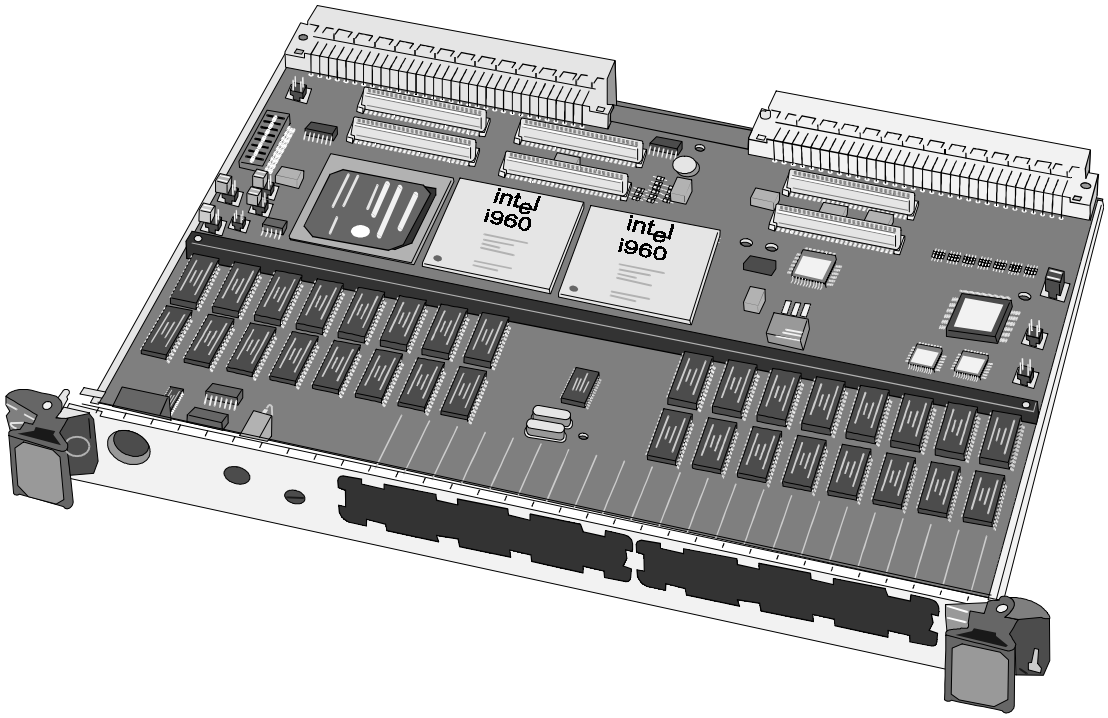
User's Manual

MIDAS-100/200 series

PMC I/O Subsystem for VMEbus and RACEway

Rev. 1.0b - Valid for 'MIDAS PCB-B'



The information in this document is subject to change without notice and should not be construed as a commitment by VMETRO. While reasonable precautions have been taken, VMETRO assumes no responsibility for any errors that may appear in this document.

VMETRO

Warranty

VMETRO products are warranted against defective materials and workmanship within the warranty period of 1 (one) year from date of invoice. Within the warranty period, VMETRO will, free of charge, repair or replace any defective unit covered by this warranty, shipping prepaid. A Return Authorization Code should be obtained from VMETRO prior to return of any defective product. With any returned product, a written description of the nature of malfunction should be enclosed. The product must be shipped in its original shipping container or similar packaging with sufficient mechanical and electrical protection in order to maintain warranty.

This warranty assumes normal use. Products subjected to unreasonably rough handling, negligence, abnormal voltages, abrasion, unauthorized parts replacement and repairs, or theft are not covered by this warranty and will if possible be repaired for time and material charges in effect at the time of repair.

VMETRO's warranty is limited to the repair or replacement policy described above and neither VMETRO nor its agent shall be responsible for consequential or special damages related to the use of their products.

Limited Liability

VMETRO does not assume any liability arising out of the application or use of any product described herein; neither does it convey any license under its patent rights nor the rights of others. VMETRO products are not designed, intended, or authorized for use as components in systems intended to support or sustain life, or for any application in which failure of the VMETRO product could create a situation where personal injury or death may occur. Should Buyer purchase or use VMETRO products for any such unintended or unauthorized application, Buyer shall indemnify and hold VMETRO and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that VMETRO was negligent regarding the design or manufacture of the part.

USA:

VMETRO, Inc.
1880 Dairy Ashford, Suite 535
Houston, TX 77077, USA
Tel.: (281) 584-0728
Fax: (281) 584-9034
Email: info@vmetro.com

Europe, Asia:

VMETRO asa
Brynsveien 5
N-0667 OSLO, Norway
Tel.: +47 2210 6090
Fax: +47 2210 6202
Email: info@vmetro.no

http://www.vmetro.com/

Contents

General Information 1

This document 1
Conventions used in this document 1
Related Documents....2

Product Overview 3

MIDAS Family 3
The MIDAS-200 Series 3
Twin i960RD I/O processors....3
The MIDAS-100 Series 4
i960RD I/O processor....4
Main Components....5
i960®RD I/O Processor 5
Universe II PCI-VMEbus Bridge 6
PXB PCI-RACEway Bridge....7

Installation 7

Board Precautions....7
Unpacking....7
Board Layout 8
Installation of PMC Modules....8
Assembly Procedure for MIDAS-x50....9
Installation in VMEbus System 12
Slot selection 12
Power consumption 12
Configuration Switch & Jumpers....13

Functional Description 14

i960®RD and its Surroundings....14
i960®RD Address Map....15
i960®RD Power-Up Options 15
FLASH Memory....16
RS232 Interface....17
Connecting two i960®RDs....17
VMEbus Interface (Universe II VME-PCI Bridge)....18
'Universe II' Power-Up Options....18
Auto-Slot ID....19
Configuration ROM....19
Switch & Jumper Descriptions 21
Reset Button 24
RACEway Interface (PXB RACEway-PCI Bridge) 24
Jumper Descriptions 24
PCI Bus Details 25
Arbitration 25
'IDSEL' Generation....27
Subtractive Decoding Agent....28
Interrupt Routing 29
Interrupt Mode Selection....29
Interrupt Jumpers....30
Interrupt Routing Tables 30

Interrupt Mode 0: 31
Interrupt Mode 1: 32
Interrupt Mode 2: 33

Appendix I: PMC I/O Routing 35

PMC I/O Routing Scheme 35

Appendix II: Universe II Configuration Examples 36

General Information 36
VMEbus Slave Images 36
PCI Master Enable 36
VMEbus Register Access Image 37
VMEbus Slave Image 0 38
VMEbus Slave Image 1 38
VMEbus Slave Image 2 39
Initialization Sequence 39
PCI Slave Images 40
PCI Target Enable - Memory & I/O Space 40
PCI Slave Image 0 41
PCI Slave Image 1 42
PCI Slave Image 2 42
Initialization Sequence 43

Appendix III: PXB Information 44

PXB Register Descriptions 44
P-Side Register Descriptions 44
X-Side Register Descriptions 51
Miscellaneous PXB Information 52
Configuration Serial EEPROM 52
PCI-to-RACEway Addressing 53
RACEway-to-PCI Addressing 55
PCI-to-PCI Bridge Operation 56
PXB Initialization Example 57

Appendix IV: 61

List of Tables 61
List of Figures 62

General Information

This document

This document has been prepared to help the customer integrate MIDAS in their VMEbus system. The following models are covered by this document:

MIDAS-120: Intelligent I/O Subsystem with single i960®RD and memory, and two PMC positions.

