupt Disabled 1 =PCI LERR Interrupt Enabled	R/W	All	0
DMA	PCI DMA Interrupt 0=PCI DMA Interrupt Disabled 1=PCI DMA Interrupt Enabled	R/W	All	0
VIRQ7-VIRQ1	VIRQ*x* Interrupt 0=VIRQ*x* Interrupt Disabled 1 =VIRQ*x* Interrupt Enabled	R/W	All	0
VOWN	VOWN Interrupt 0=VOWN Interrupt Disabled 1=VOWN Interrupt Enabled	R/W	All	0

12.3.57 PCI Interrupt Status Register (LINT_STAT)

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 still asserted will have no effect.

Register name: LINT_STAT	**Register offset: 0x304**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
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

Name	**Description**	**Type**	**Reset by**	**Reset value**
LM3	Location Monitor 3 Status/Clear 0=no Location Monitor 3 Interrupt, 1=Location Monitor 3 Interrupt active	R/Write 1 to Clear	All	0
LM2	Location Monitor 2 Status/Clear 0=no Location Monitor 2 Interrupt, 1=Location Monitor 2 Interrupt active	R/Write 1 to Clear	All	0
LM1	Location Monitor 1 Status/Clear 0=no Location Monitor 1 Interrupt, 1=Location Monitor 1 Interrupt active	R/Write 1 to Clear	All	0
LM0	Location Monitor 0 Status/Clear 0=no Location Monitor 0 Interrupt, 1=Location Monitor 0 Interrupt active	R/Write 1 to Clear	All	0
MBOX3	Mailbox 3 Status/Clear 0=no Mailbox 3 Interrupt, 1=Mailbox 3 Interrupt active	R/Write 1 to Clear	All	0
MBOX2	Mailbox 2 Status/Clear 0=no Mailbox 2 Interrupt, 1=Mailbox 2 Interrupt active	R/Write 1 to Clear	All	0
MBOX1	Mailbox 1 Status/Clear 0=no Mailbox 1 Interrupt, 1=Mailbox 1 Interrupt active	R/Write 1 to Clear	All	0

Name	Description	Type	Reset by	Reset value
MBOX0	Mailbox 0 Status/Clear 0=no Mailbox 0 Interrupt, 1=Mailbox 0 Interrupt active	R/Write 1 to Clear	All	0
ACFAIL	ACFAIL Interrupt Status/Clear 0=no ACFAIL Interrupt, 1=ACFAIL Interrupt active	R/Write 1 to Clear	All	0
SYSFAIL	SYSFAIL Interrupt Status/Clear 0=no SYSFAIL Interrupt, 1=SYSFAIL Interrupt active	R/Write 1 to Clear	All	0
SW_INT	Local Software Interrupt Status/Clear 0=no PCI Software Interrupt, 1=PCI Software Interrupt active	R/Write 1 to Clear	All	0
SW_IACK	“VME Software IACK” Status/Clear 0=no “VME Software IACK” Interrupt, 1=“VME Software IACK” Interrupt active	R/Write 1 to Clear	All	0
VERR	Local VERR Interrupt Status/Clear 0=Local VERR Interrupt masked, 1=Local VERR Interrupt active	R/Write 1 to Clear	All	0
LERR	Local LERR Interrupt Status/Clear 0=Local LERR Interrupt masked, 1=Local LERR Interrupt active	R/Write 1 to Clear	All	0
DMA	Local DMA Interrupt Status/Clear 0=Local DMA Interrupt masked, 1=Local DMA Interrupt active	R/Write 1 to Clear	All	0
VIRQ7-VIRQ1	VIRQ*x* Interrupt Status/Clear 0=VIRQ*x* Interrupt masked, 1=VIRQ*x* Interrupt active	R/Write 1 to Clear	All	0
VOWN	VOWN Interrupt Status/Clear 0=no VOWN Interrupt masked, 1=VOWN Interrupt active	R/Write 1 to Clear	All	0

12.3.58 PCI Interrupt Map 0 Register (LINT_MAP0)

This register maps various interrupt sources to one of the eight PCI interrupt pins. For example, a value of 000 maps the corresponding interrupt source to LINT_ [0].

Register name: LINT_MAP0	Register offset: 0x308

Bits	7	6	5	4	3	2	1	0
31:24	Reserved	VIRQ7			Reserved	VIRQ6		
23:16	Reserved	VIRQ5			Reserved	VIRQ4		
15:08	Reserved	VIRQ3			Reserved	VIRQ2		
07:00	Reserved	VIRQ1			Reserved	VOWN		

Name	Description	Type	Reset by	Reset value
VIRQ7-VIRQ1	PCI interrupt destination (LINT[7:0]) for VIRQx	R/W	All	0
VOWN	VMEbus ownership bit interrupt map to PCI interrupt	R/W	All	0

12.3.59 PCI Interrupt Map 1 Register (LINT_MAP1)

This register maps various interrupt sources to one of the eight PCI interrupt pins. For example, a value of 000 maps the corresponding interrupt source to LINT_ [0].

Register name: LINT_MAP1	Register offset: 0x30C

Bits	7	6	5	4	3	2	1	0
31:24	Reserved	ACFAIL			Reserved	SYSFAIL		
23:16	Reserved	SW_INT			Reserved	SW_IACK		
15:08	Reserved					VERR		
07:00	Reserved	LERR			Reserved	DMA		

Name	Description	Type	Reset by	Reset value
ACFAIL	ACFAIL interrupt destination	R/W	All	0
SYSFAIL	SYSFAIL interrupt destination	R/W	All	0
SW_INT	PCI software interrupt destination	R/W	All	0
SW_IACK	VMEbus Software IACK interrupt destination	R/W	All	0
VERR	VMEbus Error interrupt destination	R/W	All	0
LERR	PCI Bus Error interrupt destination	R/W	All	0
DMA	DMA interrupt destination	R/W	All	0

12.3.60 VMEbus Interrupt Enable Register (VINT_EN)

This register enables the various sources of VMEbus interrupts.

SW_INT can be enabled with the VME64AUTO power-up option.

Register name: VINT_EN	Register offset: 0x310

Bits	7	6	5	4	3	2	1	0
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

Name	Description	Type	Reset by	Reset value
SW_INT7	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.	R/W	All	0
SW_INT6	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.	R/W	All	0
SW_INT5	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.	R/W	All	0
SW_INT4	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.	R/W	All	0

Name	Description	Type	Reset by	Reset value
SW_INT3	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.	R/W	All	0
SW_INT2	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.	R/W	All	0
SW_INT1	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.	R/W	All	0
MBOX3	Mailbox 3 Mask 0=MBOX3 Interrupt masked, 1=MBOX3 Interrupt enabled	R/W	All	0
MBOX2	Mailbox 2 Mask 0=MBOX2 Interrupt masked, 1=MBOX2 Interrupt enabled	R/W	All	0
MBOX1	Mailbox 1 Mask 0=MBOX1 Interrupt masked, 1=MBOX1 Interrupt enabled	R/W	All	0
MBOX0	Mailbox 0 Mask 0=MBOX0 Interrupt masked, 1=MBOX0 Interrupt enabled	R/W	All	0
SW_INT	“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.	R/W	All	Power-up Option

Name	Description	Type	Reset by	Reset value
VERR	VERR Interrupt Mask 0 =PCI VERR Interrupt masked 1=PCI VERR Interrupt enabled	R/W	All	0
LERR	LERR Interrupt Mask 0 =PCI LERR Interrupt masked 1 =PCI LERR Interrupt enabled	R/W	All	0
DMA	DMA Interrupt Mask 0=PCI DMA Interrupt masked 1=PCI DMA Interrupt enabled	R/W	All	0
LINT7-LINT0	PCI Interrupt Mask 0=LINT*x* Interrupt masked 1 =LINT*x* Interrupt enabled	R/W	All	0

12.3.61 VMEbus Interrupt Status Register (VINT_STAT)

This register maps PCI Bus 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].

Register name: VINT_STAT	Register offset: 0x314

Bits	7	6	5	4	3	2	1	0
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

Name	Description	Type	Reset by	Reset value
SW_INT7	VME Software 7 Interrupt Status/Clear 0=no VME Software 7 Interrupt, 1=VME Software 7 Interrupt active	R/Write 1 to clear	All	0
SW_INT6	VME Software 6 Interrupt Status/Clear 0=no VME Software 6 Interrupt, 1=VME Software 6 Interrupt active	R/Write 1 to clear	All	0
SW_INT5	VME Software 5 Interrupt Status/Clear 0=no VME Software 5 Interrupt, 1=VME Software 5 Interrupt active	R/Write 1 to clear	All	0
SW_INT4	VME Software 4 Interrupt Status/Clear 0=no VME Software 4 Interrupt, 1=VME Software 4 Interrupt active	R/Write 1 to clear	All	0
SW_INT3	VME Software 3 Interrupt Status/Clear 0=no VME Software 3 Interrupt, 1=VME Software 3 Interrupt active	R/Write 1 to clear	All	0
SW_INT2	VME Software 2 Interrupt Status/Clear 0=no VME Software 2 Interrupt, 1=VME Software 2 Interrupt active	R/Write 1 to clear	All	0
SW_INT1	VME Software 1 Interrupt Status/Clear 0=no VME Software 1 Interrupt, 1=VME Software 1 Interrupt active	R/Write 1 to clear	All	0
MBOX3	Mailbox 3 Status/Clear 0=no Mailbox 3 Interrupt, 1=Mailbox 3 Interrupt active	R/Write 1 to clear	All	0

Name	Description	Type	Reset by	Reset value
MBOX2	Mailbox 2 Status/Clear 0=no Mailbox 2 Interrupt, 1=Mailbox 2 Interrupt active	R/Write 1 to clear	All	0
MBOX1	Mailbox 1 Status/Clear 0=no Mailbox 1 Interrupt, 1=Mailbox 1 Interrupt active	R/Write 1 to clear	All	0
MBOX0	Mailbox 0 Status/Clear 0=no Mailbox 0 Interrupt, 1=Mailbox 0 Interrupt active	R/Write 1 to clear	All	0
SW_INT	VME Software Interrupt Status/Clear 0=VME Software Interrupt inactive, 1=VME Software Interrupt active	R/Write 1 to Clear	All	Power-up Option
VERR	VERR Interrupt Status/Clear 0=VME VERR Interrupt masked, 1=VME VERR Interrupt enabled	R/Write 1 to Clear	All	0
LERR	LERR Interrupt Status/Clear 0=VME LERR Interrupt masked, 1=VME LERR Interrupt enabled	R/Write 1 to Clear	All	0
DMA	DMA Interrupt Status/Clear 0=VME DMA Interrupt masked, 1=VME DMA Interrupt enabled	R/Write 1 to Clear	All	0
LINT7-LINT0	LINT*x* Interrupt Status/Clear 0=LINT*x* Interrupt masked, 1=LINT*x* Interrupt enabled	R/Write 1 to Clear	All	0

12.3.62 VME Interrupt Map 0 Register (VINT_MAP0)

Register name: VINT_MAP0	Register offset: 0x318

Bits	7	6	5	4	3	2	1	0
31:24	Reserved	LINT7			Reserved	LINT6		
23:16	Reserved	LINT5			Reserved	LINT4		
15:08	Reserved	LINT3			Reserved	LINT2		
07:00	Reserved	LINT1			Reserved	LINT0		

Name	Description	Type	Reset by	Reset value
LINT7-LINT0	VMEbus destination of PCI Bus interrupt source	R/W	all	0

12.3.63 VME Interrupt Map 1 Register (VINT_MAP1)

This register maps VMEbus 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 010 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.

Register name: VINT_MAP1	**Register offset: 0x31C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	Reserved							
23:16	Reserved					SW_INT		
15:08	Reserved					VERR		
07:00	Reserved	LERR			Reserved	DMA		

Name	**Description**	**Type**	**Reset by**	**Reset value**
SW_INT	VMEbus Software interrupt destination	R/W	all	Power-up Option
VERR	VMEbus Error interrupt destination	R/W	all	0
LERR	PCI Bus Error interrupt destination	R/W	all	0
DMA	DMA interrupt destination	R/W	all	0

12.3.64 Interrupt STATUS/ID Out Register (STATID)

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.

Register name: STATID	**Register offset: 0x320**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	STATID [7:0]							
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
STATID [7:1]	Bits [7:1] of the STATUS/ID byte are returned when the Universe II responds to a VMEbus IACK cycle.	R/W	all	1111111
STATID [0]	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.	R	all	undefined

The reset state is designed to support the VME64 Auto ID STATUS/ID value.

12.3.65 VIRQ1 STATUS/ID Register (V1_STATID)

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.

Register name: V1_STATID	Register offset: 0x324

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 1 VMEbus interrupt	R	all	0

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.

12.3.66 VIRQ2 STATUS/ID Register (V2_STATID)

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.

Register name: V2_STATID	Register offset: 0x328

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 1 VMEbus interrupt	R	all	0

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.

12.3.67 VIRQ3 STATUS/ID Register (V3_STATID)

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.

Register name: V3_STATID	Register offset: 0x32C

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 3VMEbus interrupt	R	all	0

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.

12.3.68 VIRQ4 STATUS/ID Register (V4_STATID)

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.

Register name: V4_STATID	Register offset: 0x330

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 4 VMEbus interrupt	R	all	0

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.

12.3.69 VIRQ5 STATUS/ID Register (V5_STATID)

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.

Register name: V5_STATID	Register offset: 0x334

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 5 VMEbus interrupt	R	all	0

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.

12.3.70 VIRQ6 STATUS/ID Register (V6_STATID)

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.

Register name: V6_STATID	Register offset: 0x338

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 6 VMEbus interrupt	R	all	0

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.

12.3.71 VIRQ7 STATUS/ID Register (V7_STATID)

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.

Register name: V7_STATID	Register offset: 0x33C

Bits	7	6	5	4	3	2	1	0
31:24	Reserved							
23:16	Reserved							
15:08	Reserved							ERR
07:00	STATID [7:0]							

Name	Description	Type	Reset by	Reset value
ERR	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID	R	all	0
STATID [7:0]	STATUS/ID acquired during IACK cycle for level 7 VMEbus interrupt	R	all	0

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.

12.3.72 PCI Interrupt Map 2 Register (LINT_MAP2)

This register maps interrupt sources to one of the eight PCI interrupt pins. For example, a value of 000 maps the corresponding interrupt source to LINT_ [0].

Register name: LINT_MAP2	Register offset: 0x340

Bits	7	6	5	4	3	2	1	0
31:24	Reserved	LM3			Reserved	LM2		
23:16	Reserved	LM1			Reserved	LM0		
15:08	Reserved	MBOX3			Reserved	MBOX2		
07:00	Reserved	MBOX1			Reserved	MBOX0		

Name	Description	Type	Reset by	Reset value
LM3 [2:0]	Location Monitor 3 Interrupt destination	R/W	all	0
LM2 [2:0]	Location Monitor 2 Interrupt destination	R/W	all	0
LM1 [2:0]	Location Monitor 1 Interrupt destination	R/W	all	0
LM0 [2:0]	Location Monitor 0 Interrupt destination	R/W	all	0
MBOX3 [2:0]	Mailbox 3 Interrupt destination	R/W	all	0
MBOX2 [2:0]	Mailbox 2 Interrupt destination	R/W	all	0
MBOX1 [2:0]	Mailbox 1 Interrupt destination	R/W	all	0
MBOX0 [2:0]	Mailbox 0 Interrupt destination	R/W	all	0

12.3.73 VME Interrupt Map 2 Register (VINT_MAP2)

This register maps 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].

Register name: VINT_MAP2	**Register offset: 0x344**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	Reserved							
23:16	Reserved							
15:08	Reserved	MBOX3			Reserved	MBOX2		
07:00	Reserved	MBOX1			Reserved	MBOX0		

Name	**Description**	**Type**	**Reset by**	**Reset value**
MBOX3 [2:0]	Mailbox 3 Interrupt destination	R/W	all	0
MBOX2 [2:0]	Mailbox 2 Interrupt destination	R/W	all	0
MBOX1 [2:0]	Mailbox 1 Interrupt destination	R/W	all	0
MBOX0 [2:0]	Mailbox 0 Interrupt destination	R/W	all	0

12.3.74 Mailbox 0 Register (MBOX0)

This register is a general purpose mailbox register.

Register name: MBOX0	Register offset: 0x348

Bits	Function
31:24	MBOX0
23:16	MBOX0
15:08	MBOX0
07:00	MBOX0

Name	Description	Type	Reset by	Reset value
MBOX0 [31:0]	Mailbox	R/W	all	0

12.3.75 Mailbox 1 Register (MBOX1)

This register is a general purpose mailbox register.

Register name: MBOX1	Register offset: 0x34C

Bits	Function
31:24	MBOX1
23:16	MBOX1
15:08	MBOX1
07:00	MBOX1

Name	Description	Type	Reset by	Reset value
MBOX1 [31:0]	Mailbox	R/W	all	0

12.3.76 Mailbox 2 Register (MBOX2)

This register is a general purpose mailbox register.

Register name: MBOX2	Register offset: 0x350

Bits	Function
31:24	MBOX2
23:16	MBOX2
15:08	MBOX2
07:00	MBOX2

Name	Description	Type	Reset by	Reset value
MBOX2 [31:0]	Mailbox	R/W	all	0

12.3.77 Mailbox 3 Register (MBOX3)

This register is a general purpose mailbox register.

Register name: MBOX3	Register offset: 0x354

Bits	Function
31:24	MBOX3
23:16	MBOX3
15:08	MBOX3
07:00	MBOX3

Name	Description	Type	Reset by	Reset value
MBOX3 [31:0]	Mailbox	R/W	all	0

12.3.78 Semaphore 0 Register (SEMA0)

This register can only be accessed through byte-wide access.

Register name: SEMA0	Register offset: 0x358

Bits	7	6	5	4	3	2	1	0
31:24	SEM3	TAG3						
23:16	SEM2	TAG2						
15:08	SEM1	TAG1						
07:00	SEM0	TAG0						

Name	Description	Type	Reset by	Reset value
SEM3	Semaphore 3	R/W	all	0
TAG3 [6:0]	Tag 3	R/W	all	0
SEM2	Semaphore 2	R/W	all	0
TAG2 [6:0]	Tag2	R/W	all	0
SEM1	Semaphore 1	R/W	all	0
TAG1 [6:0]	Tag 1	R/W	all	0
SEM0	Semaphore 0	R/W	all	0
TAG0 [6:0]	Tag 0	R/W	all	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 (see “Semaphores” on page 83).

12.3.79 Semaphore 1 Register (SEMA1)

This register can only be accessed through byte-wide access.

Register name: SEMA1	Register offset: 0x35C

Bits	7	6	5	4	3	2	1	0
31:24	SEM7	TAG7						
23:16	SEM6	TAG6						
15:08	SEM5	TAG5						
07:00	SEM4	TAG4						

Name	Description	Type	Reset by	Reset value
SEM3	Semaphore 7	R/W	all	0
TAG3 [6:0]	Tag 7	R/W	all	0
SEM2	Semaphore 6	R/W	all	0
TAG2 [6:0]	Tag 6	R/W	all	0
SEM1	Semaphore 5	R/W	all	0
TAG1 [6:0]	Tag 5	R/W	all	0
SEM0	Semaphore 4	R/W	all	0
TAG0 [6:0]	Tag 4	R/W	all	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 (see “Semaphores” on page 83).

12.3.80 Master Control Register (MAST_CTL)

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 de-queuing 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.

Register name: MAST_CTL	**Register offset: 0x400**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	MAXRTRY				PWON			
23:16	VRL		VRM	VREL	VOWN	VOWN_ ACK	Reserved	
15:08	Reserved		PABS		Reserved			
07:00	BUS_NO							

Name	**Description**	**Type**	**Reset by**	**Reset value**
MAXRTRY [3:0]	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	R/W	all	1000
PWON [3:0]	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.	R/W	all	0000

Name	Description	Type	Reset by	Reset value
VRL [1:0]	VMEbus Request Level 00=Level 0,01=Level 1,10=Level 2, 11=Level 3	R/W	all	11
VRM	VMEbus Request Mode 0=Demand,1=Fair	R/W	all	0
VREL	VMEbus Release Mode 0=Release When Done (RWD), 1=Release on Request (ROR)	R/W	all	0
VOWN	VME Ownership Bit 0=Release VMEbus, 1=Acquire and Hold VMEbus	R/W	all	0
VOWN_ACK	VME Ownership Bit Acknowledge 0=VMEbus not owned, 1=VMEbus acquired and held due to assertion of VOWN	R	all	0
PABS [1:0]	PCI Aligned Burst Size 00=32 byte 01=64 byte 10=128 byte 11=256 byte Controls the PCI address boundary at which the Universe II breaks up a PCI transaction in the VME Slave channel (see “VME Slave Image Programming” on page 67) and the DMA Channel (see “FIFO Operation and Bus Ownership” on page 101). 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, 64,128, or 256 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 101).	R/W	all	00
BUS_NO [7:0]	PCI Bus Number	R/W	all	0000 0000

12.3.81 Miscellaneous Control Register (MISC_CTL)

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.

Register name: MISC_CTL	**Register offset: 0x404**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	VBTO				Reserved	VARB	VARBTO	
23:16	SW_ LRST	SW_ SYSRST	Reserved	BI	ENGBI	RE-SCIND	SYSCON	V64-AUTO
15:08	Reserved	SYSRE-SET	Reserved					
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
VBTO	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 Note: The VBTO value has a 4us resolution. The time-out may occur up to 4us earlier than the selected VBTO period.	R/W	all	0011
VARB	VMEbus Arbitration Mode 0=Round Robin 1=Priority	R/W	all	0

Name	Description	Type	Reset by	Reset value
VARBTO	VMEbus Arbitration Time-out 00=Disable Timer 01=16 µs (minimum 8µs) 10=256 µs others= Reserved	R/W	all	01
SW_LRST	Software PCI Reset 0=No effect 1=Initiate LRST_ A read always returns 0.	R/W	all	0
SW_ SYSRST	Software VMEbus SYSRESET 0=No effect 1=Initiate SYSRST* A read always returns 0.	R/W	all	0
BI	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.	R/W	all	Power-up Option
ENGBI	Enable Global BI-Mode Initiator 0=Assertion of VIRQ1 ignored 1=Assertion of VIRQ1 puts device in BI-Mode	R/W	all	0
RESCIND	RESCIND is unused in the Universe II.	R/W	all	1
SYSCON	SYSCON 0=Universe II is not VMEbus System Controller, 1=Universe II is VMEbus System Controller	R/W	all	Power-up Option
V64AUTO	VME64 Auto ID Write: 0=No effect 1=Initiate sequence This bit initiates Universe II VME64 Auto ID Slave participation.	R/W	all	Power-up Option
SYSRESET	System Reset 0=No effect 1=Initiate SYSRST* Universe II asserts SYSRESET without resetting itself.	R/W	all	0

12.3.82 Miscellaneous Status Register (MISC_STAT)

Register name: MISC_STAT	Register offset: 0x408

Bits	7	6	5	4	3	2	1	0
31:24	Reserved	LCLSIZE	Reserved		DY4AUTO	Reserved		
23:16	Reserved		MYBBSY	Reserved	DY4_DONE	TXFE	RXFE	Reserved
15:08	DY4AUTOID							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
LCLSIZE	PCI Bus Size At the rising 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	R	all	Power-up Option
DY4AUTO	DY4 Auto ID Enable 0=Disable 1=Enable	R	all	Power-up Option
MYBBSY	Universe II BBSY 0=Asserted 1=Negated	R	all	1
DY4DONE	DY4 Auto ID Done 0=Not done 1=Done	R	all	0
TXFE	PCI Target Channel Posted Writes FIFO 0=data in the FIFO 1=no data in the FIFO	R	all	1
RXFE	VME Slave Channel Posted Writes FIFO 0=data in the FIFO 1=no data in the FIFO	R	all	1
DY4 AUTOID	DY4 Auto ID	R	Power-up reset and VMEbus SYS-RESET*	0

12.3.83 User AM Codes Register (USER_AM)

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*.