MIDAS-150: Intelligent I/O Subsystem with single i960®RD and memory, and five PMC positions. Occupies two VMEbus slots.

MIDAS-220: Intelligent I/O Subsystem containing dual i960®RD with independent memories, and two PMC positions.

MIDAS-250: Intelligent I/O Subsystem containing dual i960®RD with independent memories, and five PMC positions. Occupies two VMEbus slots.

The following options are described:

-R RACEway option. Interface to 160 MB/s RACEway crossbar.

-S Symmetrical configuration. Configuration option with one PMC position on each PCI bus. Applies to MIDAS-120 and MIDAS-220 only.

Conventions used in this document

The following section describes conventions used in this document.

Symbols	*Meaning:*
	The STOP symbol indicates a section of critical importance. Overlooking this information may cause damage to the MIDAS and/or other equipment.
	Indicates important, but not crucial, information. Still, you should take notice if you want to use all capabilities built into your MIDAS.

Related Documents

This document does not include detailed information about the following main board components:

- i960®RD Intelligent I/O Processor.
- 'Universe II' VME-to-PCI bridge chip.
- 'PXB' RACEway-to-PCI bridge chip.

Since a majority of the control registers, and a large part of the complexity of the MIDAS is implemented in these chips, their documentation contains information, which is essential to the understanding of the product.

VMEbus-PCI (Tundra):	**UNIVERSE II™ USER MANUAL**
i960®RD (Intel Corp.):	**i960®RX I/O MICROPROCESSOR DEVELOPER'S MANUAL**
	i960®RD - DATA SHEET
RACEway-PCI	**PXB OVERVIEW**
	PXB BRIDGE SPECIFICATION

Documentation can be obtained from VMETRO by ordering a MIDAS Documentation Package.

The vendors of some of the major components used on MIDAS publish device data via Internet. Some components may contain design defects or errors known as errata, which may cause the component to deviate from published specifications. The respective vendors, on their web pages document current characterized errata.

Vendor	Component	Available documentation
Tundra (formerly Newbridge)	Universe II (VME-PCI Bridge)	• User Manual • Device Errata • Application Notes
http://www.tundra.com/		
Intel Corporation	80960RD (I/O Processor) Not used on MIDAS-20/50	• User Manual • Device Errata • Application Notes *and more..*
http://www.intel.com/design/i960/		

Product Overview

MIDAS Family

The MIDAS family provides complete PMC I/O Sub-systems for VMEbus and/or RACEway. The boards may carry two or five PMC - PCI Mezzanine Cards, and are designed for effective integration of high-performance PCI I/O functions into VME systems. The family ranges from a very simple 2x PMC Carrier (MIDAS-20), to a powerful Twin i960 & Memory 5x PMC Carrier (MIDAS-250), in which, twin memories and processing power from two CPUs are available.

The MIDAS-200 Series

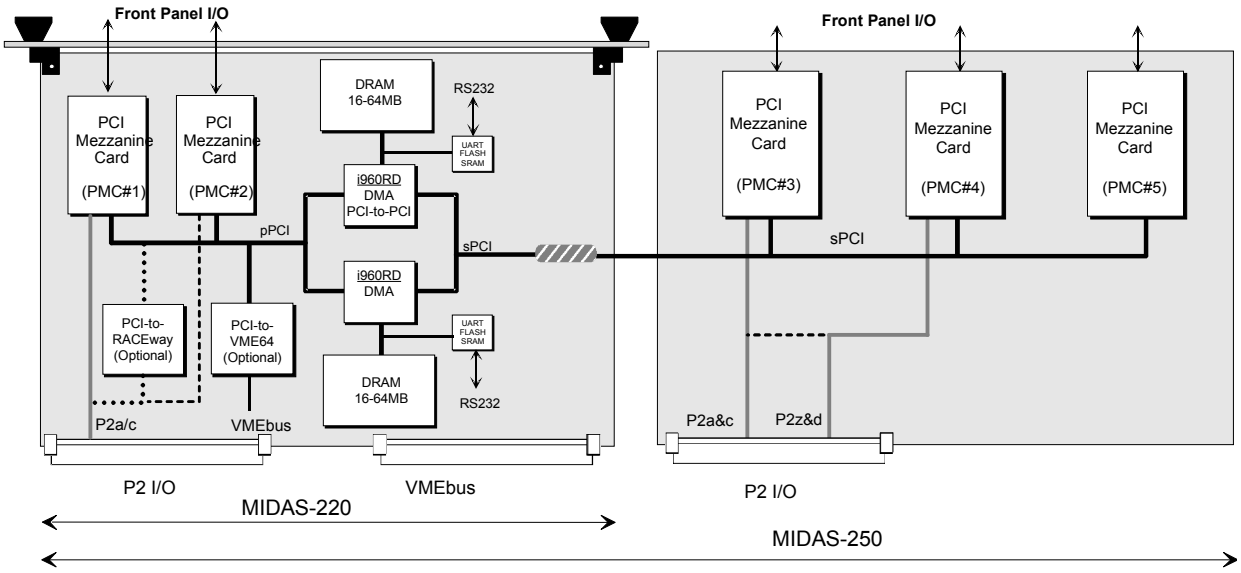


Figure 1. MIDAS-220/250 Block Diagram

The MIDAS-220 series utilize unique, independent, dual i960 processor and Memory arrays, each coupled to a PMC I/O Sub-System, for maximum data through-put to both VMEbus and/or RACEway. The boards carry two or five PMC - PCI Mezzanine Cards, and are designed for effective integration of high-performance PCI I/O functions into the VME or RACEway systems, where twin memories and/or processing power from two CPUs is required. The MIDAS-220 series is particularly well suited for applications such as Imaging, Data Acquisition, and Signal Processing.

Twin i960RD I/O processors

A central element of the MIDAS-220 series is the *twin* Intel i960RD processor architecture. The highly integrated i960RD features a powerful RISC I/O

processor, a PCI-to-PCI bridge, a memory controller and a flexible DMA-controller with linked-list capability.

Each of the two i960RDs has direct access to its own FLASH PROM, a RS232 UART, all of the PCI resources on the board, and of course, its associated DRAM bank. This allows it to offload the host CPU by running dedicated I/O drivers for the PMC modules mounted on the board.

For applications that do not need the processing power of the i960RD, but only twin DRAM memories and/or DMA, the processors may rest idle during normal operation.

The MIDAS-100 Series

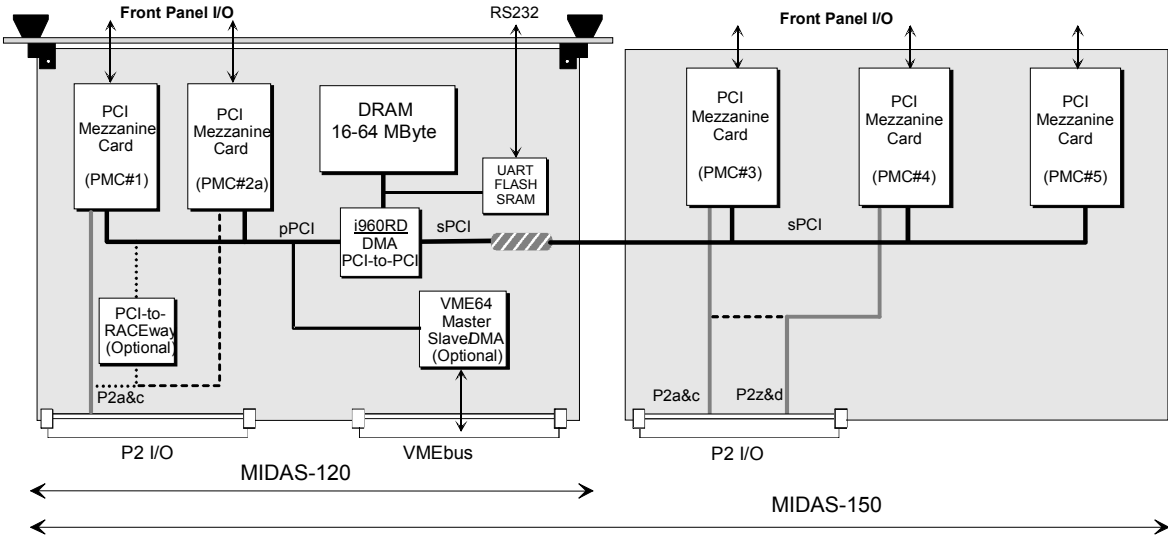


Figure 2. MIDAS-120/150 Block Diagram

The MIDAS-100 series are versatile PMC I/O Sub-System & Memory modules for VMEbus and/or RACEway, holding two or five PMC - PCI Mezzanine Cards. The boards are designed for cost-effective integration of PCI I/O functions into the VME or RACEway systems, where local memory and/or processing power is required.

i960RD I/O processor

A central element of the MIDAS-100 series is the Intel i960RD processor. The highly integrated i960RD features a powerful RISC I/O processor, a PCI-to-PCI bridge, a memory controller and a flexible DMA-controller with linked-list capability.

The i960RD has direct access to FLASH PROM, a RS232 UART, all of the PCI resources on the board, and of course, the DRAM. This allows it to offload the

host CPU by running dedicated I/O drivers for the PMC modules mounted on the board.

Main Components

i960®RD I/O Processor

The i960 RD processor integrates a high-performance 80960 “core” into a Peripheral Components Interconnect (PCI) functionality. This integrated processor addresses the needs of intelligent I/O applications. The primary functional units include an i960 core processor, PCI to PCI bus bridge, PCI-to-80960 Address Translation Unit, Messaging Unit, Direct Memory Access (DMA) Controller, Memory Controller, and Secondary PCI bus Arbitration Unit.

The PCI Bus is an industry standard, high performance, low latency system bus, which operates up to 132 Mbytes/sec. The PCI-to-PCI bridge provides a connection path between two independent 32-bit PCI buses and provides the ability to overcome PCI electrical loading limits. The addition of the i960 core processor brings intelligence to the PCI bus bridge.

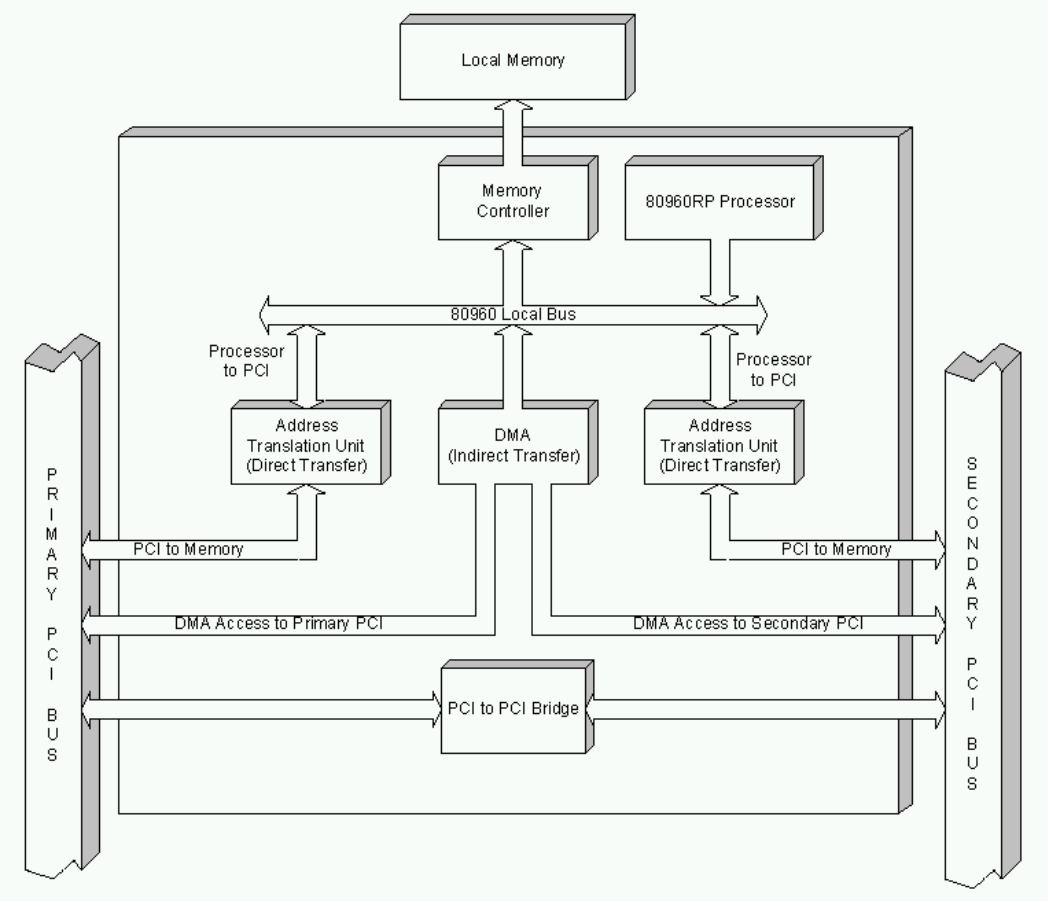


Figure 3. i960®RD Data Flow

The i960 RD processor is a multi-function PCI device: Function 0 is the PCI-to-PCI bridge, Function 1 is the Address Translation Unit. The i960 RD processor contains PCI configuration space, which is accessible through the primary PCI bus. This multi-function PCI device is fully compliant with the PCI Local Bus Specification Revision 2.1.