Register name: USER_AM	Register offset: 0x40C

Bits	7	6	5	4	3	2	1	0
31:24	0	1	USER1AM				Reserved	
23:16	0	1	USER2AM				Reserved	
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
USER1AM [3:0]	User AM Code 1	R/W	all	0000
USER2AM [3:0]	User AM Code 2	R/W	all	0000

12.3.84 Universe II Specific Register (U2SPEC)

Register name: U2SPEC	Register offset: 0x4FC

Bits	7	6	5	4	3	2	1	0
31:24	Universe Reserved							
23:16	Universe Reserved							
15:08	Universe Reserved	DS0/DS1	AS	DTKFLTR	Reserved	MASt11	READt27	
07:00	Universe Reserved					POSt28	Reserved	PREt28

Name	Description	Type	Reset by	Reset value
DS0/DS1	Data Strobe Filtering This filter eliminates the noise on DS1* and DS0*. 0=Disable 1=Enable	R/W	all	0
AS	Address Strobe Filtering This filter eliminates the noise on AS*. 0=Disable 1=Enable	R/W	all	0
DTKFLTR	VME DTACK* Inactive Filter 0=Slower but better filter 1=Faster but poorer filter	R/W	all	0
MASt11	VME Master Parameter t11 Control (DS* high time during BLT's and MBLT's) 0=Default 1=Faster	R/W	all	0
READt27	VME Master Parameter t27 Control (Delay of DS* negation after read) 00=Default 01=Faster 10=No Delay	R/W	all	00
POSt28	VME Slave Parameter t28 Control (Time of DS* to DTACK* for posted-write) 0=Default 1=Faster	R/W	all	0

Name	Description	Type	Reset by	Reset value
PREt28	VME Slave Parameter t28 Control (Time of DS* to DTACK* for prefetch read) 0=Default 1=Faster	R/W	all	0

12.3.85 VMEbus Slave Image 0 Control Register (VSI0_CTL)

The state of PWEN and PREN are ignored if LAS is not programmed memory space.

Register name: VSI0_CTL	Register offset: 0xF00

Bits	7	6	5	4	3	2	1	0
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	Description	Type	Reset by	Reset value
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR VME	0

Name	Description	Type	Reset by	Reset value
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.86 VMEbus Slave Image 0 Base Address Register (VSI0_BS)

The base address specifies the lowest address in the address range that is decoded. This image has 4 Kbyte resolution.

Register name: VSI0_BS	Register offset: 0xF04

Bits	Function	
31:24	BS	
23:16	BS	
15:08	BS	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
BS[31:12]	Base Address	R/W	PWR VME	0

12.3.87 VMEbus Slave Image 0 Bound Address Register (VSI0_BD)

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.

Register name: VSI0_BD	Register offset: 0xF08

Bits	Function	
31:24	BD	
23:16	BD	
15:08	BD	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
BD[31:12]	Bound Address	R/W	PWR VME	0

12.3.88 VMEbus Slave Image 0 Translation Offset (VSI0_TO)

This image has 4 Kbyte resolution.

Register name: VSI0_TO	Register offset: 0xF0C

Bits	Function	
31:24	TO	
23:16	TO	
15:08	TO	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
TO[31:12]	Translation Offset	R/W	PWR VME	0

12.3.89 VMEbus Slave Image 1 Control (VSI1_CTL)

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.

Register name: VSI1_CTL	**Register offset: 0xF14**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=Reserved 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	1
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	1
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.90 VMEbus Slave Image 1 Base Address Register (VSI1_BS)

The base address specifies the lowest address in the address range that will be decoded.

Register name: VSI1_BS	Register offset: 0xF18

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS[31:16]	Base Address	R/W	PWR VME	0

12.3.91 VMEbus Slave Image 1 Bound Address Register (VSI1_BD)

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.

Register name: VSI1_BD	Register offset: 0xF1C

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD[31:16]	Bound Address	R/W	PWR VME	0

12.3.92 VMEbus Slave Image 1 Translation Offset (VSI1_TO)

Register name: VSI1_TO	Register offset: 0xF20

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO[31:16]	Translation Offset	R/W	PWR VME	0

12.3.93 VMEbus Slave Image 2 Control (VSI2_CTL)

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.

Register name: VSI2_CTL	**Register offset: 0xF28**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=Reserved 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.94 VMEbus Slave Image 2 Base Address Register (VSI2_BS)

Register name: VSI2_BS	Register offset: 0xF2C

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS[31:16]	Base Address	R/W	PWR VME	0

12.3.95 VMEbus Slave Image 2 Bound Address Register (VSI2_BD)

Register name: VSI2_BD	Register offset: 0xF30

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD[31:16]	Bound Address	R/W	PWR VME	0

12.3.96 VMEbus Slave Image 2 Translation Offset (VSI2_TO)

Register name: VSI2_TO	Register offset: 0xF34

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO[31:16]	Translation Offset	R/W	PWR VME	0

12.3.97 VMEbus Slave Image 3 Control (VSI3_CTL)

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.

Register name: VSI3_CTL	**Register offset: 0xF3C**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=Reserved 001=A24 010=A32 011= Reserved 100=Reserved, 101=Reserved 110=User1 111=User2	R/W	PWR VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.98 VMEbus Slave Image 3 Base Address Register (VSI3_BS)

Register name: VSI3_BS	Register offset: 0xF40

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS[31:16]	Base Address	R/W	PWR VME	0

12.3.99 VMEbus Slave Image 3 Bound Address Register (VSI3_BD)

Register name: VSI3_BD	Register offset: 0xF44

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD[31:16]	Bound Address	R/W	PWR VME	0

12.3.100 VMEbus Slave Image 3 Translation Offset (VSI3_TO)

Register name: VSI3_TO	Register offset: 0xF48

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO[31:16]	Translation Offset	R/W	PWR VME	0

12.3.101 Location Monitor Control Register (LM_CTL)

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.

Register name: LM_CTL	**Register offset: 0xF64**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	Reserved						
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011=Reserved 100=Reserved 101=Reserved 110=User1 111=User2 others= Reserved	R/W	PWR VME	0

12.3.102 Location Monitor Base Address Register (LM_BS)

The base address specifies the lowest address in the 4 Kbyte range that will be decoded as a location monitor access.

Register name: LM_BS	Register offset: 0xF68

Bits	7	6	5	4	3	2	1	0
31:24	BS							
23:16	BS							
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
BS [31:12]	Base Address	R/W	PWR VME	0

12.3.103 VMEbus Register Access Image Control Register (VRAI_CTL)

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.

Register name: VRAI_CTL	**Register offset: 0xF70**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	Reserved						
23:16	PGM		SUPER		Reserved		VAS	
15:08	Reserved							
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	Power-up Option
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11
VAS	VMEbus Address Space 00=A16 01=A24 10=A32 all others are reserved	R/W	PWR VME	Power-up Option

12.3.104 VMEbus Register Access Image Base Address Register (VRAI_BS)

The base address specifies the lowest address in the 4 Kbyte VMEbus Register Access Image.

Register name: VRAI_BS	Register offset: 0xF74

Bits	Function	
31:24	BS	
23:16	BS	
15:08	BS	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
BS[31:12]	The base address specifies the lowest address in the 4 Kbyte VMEbus Register Access Image.	R/W	PWR VME	Power-up Option

The reset state is a function of the Power-up Option behavior of the VAS field in VRAI_CTL. Table 35 shows the behavior of the VAS field.

Table 35: Power-up Option Behavior of the VAS field in VRAI_CTL

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

12.3.105 VMEbus CSR Control Register (VCSR_CTL)

Register name: VCSR_CTL	Register offset: 0xF80

Bits	7	6	5	4	3	2	1	0
31:24	EN	Reserved						
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							LAS

Name	Description	Type	Reset by	Reset value
EN	Image Enable The EN bit is set to a value of 1 whenever a VME64 monarch acquires the Status/ID vector for the level 2 interrupt during VME64 Auto ID. 0=Disable 1=Enable	R/W	PWR,VME	0
LAS	PCI Bus Address Space 00= PCI Bus Address Space 01= PCI Bus I/O Space	R/W	PWR,VME	0

12.3.106 VMEbus CSR Translation Offset (VCSR_TO)

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.

Register name: VCSR_TO	Register offset: 0xF84

Bits	7	6	5	4	3	2	1	0
31:24	TO							
23:16	TO					Reserved		
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
TO [31:24]	Translation Offset	R/W	PWR VME	0
TO [23:19]	Translation Offset	R/W	PWR VME	Power-up Option

12.3.107 VMEbus AM Code Error Log (V_AMERR)

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.

Register name: V_AMERR	Register offset: 0xF88

Bits	7	6	5	4	3	2	1	0
31:24	AMERR						IACK	M_ERR
23:16	V_STAT	Reserved						
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
AMERR [5:0]	VMEbus AM Code Error Log	R	PWR, VME	0
IACK	VMEbus IACK Signal	R	PWR, VME	0
M_ERR	Multiple Error Occurred 0=Single error 1=At least one error has occurred since the logs were frozen	R	PWR, VME	0
V_STAT	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	R/W	PWR, VME	0

12.3.108 VMEbus Address Error Log (VAERR)

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. The register contents are qualified by the V_STAT bit of the V_AMERR register.

Register name: VAERR	Register offset: 0xF8C

Bits	7	6	5	4	3	2	1	0
31:24	VAERR							
23:16	VAERR							
15:08	VAERR							
07:00	VAERR							

Name	Description	Type	Reset by	Reset value
VAERR [31:1]	VMEbus address error log	R	PWR, VME	0

12.3.109 VMEbus Slave Image 4 Control (VSI4_CTL)

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.

Register name: VSI4_CTL	**Register offset: 0xF90**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR, VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR, VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR, VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR, VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR, VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=A16 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR, VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR, VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR, VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR, VME	0

12.3.110 VMEbus Slave Image 4 Base Address Register (VSI4_BS)

The base address specifies the lowest address in the address range that is decoded.

This image has a 4 Kbyte resolution.

Register name: VSI4_BS	Register offset: 0xF94

Bits	Function	
31:24	BS	
23:16	BS	
15:08	BS	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
BS[31:12]	Base Address	R/W	PWR VME	0

12.3.111 VMEbus Slave Image 4 Bound Address Register (VSI4_BD)

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.

Register name: VSI4_BD	**Register offset: 0xF98**

Bits	Function	
31:24	BD	
23:16	BD	
15:08	BD	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
BD[31:12]	Bound Address	R/W	PWR VME	0

12.3.112 VMEbus Slave Image 4 Translation Offset (VSI4_TO)

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.

Register name: VSI4_TO	Register offset: 0xF9C

Bits	Function	
31:24	TO	
23:16	TO	
15:08	TO	Reserved
07:00	Reserved	

Name	Description	Type	Reset by	Reset value
TO[31:12]	Translation Offset	R/W	PWR VME	0

12.3.113 VMEbus Slave Image 5 Control (VSI5_CTL)

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.

Register name: VSI5_CTL	**Register offset: 0xFA4**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=Reserved 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.114 VMEbus Slave Image 5 Base Address Register (VSI5_BS)

The base address specifies the lowest address in the address range that is decoded.

Register name: VSI5_BS	Register offset: 0xFA8

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS[31:16]	Base Address	R/W	PWR VME	0

12.3.115 VMEbus Slave Image 5 Bound Address Register (VSI5_BD)

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.

Register name: VSI5_BD	Register offset: 0xFAC

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD[31:16]	Bound Address	R/W	PWR VME	0

12.3.116 VMEbus Slave Image 5 Translation Offset (VSI5_TO)

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.

Register name: VSI5_TO	Register offset: 0xFB0

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO[31:16]	Translation Offset	R/W	PWR VME	0

12.3.117 VMEbus Slave Image 6 Control (VSI6_CTL)

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.

Register name: VSI6_CTL	**Register offset: 0xFB8**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=Reserved 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.118 VMEbus Slave Image 6 Base Address Register (VSI6_BS)

The base address specifies the lowest address in the address range that is decoded.

Register name: VSI6_BS	Register offset: 0xFBC

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS[31:16]	Base Address	R/W	PWR VME	0

12.3.119 VMEbus Slave Image 6 Bound Address Register (VSI6_BD)

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.

Register name: VSI6_BD	Register offset: 0xFC0

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD[31:16]	Bound Address	R/W	PWR VME	0

12.3.120 VMEbus Slave Image 6 Translation Offset (VSI6_TO)

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.

Register name: VSI6_TO	Register offset: 0xFC4

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO[31:16]	Translation Offset	R/W	PWR VME	0

12.3.121 VMEbus Slave Image 7 Control (VSI7_CTL)

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.