Universe II PCI-VMEbus Bridge

The Universe II is a second-generation high performance VMEbus to PCI bridge manufactured by Tundra Semiconductor. It features:

- Fully compliant 33 MHz PCI local bus interface.
- Fully compliant, high performance 64-bit VMEbus interface.
- Integral FIFOs for write posting to maximize bandwidth utilization.
- Programmable DMA controller with linked list support.
- VMEbus transfer rates of 60-70 Mbytes/sec.
- Complete suite of VMEbus address and data transfer modes:

- A32/A24/A16 master and slave

- D64 (MBLT)/D32/D16/D08 master and slave

- BLT, ADOH, RMW, LOCK

- Flexible register set, programmable from both the PCI bus and VMEbus ports.
- Full VMEbus system controller functionality.

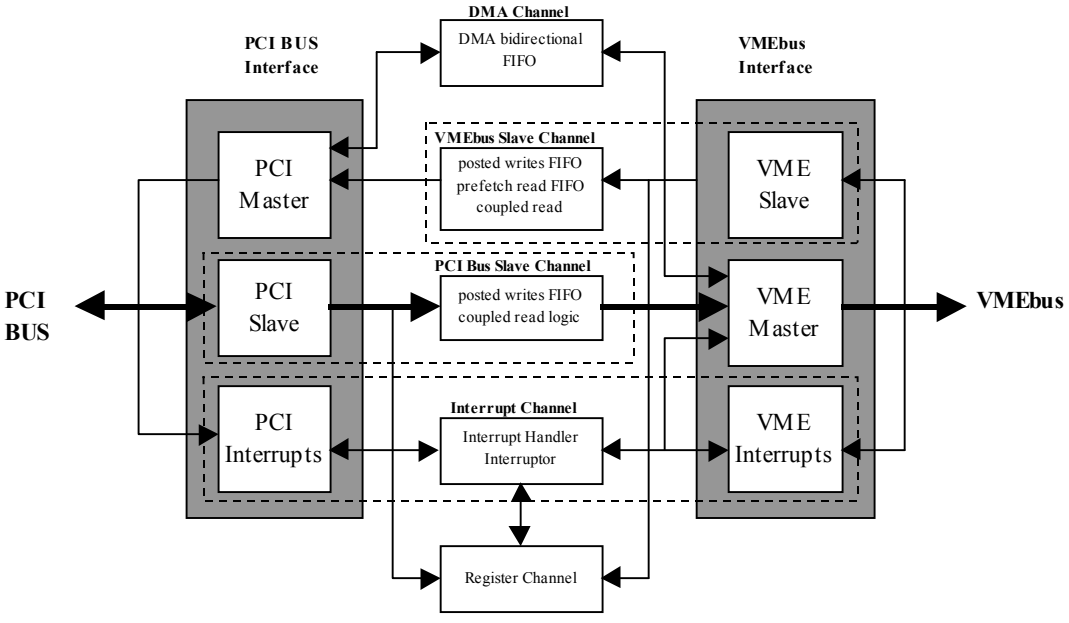


Figure 4 PCI-VME bridge functional diagram

PXB PCI-RACEway Bridge

The PXB is a high performance RACEway to PCI bridge developed by Mercury Computer Systems. It features:

- Bridges a 32 bit, 33MHz PCI bus with a 32 bit, 40MHz RACEway switching fabric.
- Compliant to Rev 2.1 PCI local bus specification, including delayed operations.
- Compliant to Rev 1.0 PCI to PCI Bridge specification.
- Bridges up to sixteen 32 bit, 33MHz PCI busses
- Able to sustain up to 125MB/sec with large memory write transfers, and 100MB/sec with large memory read transfers.
- Integral FIFOs for write posting to maximize bandwidth utilization.

Installation

Board Precautions

The MIDAS circuit board is sensitive to static electricity and can be damaged by a static discharge. Always wear a grounded anti-static wrist strap and use grounded, static protected work surfaces when touching the circuit board and its components.

When the board is not installed, always keep in the static-protective envelope.

Unpacking

All precautions described above must be taken when unpacking the MIDAS from its shipping container. Verify that no damage has occurred in the shipment. Refer to packing list and verify that all items are present.