Register name: VSI7_CTL	**Register offset: 0xFCC**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	EN	PWEN	PREN	Reserved				
23:16	PGM		SUPER		Reserved	VAS		
15:08	Reserved							
07:00	LD64EN	LLRMW	Reserved				LAS	

Name	**Description**	**Type**	**Reset by**	**Reset value**
EN	Image Enable 0=Disable 1=Enable	R/W	PWR VME	0
PWEN	Posted Write Enable 0=Disable 1=Enable	R/W	PWR VME	0
PREN	Prefetch Read Enable 0=Disable 1=Enable	R/W	PWR VME	0
PGM	Program/Data AM Code 00=Reserved 01=Data 10=Program 11=Both	R/W	PWR VME	11
SUPER	Supervisor/User AM Code 00=Reserved 01=Non-Privileged 10=Supervisor 11=Both	R/W	PWR VME	11

Name	Description	Type	Reset by	Reset value
VAS	VMEbus Address Space 000=Reserved 001=A24 010=A32 011= Reserved 100=Reserved 101=Reserved 110=User1 111=User2	R/W	PWR VME	0
LD64EN	Enable 64-bit PCI Bus Transactions 0=Disable 1=Enable	R/W	PWR VME	0
LLRMW	Enable PCI Bus Lock of VMEbus RMW 0=Disable 1=Enable	R/W	PWR VME	0
LAS	PCI Bus Address Space 00=PCI Bus Memory Space 01=PCI Bus I/O Space 10=PCI Bus Configuration Space 11=Reserved	R/W	PWR VME	0

12.3.122 VMEbus Slave Image 7 Base Address Register (VSI7_BS)

The base address specifies the lowest address in the address range that is decoded.

Register name: VSI7_BS	Register offset: 0xFD0

Bits	Function
31:24	BS
23:16	BS
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BS[31:16]	Base Address	R/W	PWR VME	0

12.3.123 VMEbus Slave Image 7 Bound Address Register (VSI7_BD)

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.

Register name: VSI7_BD	Register offset: 0xFD4

Bits	Function
31:24	BD
23:16	BD
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
BD[31:16]	Bound Address	R/W	PWR VME	0

12.3.124 VMEbus Slave Image 7 Translation Offset (VSI7_TO)

Register name: VSI7_TO	Register offset: 0xFD8

Bits	Function
31:24	TO
23:16	TO
15:08	Reserved
07:00	Reserved

Name	Description	Type	Reset by	Reset value
TO[31:16]	Translation Offset	R/W	PWR VME	0

12.3.125 VMEbus CSR Bit Clear Register (VCSR_CLR)

This register implements the Bit Clear Register as defined in the *VME64 Specification*. The RESET bit must be written to only from the VMEbus.

Register name: VCSR_CLR	Register offset: 0xFF4

Bits	7	6	5	4	3	2	1	0
31:24	RESET	SYSFAIL	FAIL	Reserved				
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
RESET	Board Reset Reads: 0=LRST_ not asserted 1=LRST_ asserted Writes: 0=no effect 1=negate LRST_	R/W	PWR VME	0
SYSFAIL	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted 1=VXSYSFAIL asserted Writes: 0=no effect 1=negate VXSYSFAIL	R/W	all	Power-up Option
FAIL	Board Fail 0=Board has not failed	R	PWR VME	0

12.3.126 VMEbus CSR Bit Set Register (VCSR_SET)

This register implements the Bit Set Register as defined in the *VME64 Specification*. The RESET bit must 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.

Register name: VCSR_SET	**Register offset: 0xFF8**

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
31:24	RESET	SYSFAIL	FAIL	Reserved				
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							

Name	**Description**	**Type**	**Reset by**	**Reset value**
RESET	Board Reset Reads: 0=LRST_ not asserted 1=LRST_ asserted Writes: 0=no effect 1=assert LRST_	R/W	PWR VME	0
SYSFAIL	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted 1=VXSYSFAIL asserted Writes: 0=no effect 1=assert VXSYSFAIL	R/W	all	Power-up Option
FAIL	Board Fail 0=Board has not failed	R	PWR VME	0

12.3.127 VMEbus CSR Base Address Register (VCSR_BS)

The base address specifies one of 31 available CR/CSR windows as defined in the *VME64 Specification*. Each window consumes 512 Kbytes of CR/CSR space.

Register name: VCSR_BS	Register offset: 0xFFC

Bits	7	6	5	4	3	2	1	0
31:24	BS					Reserved		
23:16	Reserved							
15:08	Reserved							
07:00	Reserved							

Name	Description	Type	Reset by	Reset value
BS [23:19]	Base Address	R/W	PWR VME	0

VCSR_BS register is accessed with an 8-bit transfer.

Bits [31:27] of the register are compared with address lines [23:19].

A. Packaging Information

This appendix discusses the following topics:

A.1 313 Pin PBGA Package

Figure 25: 313 PBGA - Bottom View

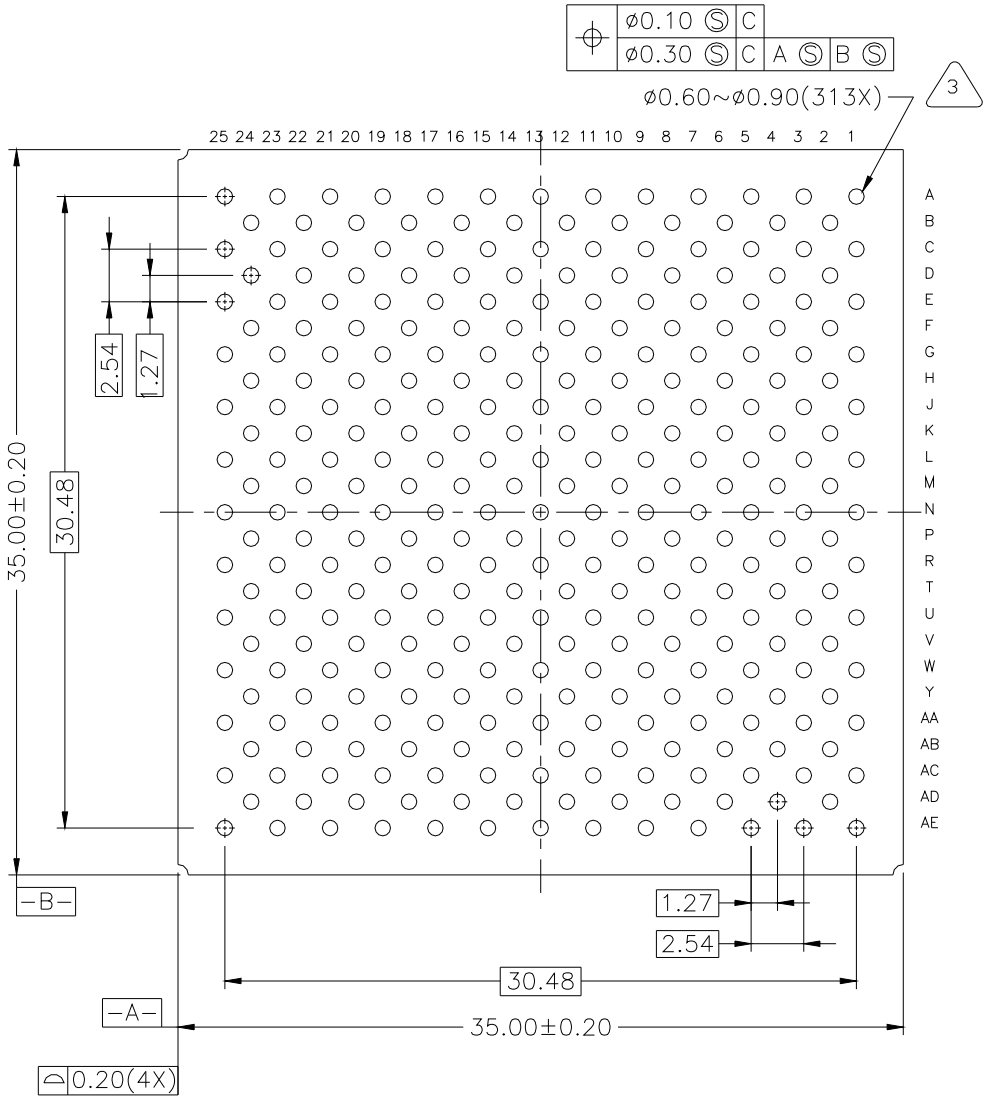
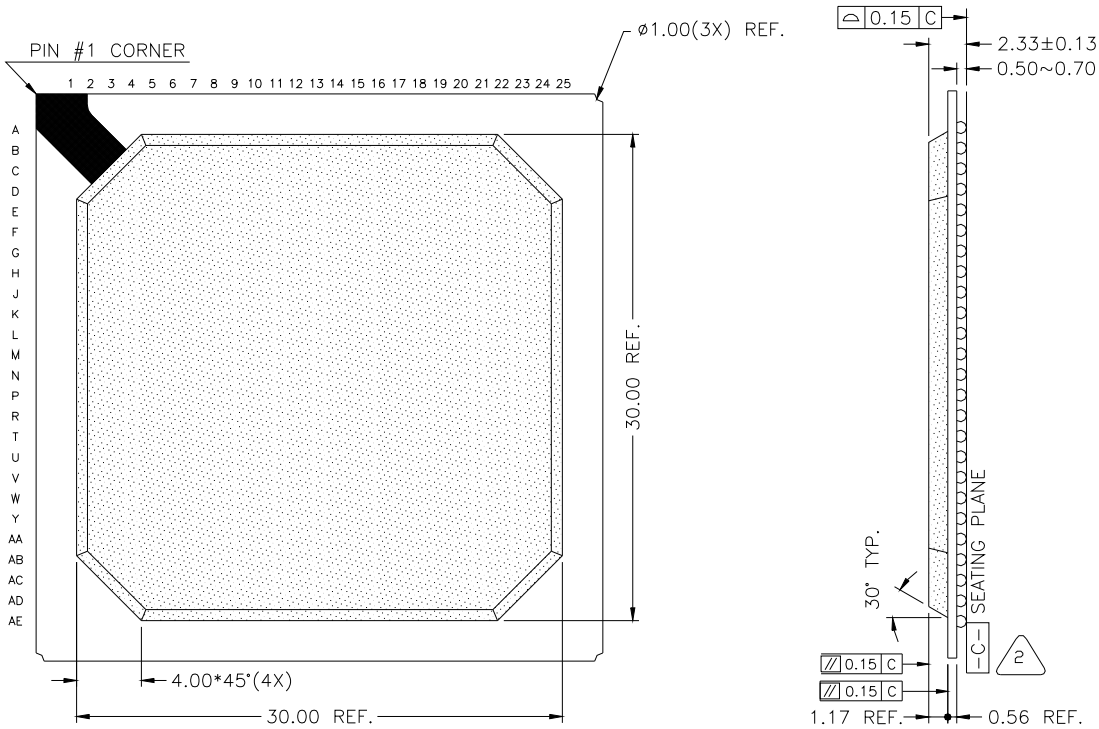


Figure 26: 313 PBGA - Top and Side View



B. Performance

This chapter discusses the following topics:

B.1 Overview

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:

Where relevant, descriptions of factors affecting performance and how they might be controlled in different environments are discussed.

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 original *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.

B.2 PCI Slave Channel

This channel supports both coupled and decoupled transactions. Each type of transaction, and the performance of each, are discussed in the following sections.

B.2.1 Coupled Cycles

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.

B.2.1.1 Request of VMEbus

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 27). The request for the VMEbus occurs shortly after the cycle is retried.

B.2.1.2 Read Cycles

Once the coupled channel owns the VMEbus, the Universe II propagates the cycle out to the VMEbus. Figure 27 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 28 shows several of these accesses occurring consecutively.

Figure 27: Coupled Read Cycle - Universe II as VME Master

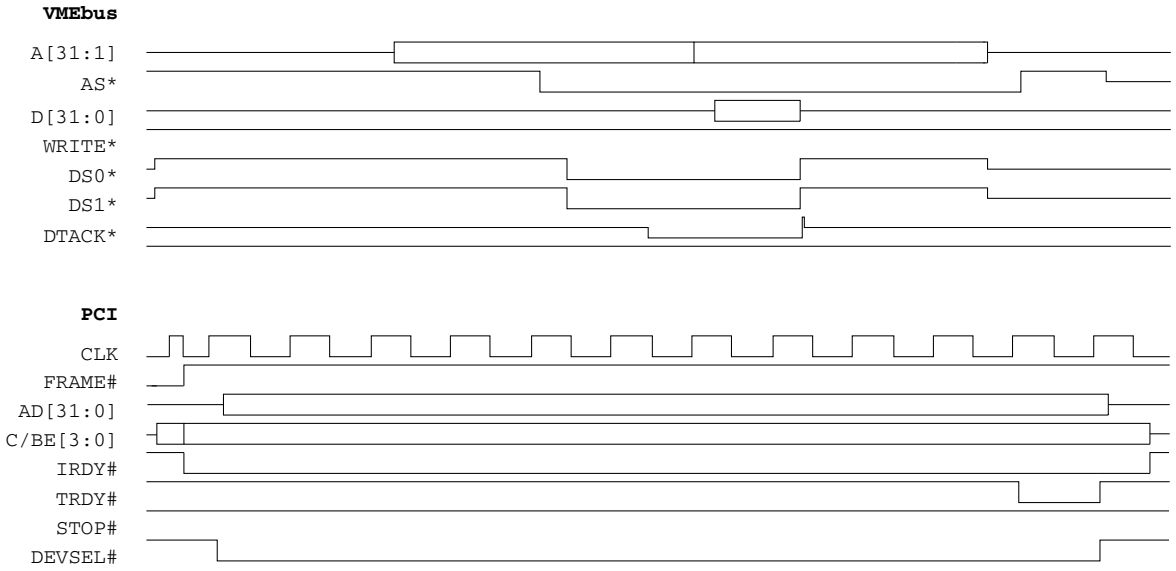
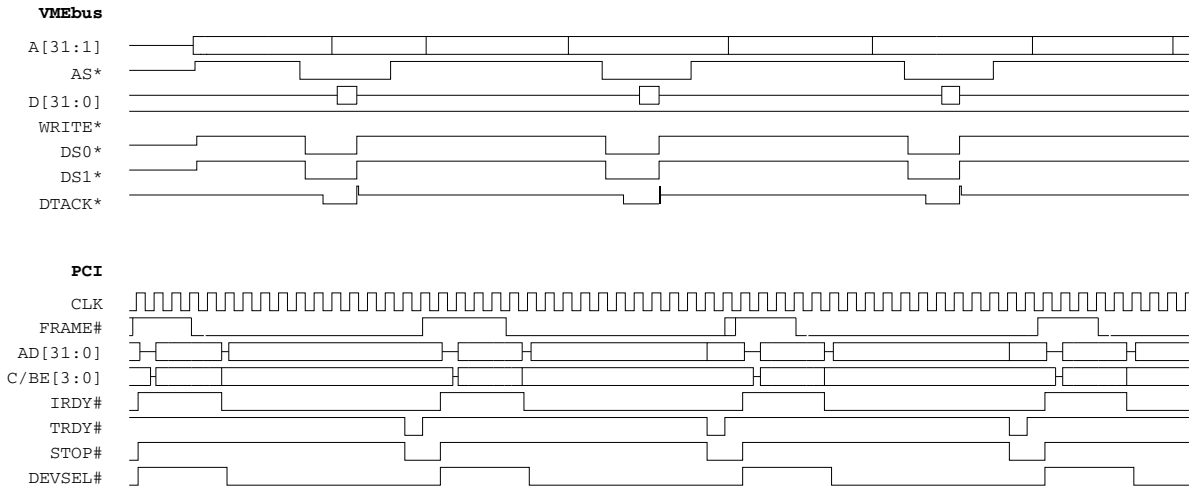


Figure 28: Several Coupled Read Cycles - Universe II as VME Master

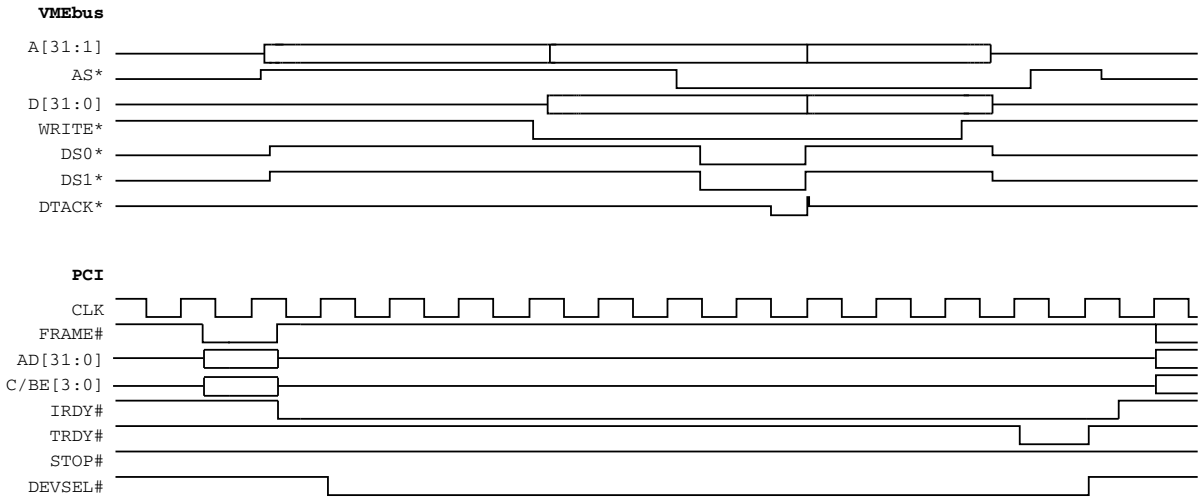


B.2.1.3 Write Cycles

The performance of coupled write cycles is similar to that of coupled read cycles except that an extra wait state is inserted. Figure 29 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 29: Coupled Write Cycle - Universe II as VME Master



B.2.2 Decoupled Cycles

Only write transactions can be decoupled in the PCI Target Channel.

B.2.2.1 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.

B.2.2.2 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.

B.2.2.3 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 -338), 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 30 shows the Universe II queueing 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 30: Several Non-Block Decoupled Writes - Universe II as VME Master

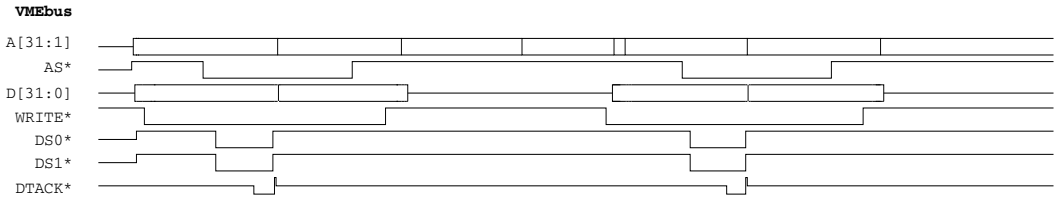
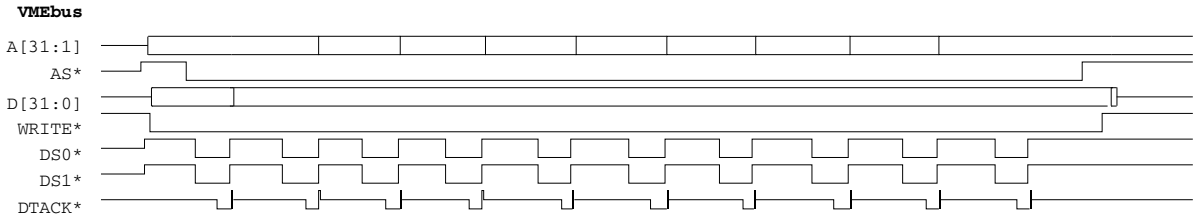


Figure 31: BLT Decoupled Write - Universe II as VME Master



B.3 VME Slave Channel

This channel supports both coupled and decoupled transactions. Each type of transaction, and the performance of each, are discussed in the following sections.

B.3.1 Coupled Cycles

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.

B.3.1.1 Block vs. non-Block Transfers

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.

B.3.1.2 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.

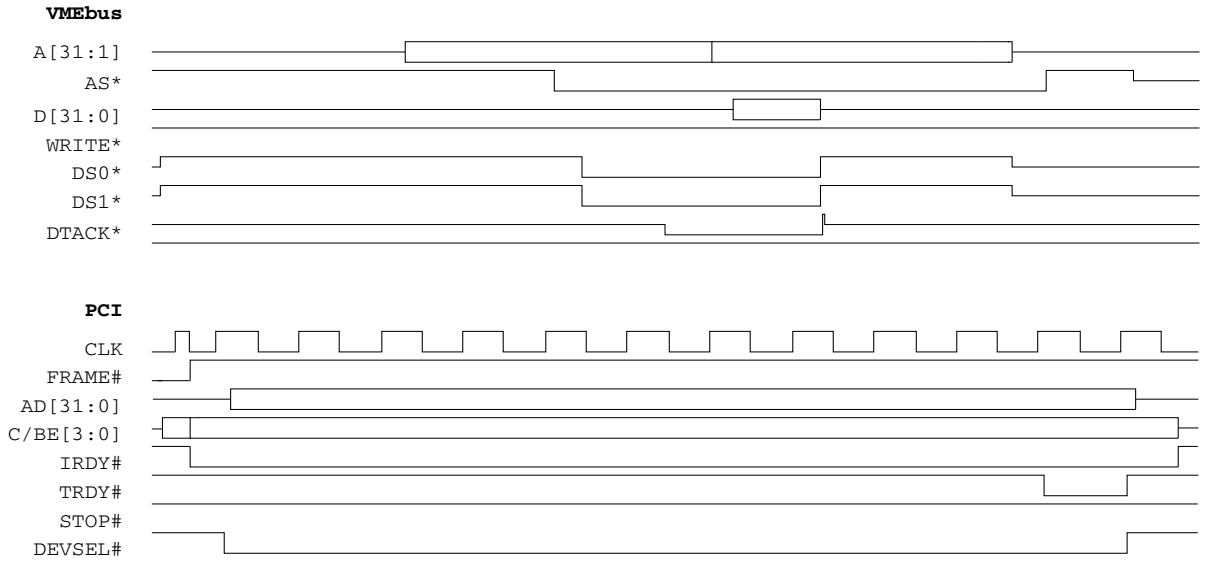
The address of an incoming VME transaction is decoded and translated to an equivalent PCI transaction. Typically, four PCI clock periods elapse between the initial assertion of AS* on the VMEbus and the assertion of REQ_ on the PCI bus. During the data only portion of subsequent beats in block transfers, 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 32 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 32: Coupled Read Cycle - Universe II as VME Slave



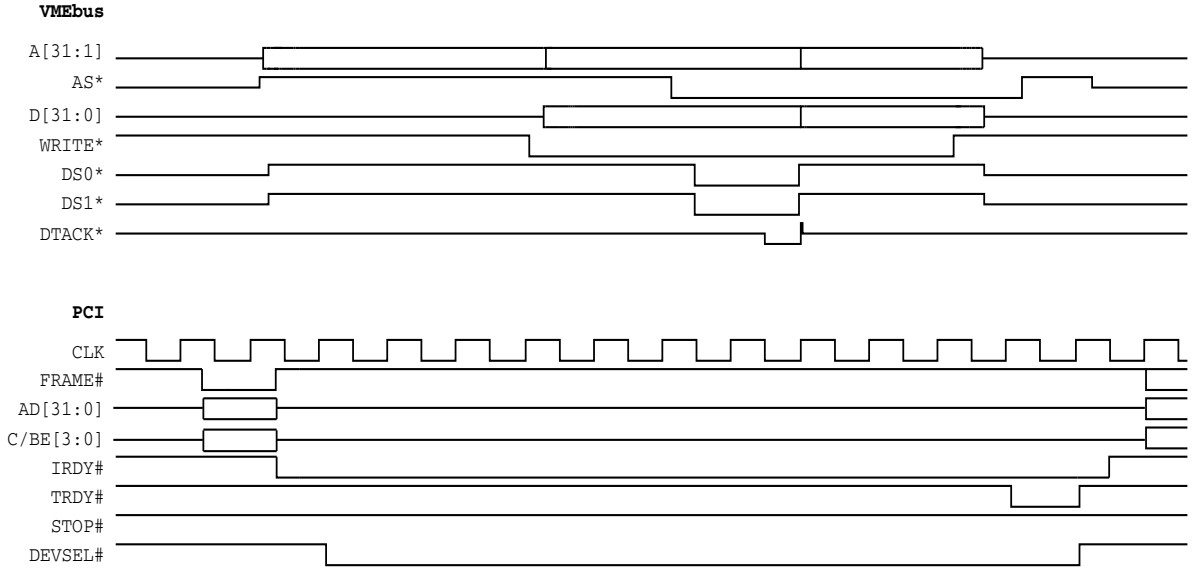
B.3.1.3 Write Cycles

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 33 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 33: Coupled Write Cycle - Universe II as VME Slave (bus parked at Universe II)



B.3.2 Decoupled Cycles

B.3.2.1 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 34: Non-Block Decoupled Write Cycle - Universe II as VME Slave