Board Layout

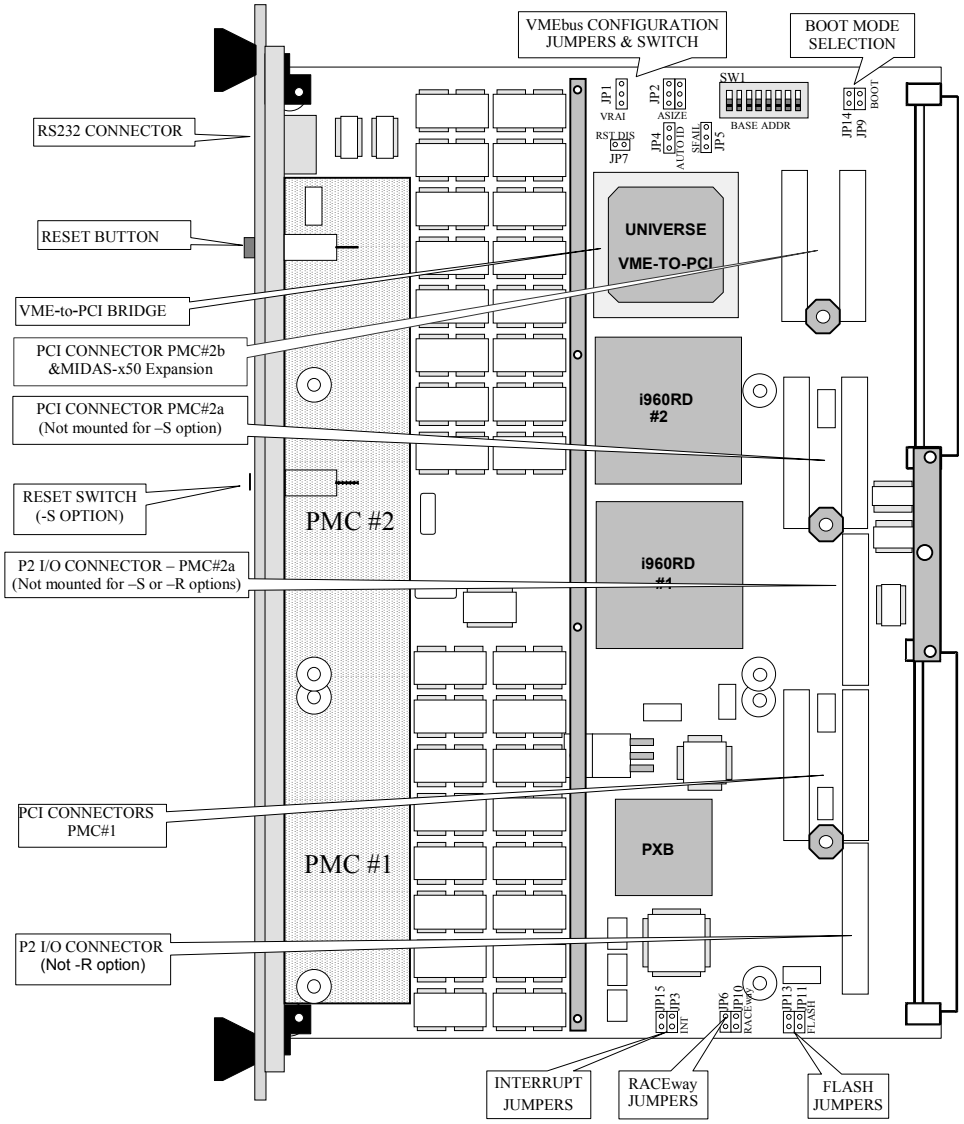


Figure 5. MIDAS Board Layout (No model has all parts mounted.)

Installation of PMC Modules

The MIDAS is shipped with two PMC filler panels mounted in the front panel. They act as EMC shielding in unused PMC positions. Before installing a PMC module, the filler panel(s) must be removed. This is done by pushing them out from the backside of the front panel.

Four screws must be used to secure each PMC on the MIDAS board.

Note: Be extremely careful when inserting screws to secure PMC modules. Touching component leads, or the printed circuit board itself, with a screwdriver may cause permanent damage to the board

Assembly Procedure for MIDAS-x50

The MIDAS-x50 is a dual slot VMEbus board that mates the back-plane connectors in two neighbor slots. The MIDAS-x50 module can handle the insertion and extraction forces applied when installing or removing it from the backplane. However, this requires that the assembly procedure described in this section be followed.

WARNING: The MIDAS-x50 boards may be destroyed during insertion or extraction from a VMEbus system if this procedure is not followed.

STEP#1: Dismount MEZZ-x50 board from MIDAS-x20 board.

- Place the MIDAS-x50 board on a smooth static protected work surface with the bottom side of the MIDAS-x20 board facing up.
- From the bottom side of the MIDAS-x20 PCB, remove the 5 screws holding the metal spacers between the MEZZ-x50 and MIDAS-x20. (these screws are located close to the edge of the board in each corner, and between the VMEbus connectors). **Note:** Do not throw away the screws. They are needed later in this procedure.
- Pull the boards carefully apart. Use hand force only, applied to the two upper VMEbus connectors for both boards.
- If the small SPACER-x50 PCB is attached to the MIDAS-x20 PCB after the separation, remove it and mount it on the bottom side of the MEZZ-x50 board instead.

STEP#2: Mount PMC modules 1 and 2 on the MIDAS-x20 board.

- Place the MIDAS-x20 board on a smooth static protected work surface.
- Install PMC module #1 in the lower PMC position
- Install PMC module #2 in the upper PMC position
- Secure PMC modules with screws on the bottom side of the MIDAS-x20 board.

STEP#3: **Mount PMC modules 3, 4 and 5 on the MEZZ-x50 board.**

- Place the MEZZ-x50 board on a smooth static protected work surface.
- Install PMC module #3 in the lower PMC position.
- Install PMC module #4 in the middle PMC position.
- Install PMC module #5 in the upper PMC position.
- Secure the PMC modules with screws on the bottom side of the MEZZ-x50 board.

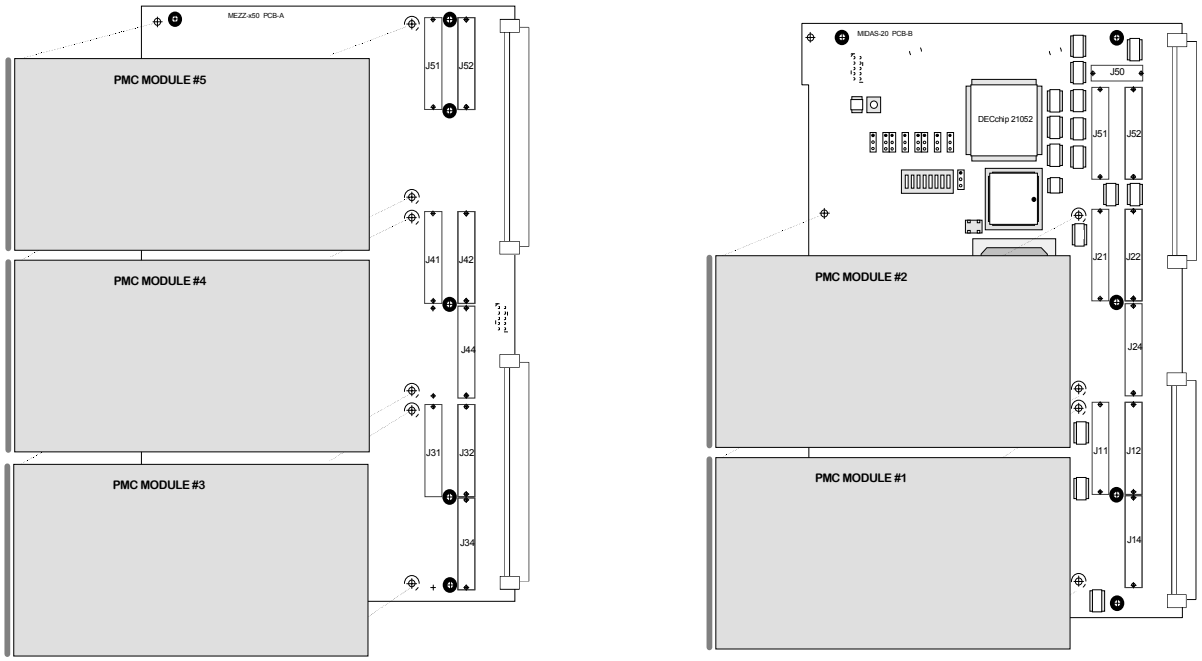


Figure 6. Steps 2&3: Mount PMC modules on the MEZZ-x50 and MIDAS-x20 boards.