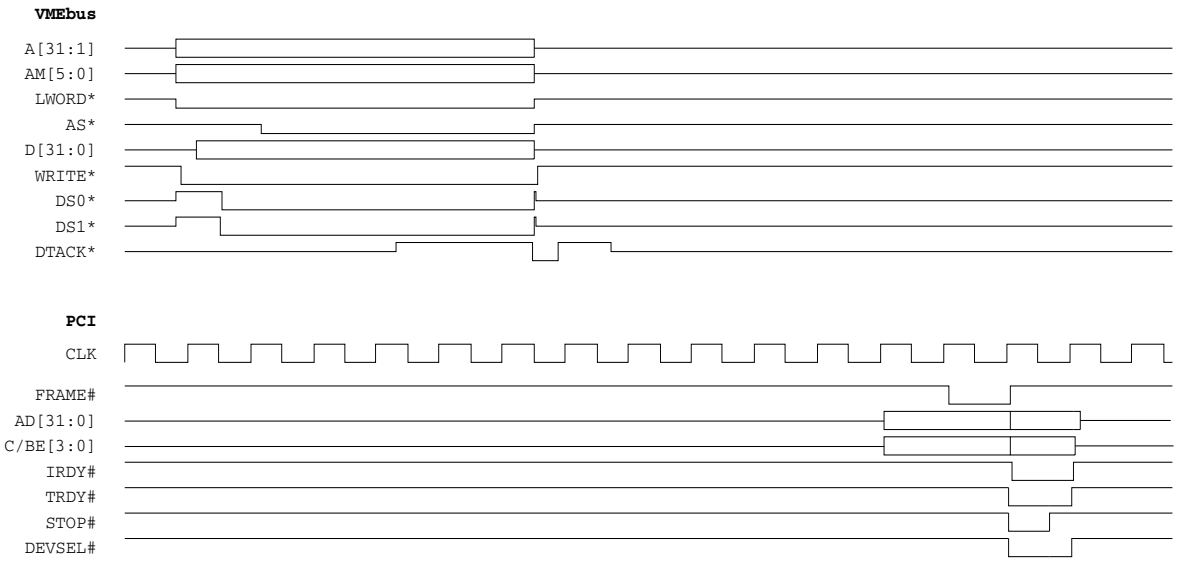


Figure 35: BLT Decoupled Write Cycle - Universe II as VME Slave

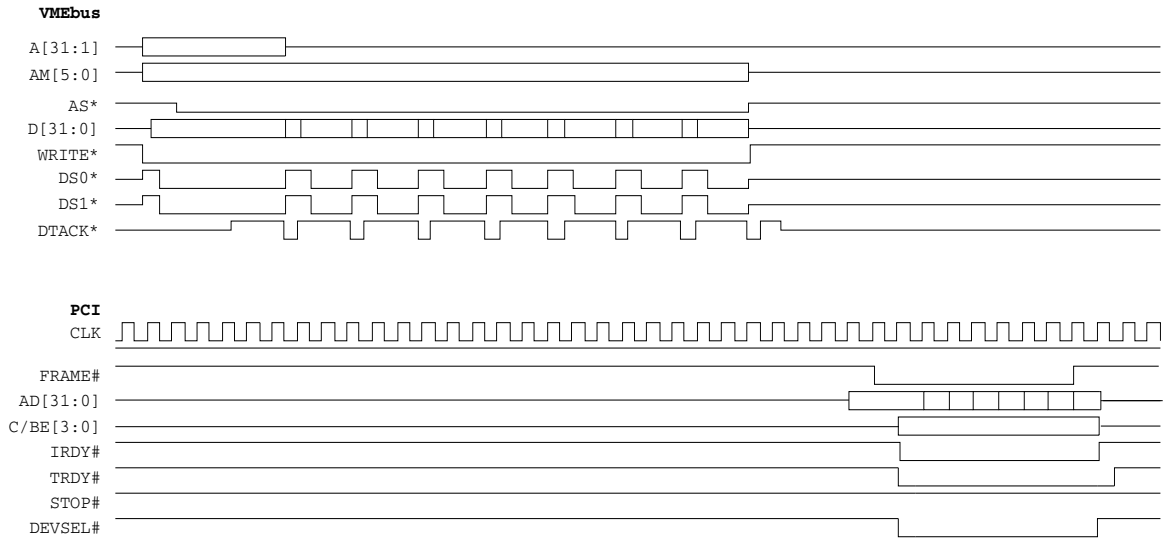
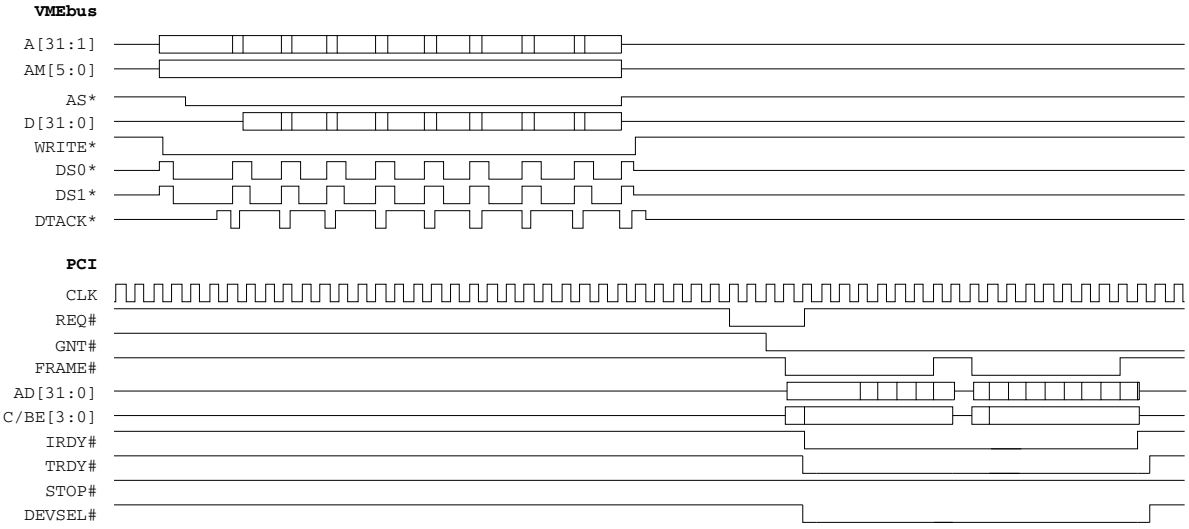


Figure 36: MBLT Decoupled Write Cycle - Universe II as VME Slave



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 deliquesce 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 decreed 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 27 through Figure 36 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.

B.3.2.2 Prefetched Read Cycles

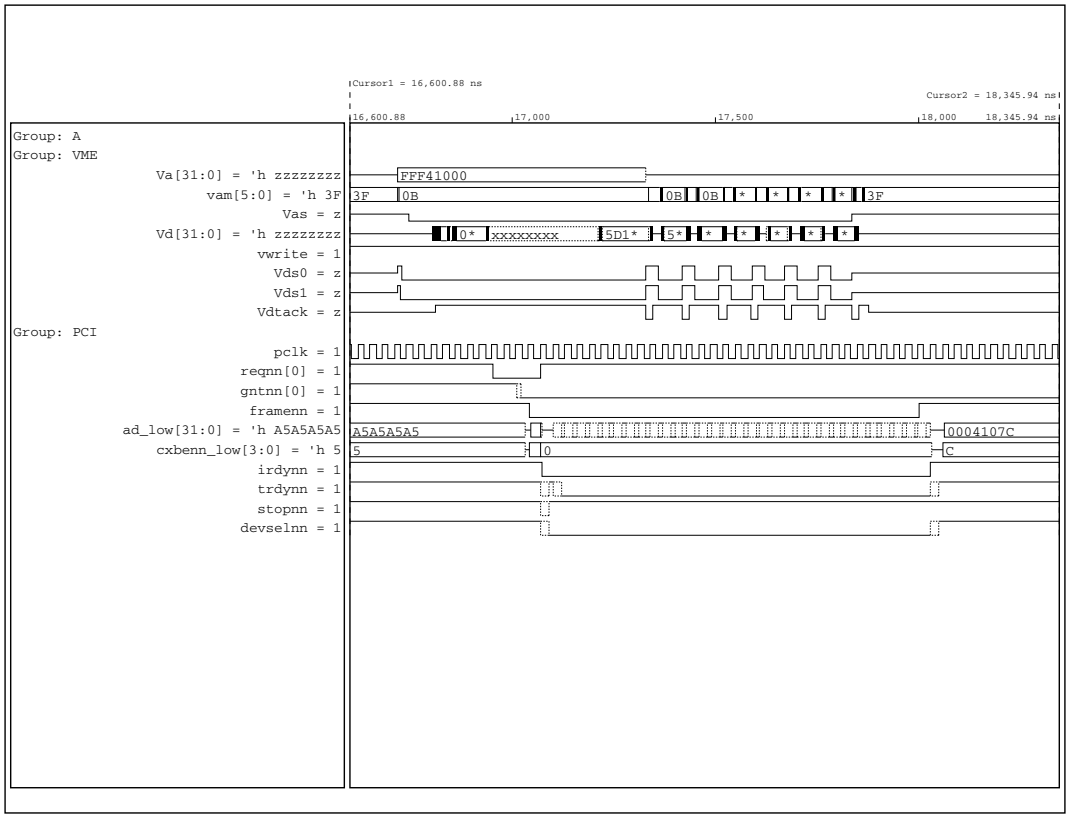
To minimize its slave response, the Universe II generates prefetched reads to the PCI bus in response to BLT and MBLT reads coming in from the VMEbus. This option must first be enabled on a per image basis.

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 37: BLT Pre-fetched Read Cycle - Universe II as VME Slave



B.4 DMA Channel and 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.

B.4.1 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.

B.4.2 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.

B.4.2.1 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.

B.4.2.2 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.

B.4.3 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 38: PCI Read Transactions During DMA Operation

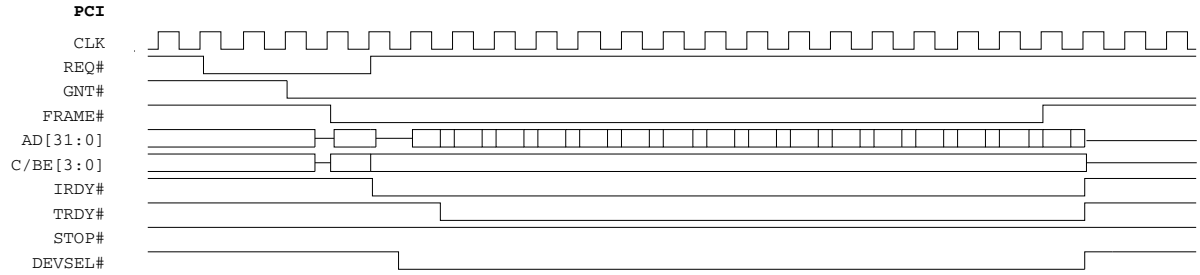
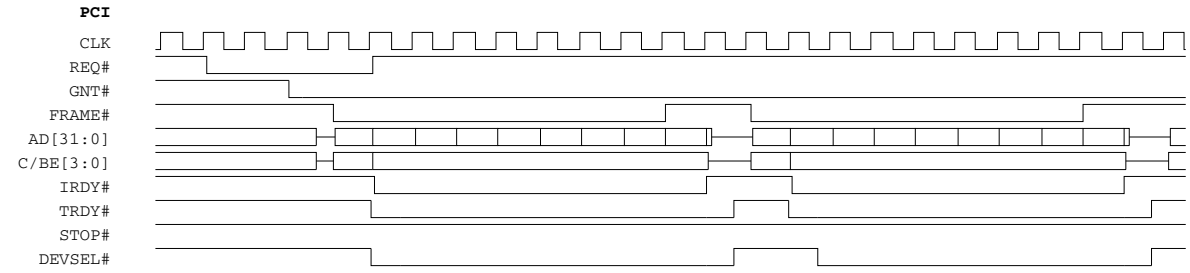


Figure 39: Multiple PCI Read Transactions During DMA Operation



B.5 Universe II Specific Register

The Universe II Specific Register, U2SPEC, offset 0x4FC, can be used to improve the performance of the Universe II by reducing the latency of key VMEbus timing elements. This register is present in versions of the Universe device which have a Revision ID of 01or 02 — defined in the PCI_CLASS register, offset 008.