Note: Before proceeding, make sure the switch and jumper settings of the MIDAS-x20 board are set according to the needs of your application.

STEP#4: **Mount MEZZ-x50 with PMC modules on the MIDAS-x20 board.**

- If the SPACER-x50 board and the five metal spacers are not already mounted on the bottom side of the MEZZ-x50 board, do it now.
- Place the MIDAS-x20 board on a smooth, static protected work surface.
- Carefully position the MEZZ-x50 board over the MIDAS-x20, so that the three connectors on the bottom side of the spacer, are aligned with the connectors J50, J51 and J52, on the MIDAS-x20. **Make sure that none of the five metal spacers mounted on the MEZZ-x50 board touch components or component leads, on the MIDAS-x20, in the process.**
- Push the MEZZ-x50 board down, so that all connectors mate completely.

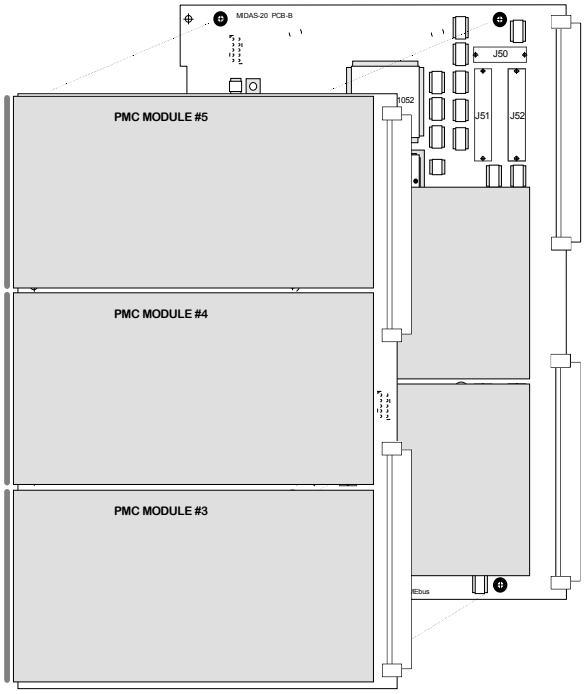


Figure 7. Step 4: Mount the MEZZ-x50 with PMC modules on the MIDAS-x20 board.

STEP#5: **Mount and fasten screws to all metal spacers on the back of the MIDAS-x20 board.**

- Take the 5 screws removed in step#1 of this procedure, and mount them on the back side of the MIDAS-x20 board, through the holes which mate the 5 metal spacers on the MEZZ-x50.
- Fasten all five screws with a suitable screwdriver.
- Verify that the screws attaching the metal spacers to the MEZZ-x50 are also fastened.

- **All screws holding the five metal spacers on both boards, must be firmly fastened in order to make the MIDAS-x50 mechanically stable.**

Installation in VMEbus System

Slot selection

The MIDAS can be installed in any slot in a 6U VMEbus chassis, **as long as the daisy-chains for the bus grant and interrupt acknowledge signals are continuous from slot#1** to the slot in which the MIDAS is installed.

Installation into slot#1 of a VMEbus system is automatically detected, as specified in the VME64 specification. System controller functions are also enabled as a consequence.

WARNING: Do not install the board in a powered system!

Power consumption

WARNING: Due to its power consumption, the MIDAS board requires forced air cooling for reliable operation. Operation on extender boards is not recommended.

Model	Memory Size	Operating Mode	Power Consumption typ. @ +5V
MIDAS-220Y/M	32MB	Both processors are in the idle state.	14.5W
MIDAS-220 Y/M	32MB	One processor transferring data across VME and Raceway, the other processor idle	14.7W
MIDAS-220 Y/M	32MB	One processor transferring data across VME and Raceway, the other processor performing an extended selftest	15.2W
MIDAS-220 SR	128MB	Both processors are in the idle state.	15.0W
MIDAS-220 SR	128MB	One processor transferring data across VME and Raceway, the other processor idle	15.7W
MIDAS-220 SR	128MB	One processor transferring data across VME and Raceway, the other processor performing an extended selftest	17.0W
MIDAS-120	32MB	Processor Idle	10.5W
MIDAS-120	32MB	Processor running selftest	11.5W
MIDAS-120	32MB	Processor transferring data over VME	11.5W

Table 1. Power Consumption

Configuration Switch & Jumpers

The MIDAS has a large number of configuration registers, which need to be initialized before the board is operational. Most registers are normally initialized by an i960®RD, but in some applications, an external host processor can perform this initialization, or parts of it, over VMEbus or RACEway. To allow this, power up options in some MIDAS components are loaded from jumpers or a DIPswitch.

A detailed description of each jumper is included in the relevant part of the "Functional Description" section.

Functional Description

i960®RD and its Surroundings

As indicated in Figure 8, each i960®RD on MIDAS has its own private FLASH memory and RS-232 interface, in addition to its own block of DRAM.

Note that the UART and FLASH must be located in memory regions, with an 8-bit bus width.