B.5.1 Overview of the U2SPEC Register

Although the VMEbus is asynchronous, there are a number of maximum and minimum timing parameters which must be followed. These requirements are detailed in the *VME64 Specification*.

In order to qualify as compliant the master, slave and location monitor devices must guarantee they meet these timing parameters independent of their surroundings. They must assume zero latency between themselves and the VMEbus. This, in practice, is never the case. Buffers, transceivers and the backplane itself, all introduce latencies that combine to produce additional system delay. The consequence of such delay is the degradation of overall performance.

The Universe II's U2SPEC register enables users to compensate for the latencies which are inherent to their VMEbus system designs. Through the use of this register, users can reduce the inherent delay associated with five key VMEbus timing parameters.

Use of the U2SPEC register may result in violation of the *VME64 Specification*.

B.5.2 Adjustable VME Timing Parameters

B.5.2.1 VME DTACK* Inactive Filter (DTKFLTR)

In order to overcome the DTACK* noise typical of most VME systems, the Universe II quadruple samples this signal with the 64 MHz clock. The extra sampling is a precaution that results in decreased performance. Users who believe their systems to have little noise on their DTACK* lines can elect to filter this signal less, and therefore increase their Universe II response time.

B.5.2.2 VME Master Parameter t11 Control (MASt11)

According to the *VME64 Specification*, a VMEbus master must not drive DS0* low until both it and DS1* have been simultaneously high for a minimum of 40 ns. The MASt11 parameter in the U2SPEC register, however, allows DS0* to be driven low in less than 40 ns.

B.5.2.3 VME Master Parameter t27 Control (READt27)

During read cycles, the VMEbus master must guarantee the data lines are valid within 25 ns after DTACK* is asserted. The master must not latch the data and terminate the cycle for a minimum of 25 ns after the falling edge of DTACK*.

The READt27 parameter in the U2SPEC register supports faster cycle termination with one of two settings. One setting allows data to be latched and the cycle terminated with an associated delay that is less than 25 ns. The second setting results in no delay in latching and termination.

B.5.2.4 VME Slave Parameter t28 Control (POSt28)

According to the *VME64 Specification*, VMEbus slaves must wait at least 30 ns after the assertion of DS* before driving DTACK* low. When the Universe II is acting as a VME slave, the POSt28 parameter in the U2SPEC register enables DTACK* to be asserted in less than 30 ns when executing posted writes.

B.5.2.5 VME Slave Parameter t28 Control (PREt28)

VMEbus slaves must wait at least 30ns after the assertion of DS* before driving DTACK* low. When the Universe II is acting as a VME slave in the transaction, PREt28 parameter in the U2SPEC register enables DTACK* to be asserted in less than 30 ns when executing pre-fetched reads.

B.6 Performance Summary

Table 36: 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 — sustained perf (32-byte PABS) — sustained perf (64-byte PABS) D32 BLT — VME cycle time — sustained perf (32-byte PABS) — sustained perf (64-byte PABS) D64 MBLT — VME cycle time — sustained performance (32-byte PABS) — sustained perf (64-byte PABS)	 180 ns 23 Mbytes/s 43 Mbytes/s 119 ns 32 Mbytes/s 35 Mbytes/s 119 ns 53 Mbytes/s 59 Mbytes/s

Table 37: VME Slave Channel Performance

Cycle Type	Performance VME Slave Response (ns)
Coupled Read — non-block — D32 BLT — D64 BLT	 301 293 322
Coupled Write — non-block — D32 BLT — D64 BLT	 278 264 292

Table 37: VME Slave Channel Performance

Cycle Type	Performance VME Slave Response (ns)
Pre-fetched Read — VME slave response (1st data beat) — VME slave response (other data beats)	 293 57
Decoupled Write — non-block slave response — block slave response (1st data beat) — block slave response (other data beats)	 127 127 50

Table 38: DMA Channel Performance

Cycle Type	Performance Mbytes/s
PCI Reads — 32-byte PABS — 64byte PABS	 97 (194)[a] 118 (236)
PCI Writes — 32-byte PABS — 64byte PABS	 98 (196) 125 (250)
VME Reads — non-block D32 — D32 BLT — D64 MBLT	 18 22 45
VME Writes — non-block D32 — D32 BLT — D64 MBLT	 22 32 65

a. 64-bit PCI performance in brackets.

C. Reliability Prediction

This appendix discusses the following topics:

C.1 Overview

This section is designed to help the user to estimate the inherent reliability of the Universe II. 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.

C.2 Physical Characteristics

- CMOS gate array
- 120,000 two-input NAND gate equivalence
- 0.5 µm feature size
- 309 mils x 309 mils scribed die size

C.3 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

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 consumes 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.

The HTOL FIT rate is 67 FITs. The HTOL test showed 67 FITs based on the calculation of 60% Confidence Level (C.L.) and Activation Energy Ea=0.7eV for 0.5um process at stress condition 1.1 x Vcc at 125°C ambient temperature This FIT rate is approximately equivalent to 90% C.L. 167 FITs. Calculations were based on a 100 piece sample size for three lots. The test conditions were at 125°C/Bias, 5.5 V at 1 MHz per MIL-STD-883D.M1015.8.

FIT 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.

C.4 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 shown in Table 39.

Table 39: Ambient to Junction Thermal Impedance

Air Flow (m/s)	0	1	2
313 PBGA	15.10	13.10	11.70

For example, the maximum ambient temperature of the 313 PBGA, 32-bit PCI environment with 100 LFPM blowing past the Universe II is:

$T_a \leq T_j - \theta_{ja} * P$

$T_a \leq 125 - 13.1* 2.0$

$T_a \leq 98.8$ °C

Therefore the maximum rated ambient temperature for the Universe II in this environment is 98.8°C. Further improvements can be made by adding heat sinks to the PBGA package.

T_j values of Universe II are calculated as follows (Tj = θja * P + Ta)

Table 40: Maximum Universe II Junction Temperature

Extended (125 °C Ambient)[a]	Industrial (85 °C Ambient)	Commercial: (70 °C Ambient)
Tj=13.1*2.7+125=160.37°C	Tj=13.1*2.7+85=120.37°C	Tj=13.1*2.7+70=105.37°C

a. IDT recommends that the maximum junction temperature of the Universe II does not exceed 150 °C. This temperature limit can be achieved by using heat dissipation techniques, such as heat sinks and forced airflows.

Table 41 shows the simulated Psi jt and Theta jc thermal characteristics of the Universe II package.

Table 41: Thermal Characteristics of Universe II

Interface	Result
Psi jt	0.25 °C/watt
Theta jc (junction to case)	4.7 °C/watt

These values were obtained under the following PCB and environmental conditions:

- PCB condtionsMaximum junction
  - PCB standard: JEDEC JESD51-9
  - PCB layers: 4
  - PCB dimensions: 101.6 mm x 114.3 mm
  - PCB thickness: 1.6 mm
- Environmental conditions
  - Maximum junction temperature: 125°C
  - Ambient temperature: 70°C
  - Power dissipation: 3 W

C.5 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.

D. Endian Mapping

Universe II has Little-endian mapping. Little-endian refers to a method of formatting data where address 0 (or the smallest address referencing the data) points to the least significant byte of the data. Data in a system must be consistent; that is, the system must be entirely big-endian or little-endian.

This chapter discusses the following topics:

D.1 Overview

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.

D.2 Little-endian Mode

Table 42 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 42: Mapping of 32-bit Little-Endian PCI Bus to 32-bit VMEbus

PCI Bus											
Byte Enables				Address		Byte Lane Mapping			VMEbus			
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	0	1	0	0
						D8-D15	<->	D16-D23				
						D16-D23	<->	D8-D15				
0	0	0	1	0	1	D8-D15	<->	D16-D23	1	0	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 43 below.

Table 43: 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

E. Typical Applications

This chapter discusses the following topics:

E.1 Overview

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.

E.2 VME Interface

E.2.1 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 40) 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 40 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
- Meet Universe II skew requirements, and
- Minimize part counts.

U15 and U17 in Figure 40 are optional devices and provide better noise immunity for the system.

Figure 40: Universe II Connections to the VMEbus Through TTL Buffers

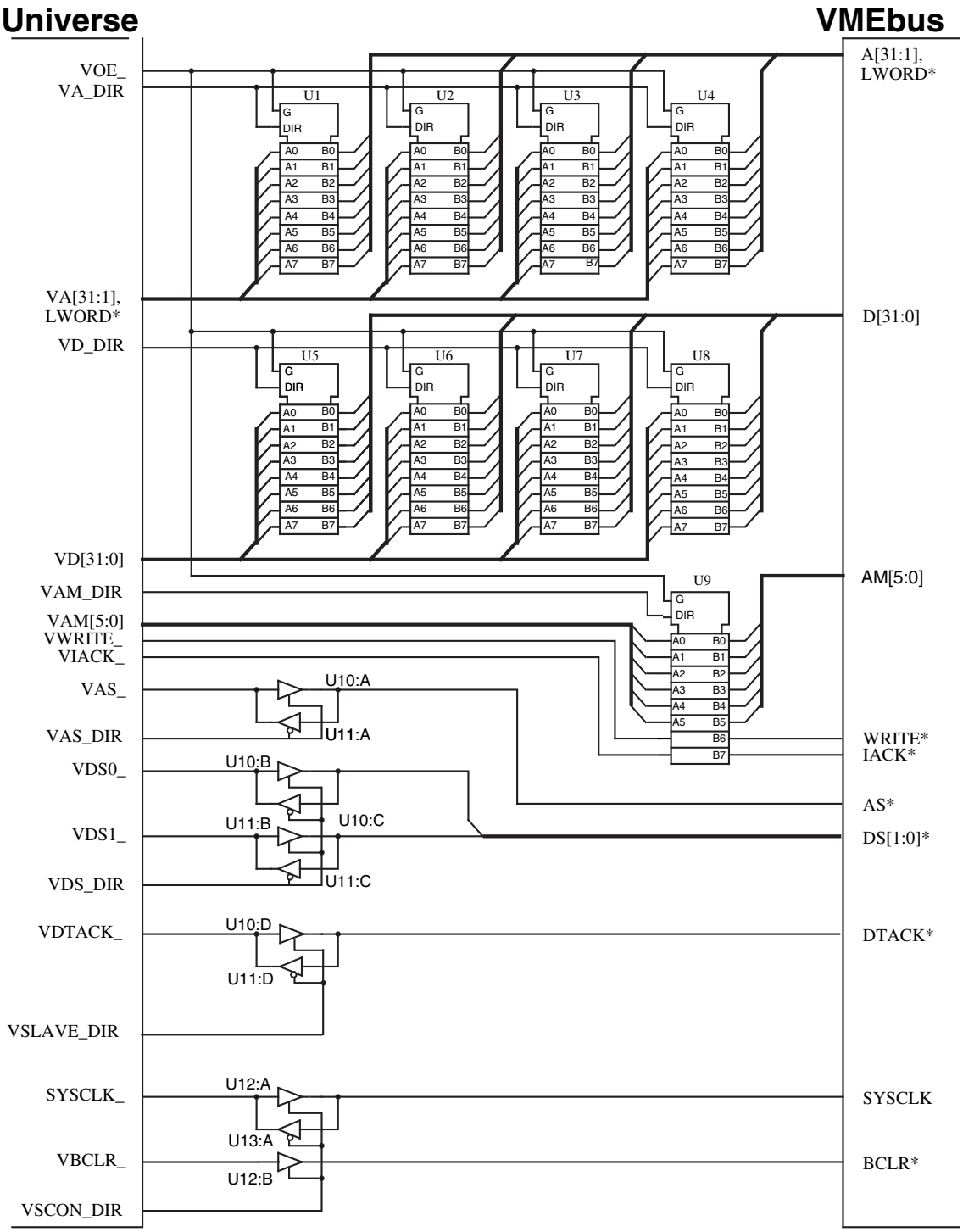
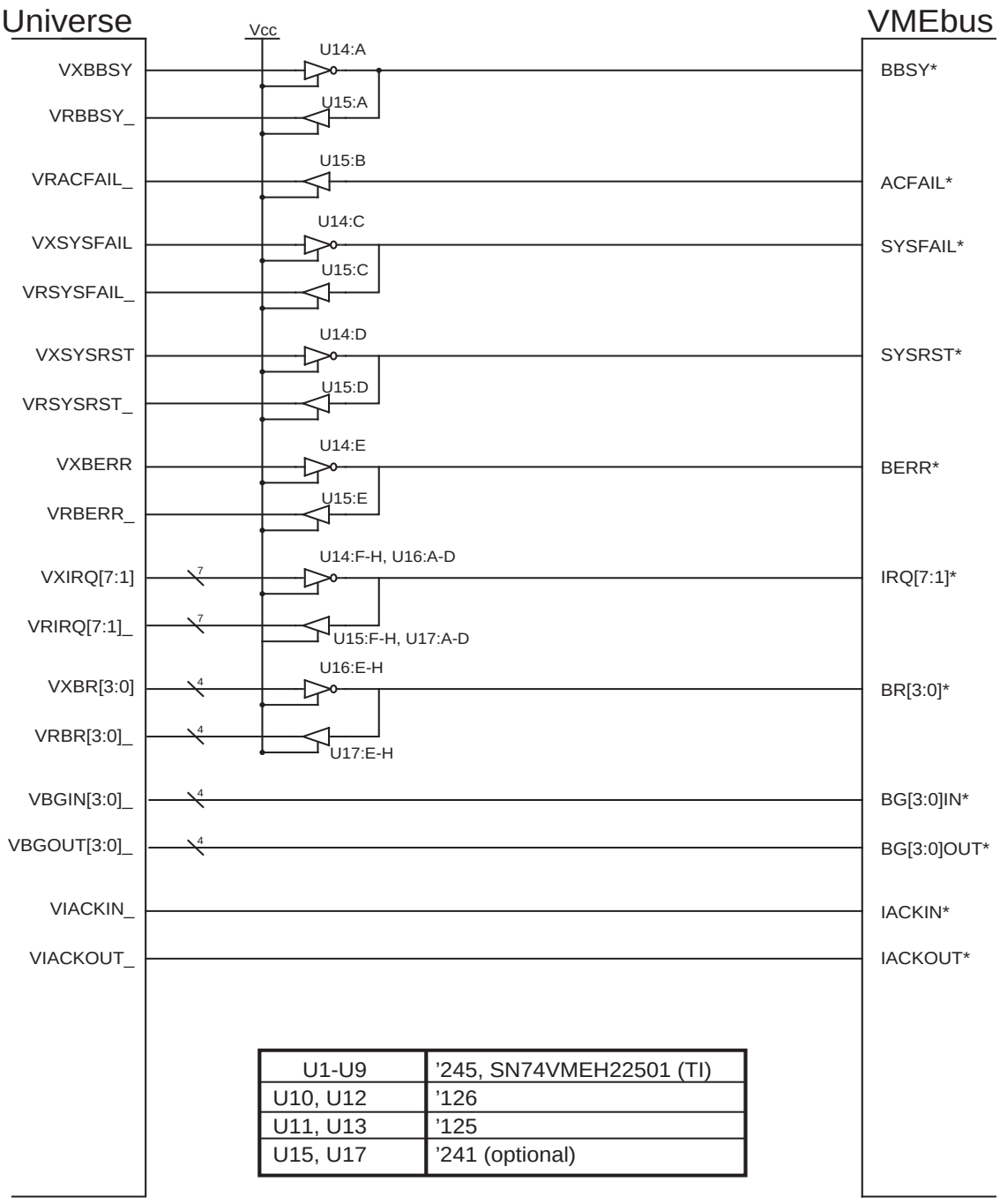