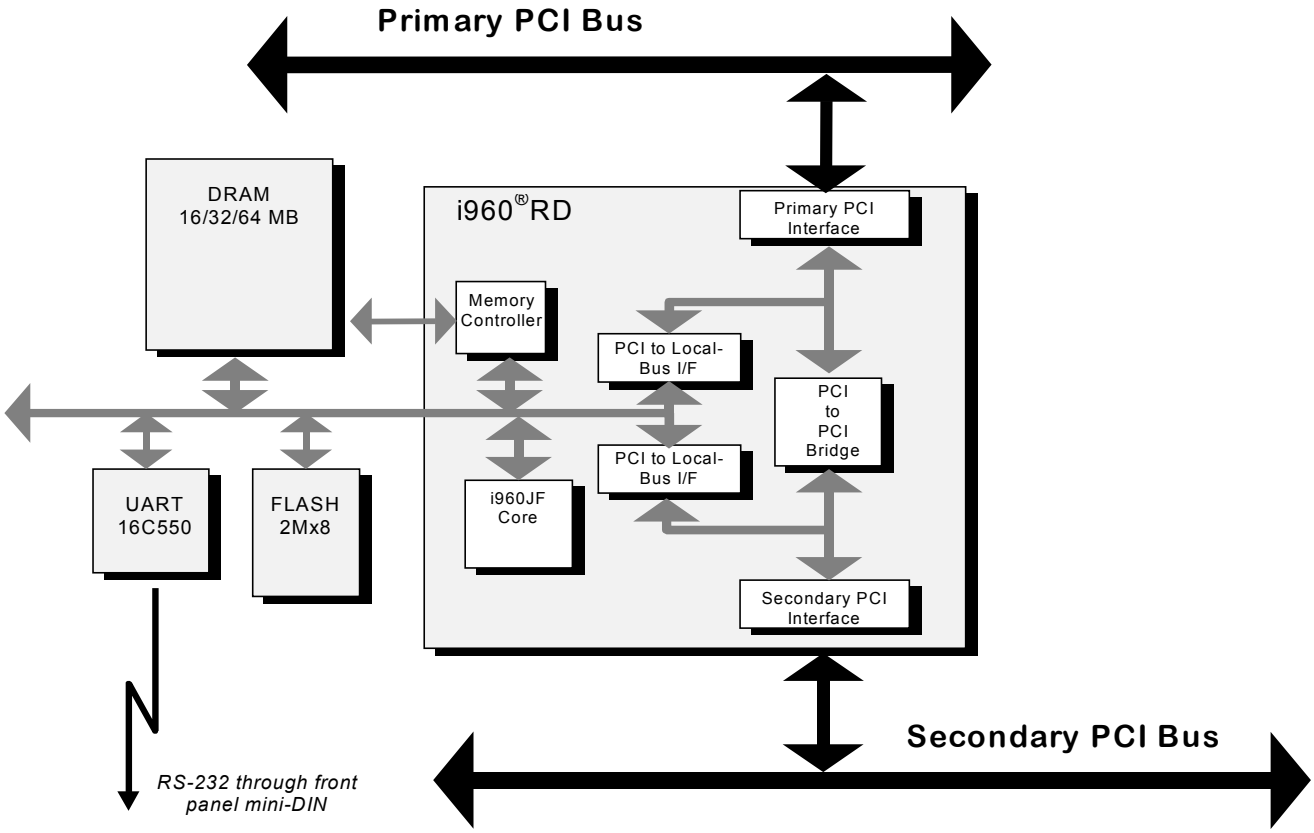


Figure 8. i960®RD with External Devices

i960®RD Address Map

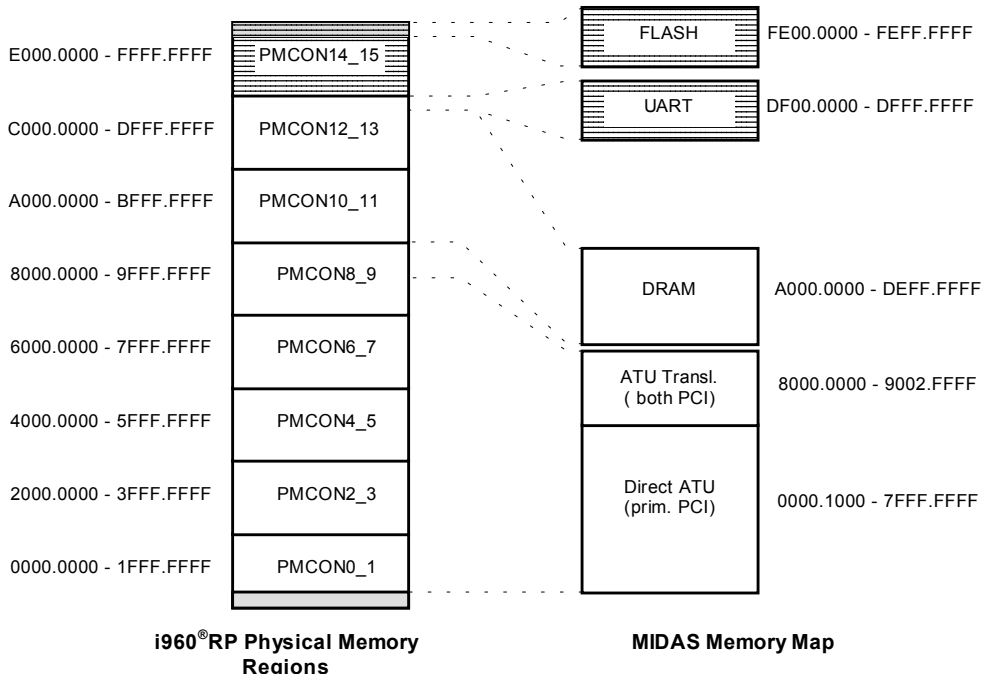


Figure 9. Local Memory Address Space

i960®RD Power-Up Options

The i960®RD monitors a number of pins during reset to define its mode of operation. The most important are:

Secondary PCI Arbiter Enable/Disable:

For both i960®RDs, the secondary PCI arbiter is always enabled.

i960®RD Boot Modes

Two of the power-up options control the boot process of the i960®RD. These two options can be controlled through two jumpers (JP9 & JP14) located in the top right corner of the board.

Jumpers:	i960®RD #1	i960®RD #2
	Processor: Boot directly PCI: Retry until CSR set	Processor: Boot directly PCI: Retry until CSR set
	Processor: Boot directly PCI: Retry until CSR set	Processor: Sleep until CSR set PCI: Accept config. cycles

	Processor: Sleep until CSR set[1] PCI: Accept config. cycles	Processor: Boot directly PCI: Retry until CSR set[2]
	Processor: Sleep until CSR set PCI: Accept config. cycles	Processor: Sleep until CSR set PCI: Accept config. cycles

FLASH Memory

Each i960®RD on the MIDAS-220 has a private 16Mbit FLASH device. A 95ns 2Mx8 flash-file device is used.

Jumpers:	Function:
	JP13 - Removed / JP11 - Inserted: * Locked FLASH blocks are write/erase protected. With some restrictions, blocks can be locked. Ref. i28F016SC/S5 datasheet. ('RD#' pin is at V_{IH} level with this jumper setting)
	JP13 & JP11 – Removed: Entire FLASH is write protected.

Avoid the following jumper settings

	JP13 & JP11 - Inserted: ** No Protection. Entire FLASH can be erased/written. Lock bits can be altered.
	JP13 - Inserted / JP11 - Removed: Indeterminate state - Memory contents is **not** protected

* **Normal Operation** Unlocked flash blocks may be erased and re-programmed.

**** This setting should only be used DURING flash updates of locked flash blocks (Normally only the MIDAS ® Monitor).**

[1] If U44 (a PLD mounted on the backside of the i960®RD#1, on the opposite side of the board) is labeled "1002-A", i960®RD#1 will boot directly.
[2] If U44 is labeled "1002-A", the PCI bus will accept config. cycles.