Figure 41: Universe II Connections to the VMEbus Through TTL Buffers



U1-U9	’245, SN74VMEH22501 (TI)
U10, U12	’126
U11, U13	’125
U15, U17	’241 (optional)

U14 and U16 were originally 642 buffers, but the 642 buffer is now obsolete. IDT recommends any other open drain buffer as a replacement, such as a F06.

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 45.

Table 44: 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 45: 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

a. 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.

F Series transceivers meet the requirements specified in Table 44 and Table 45. 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.

E.2.1.1 Pull-down resistors

The Universe II has internal pull-down resistors which are used for its default power-up option state. (Note that REQ64_ has an internal pull-up.) These internal pull-down resistors, ranging from 25kΩ-500kΩ, 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, IDT recommends connecting a 1K resistor to ground, in parallel, with the internal pull-down resistor.

IDT recommends that any pins controlling the power-up options which are critical to the application at power-up 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.

E.2.2 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.

E.2.3 Power-up Options

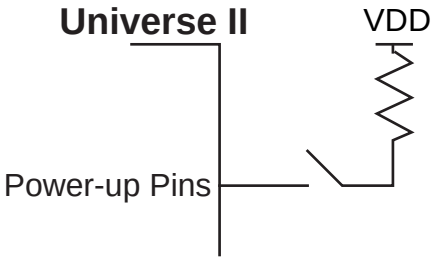
Power-up options for the automatic configuration of slave images and other Universe II features are provided through the state of the VME address and data pins, VA[31:1] and VD[31:27]. All of these signals are provided with internal pull-downs to bias these signals to their default conditions. Should 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 “Resets, Clocks and Power-up Options” on page 129.

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_.

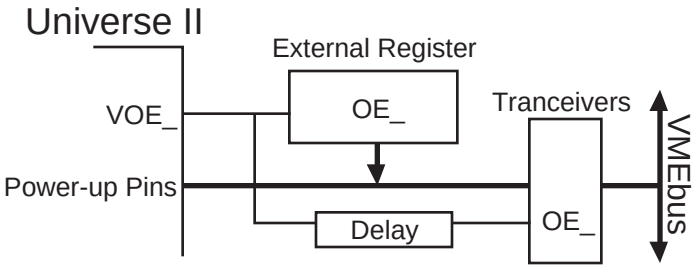
Configuration of power-up options is most easily accomplished through passive 10k pull-up resistors on the appropriate VA and VD pins. The configurations may be made user-configurable through jumpers or switches as shown in Figure 42

Figure 42: 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 43 shows one such implementation. The delay for enabling of the VMEbus transceivers could be implemented though clocked latches.

Figure 43: Power-up Configuration Using Active Circuitry



E.2.3.1 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.

E.3 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 4.7kΩ resistor) at reset for 64-bit PCI (see "PCI Bus Width" on page 138). 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.

E.3.1 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 46.

Table 46: Reset Signals

Group	Signal Name	Direction
VMEbus	VXSYSRST	output
	VRSYSRST_	input
PCI bus	LRST_	output
	RST_	input
	VME_RESET_	input
Power-up	PWRRST_	input

E.3.1.1 VMEbus Resets

The VMEbus resets are connected to the VMEbus as indicated in Figure 40 on page 364 through external buffers.

E.3.1.2 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_ and reset_source1 and reset_source2 and...

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.

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.

E.3.1.3 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_.

E.3.1.4 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.

E.3.2 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.

E.4 Manufacturing Test Pins

The Universe II has several signals used for manufacturing test purposes. They are listed in Table 24 on page 140, along with the source to which they should be tied.

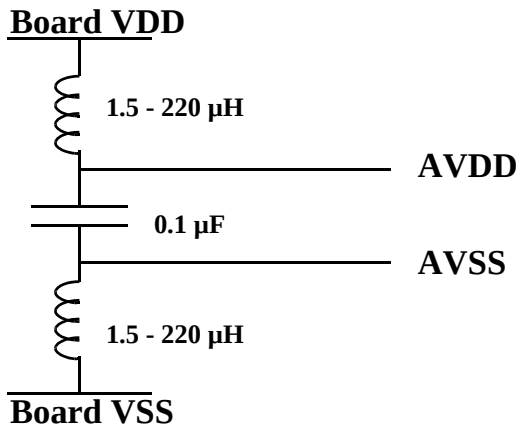
E.5 Decoupling V_{DD} and V_{SS} on the Universe II

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:

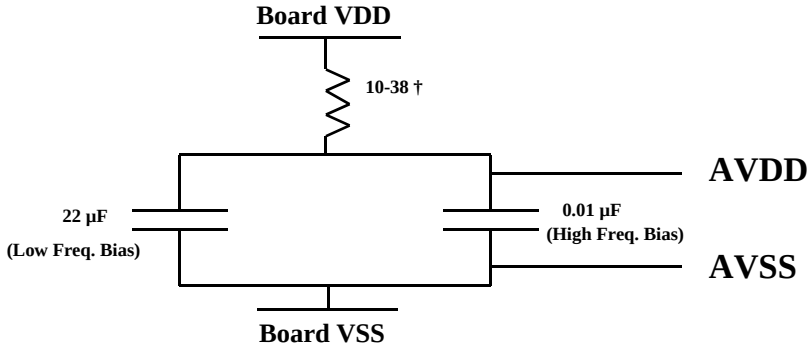
- a 0.1µF capacitor between the AV_{DD} and AV_{SS} pins, and
- corresponding inductors between the pins and the board power and ground planes (See Figure 44). These inductors are not necessary, but they are recommended.

Figure 44: Analog Isolation Scheme



The Noise Filter Scheme filters out the noise using two capacitors to filter high and low frequencies (See Figure 45).

Figure 45: 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.

F. Ordering Information

This appendix discusses Universe II's ordering information.

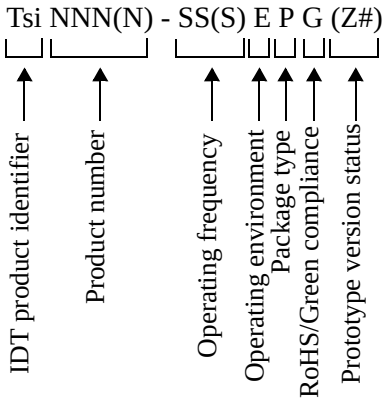
F.1 Ordering Information

IDT products are designated by a product code. When ordering, refer to products by their full code. Table 47 details the available part numbers.

Table 47: Standard Ordering Information

Part Number	PCI Frequency	Voltage	Temperature	Package
CA91C142D-33CE	33 MHz	5 V	Commercial (0° to 70°C)	PBGA
CA91C142D-33CEV	33 MHz	5 V	Commercial (0° to 70°C)	PBGA (RoHS/Green)
CA91C142D-33IE	33 MHz	5 V	Industrial (-40° to 85°C)	PBGA
CA91C142D-33IEV	33 MHz	5 V	Industrial (-40° to 85°C)	PBGA (RoHS/Green)
CA91C142D-25EE	25 MHz	5 V	Extended (-55° to 125°C)	PBGA

The IDT “Tsi” part numbering system is explained as follows.



- ( ) – Indicates optional characters.
- Tsi – IDT “Tsi” product identifier.
- NNNN – Product number (may be three or four digits).
- SS(S) – Maximum operating frequency or data transfer rate of the fastest interface. For operating frequency numbers, M and G represent MHz and GHz. For transfer rate numbers, M and G represent Mbps and Gbps.
- E – Operating environment in which the product is guaranteed. This code may be one of the following characters:
  - C - Commercial temperature range (0 to +70°C)
  - I - Industrial temperature range (-40 to +85°C)
  - E - Extended temperature range (-55 to +125°C)
- P – The Package type of the product:
  - B - Ceramic ball grid array (CBGA)
  - E, L, J, and K - Plastic ball grid array (PBGA)
  - G - Ceramic pin grid array (CPGA)
  - M - Small outline integrated circuit (SOIC)
  - Q - Plastic quad flatpack (QFP)

- G – IDT “Tsi” products fit into three RoHS-compliance categories:
  - Y - RoHS Compliant (6of6) – These products contain none of the six restricted substances above the limits set in the EU Directive 2002/95/EC.
  - Y - RoHS Compliant (Flip Chip) – These products contain only one of the six restricted substances: Lead (Pb). These flip-chip products are RoHS compliant through the Lead exemption for Flip Chip technology, Commission Decision 2005/747/EC, which allows Lead in solders to complete a viable electrical connection between semiconductor die and carrier within integrated circuit Flip Chip packages.
  - V - RoHS Compliant/Green - These products follow the above definitions for RoHS Compliance and meet JIG (Joint Industry Guide) Level B requirements for Brominated Flame Retardants (other than PBBs and PBDEs).
- Z# – Prototype version status (optional). If a product is released as a prototype then a “Z” is added to the end of the part number. Further revisions to the prototype prior to production release would add a sequential numeric digit. For example, the first prototype version of device would have a “Z,” a second version would have “Z1,” and so on. The prototype version code is dropped once the product reaches production status.

CORPORATE HEADQUARTERS
6024 Silver Creek Valley Road
San Jose, CA 95138

for SALES:
800-345-7015 or 408-284-8200
fax: 408-284-2775
www.idt.com

for Tech Support:
email: VMEhelp@idt.com
phone: 408-360-1538
Document: May 12, 2010

May 2010