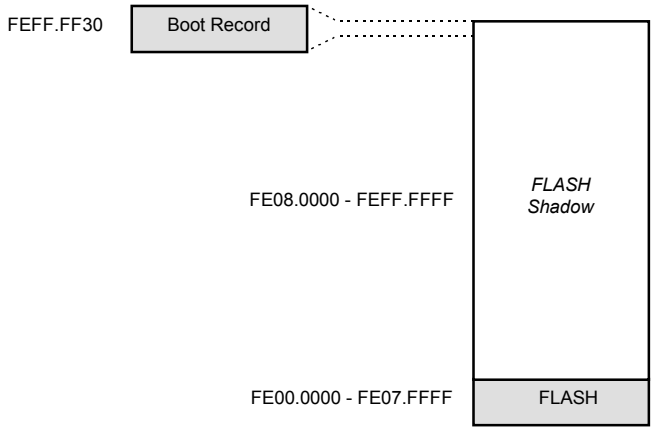


Figure 10. FLASH Boot Address

RS232 Interface

Each i960®RD on the MIDAS-220 has a private UART for serial port communication. The 'TL16C550' is used for this purpose.

Both RS-232 interfaces share a mini-DIN connector for connection through the front panel.

The UART is located at a fixed address in the i960 address space. The base address is 0xDF00.0000. The eight most significant address bits are used for the address decoding, which means that a 16MB window is occupied.

RS232 Cables

MIDAS-100 and MIDAS-200 series boards are shipped with a short converter cable to allow standard D-SUB cables to connect to the non-standard, Mini-DIN MIDAS connector.

The cable for the MIDAS-200 board is a split cable, with one 9 pin D-sub connector for each processor. The shorter cable connects to the UART of i960®RD#1, while the longer cable connects to processor number two.

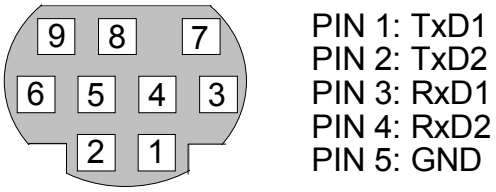


Figure 11. Pin definition for the RS232 female connector mounted on the front panel

Connecting two i960®RDs

To maximize the utilization of the i960®RD's resources, the two processors are connected as shown in Figure 12.

The primary PCI bus interface on i960®RD#1 is connected to the secondary interface on i960®RD#2, and vice versa. The primary PCI bus in MIDAS terms connects to the primary side of i960®RD#1.

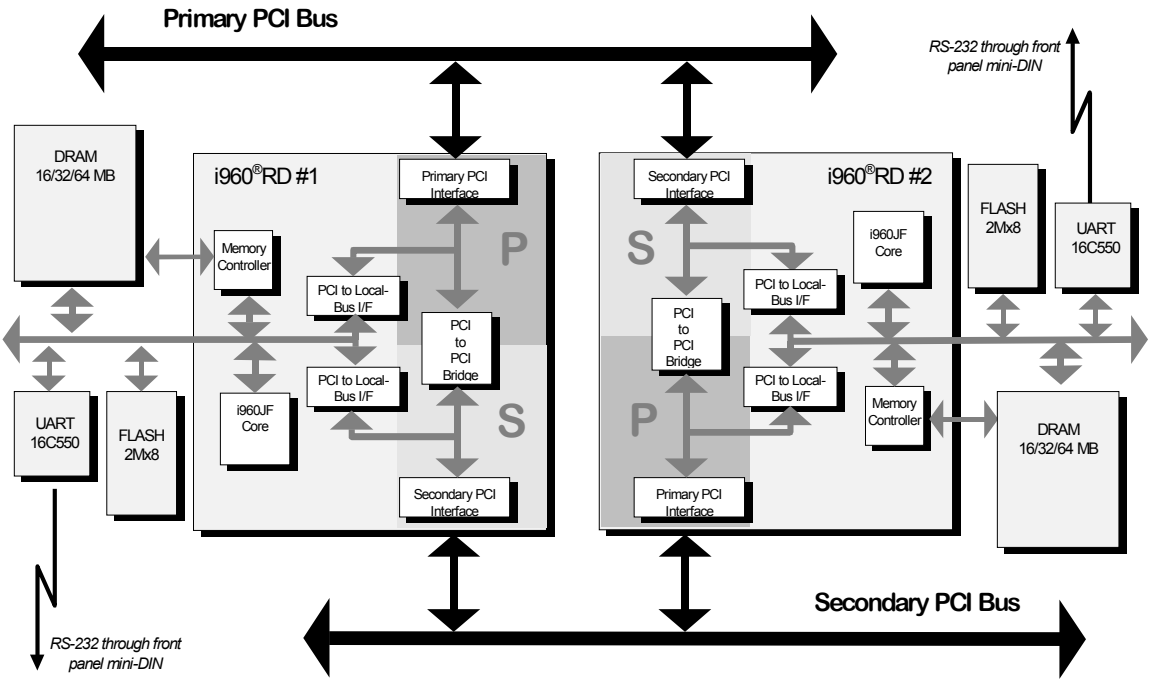


Figure 12. Connection of two i960®RDs on MIDAS-220

Note that for any connection of two i960®RDs in parallel between two PCI busses, like on the MIDAS-220, introduces restrictions on the programming of the address translation units, as well as on the P2P bridge.

The P2P bridge in i960®RD#2 should **not** be enabled, in order to simplify the generation of a valid address map.

VMEbus Interface (Universe II VME-PCI Bridge)

The VME-PCI bridge on MIDAS has a number of configuration registers, which need to be initialized before the bridge is fully operational. One i960®RD processor or a host processor residing on VMEbus does this initialization.

'Universe II' Power-Up Options

The 'Universe II' loads a number of power-up options after power-up or system reset. A detailed description of these options is found in the UNIVERSE II™ USER MANUAL. Table 2 shows how they are controlled on the MIDAS.

Option:	Enabled by	Register	Field	MIDAS
VME Register Access Slave Image	PWR/SYS	VRAI_CTL	EN	*Jumper - JP1*
			VAS	*Jumpers - JP2-3*
		VRAI_BS	BS	*Switch - SW1*
VME CR/CSR Slave Image	PWR/SYS	VCSR_CTL	LAS	I/O Space
		VCSR_TO	TO	'0x00'
		PCI_CSR	MAST_EN	Enabled
PCI Slave Image	PWR/SYS	LSI0_CTL	EN	Disabled
			LAS	Mem. space
			VAS	A16
		LSI0_BS	BS	0x0
		LSI0	BD	0x0
PCI Register Access	PWR/SYS	PCI_BS	SPACE	I/O Space
PCI Bus Size	RST#	MISC_STAT	LCLSIZE	32-bit
Auto-ID	PWR/SYS	MISC_STAT	DY4AUTO	Disabled
			VME64AUTO	*Jumper - JP4*
BI-Mode	PWR/SYS	MISC_CTL	BI	Disabled
SYSFAIL* Assertion	PWR/SYS	VCSR_SET	SYSFAIL	*Jumper - JP5*
		VCSR_CLR	SYSFAIL	
Auto-Syscon Detect	SYS	MISC_CTL	SYSCON	Enabled

Table 2. 'Universe II' Power-Up Options

Auto-Slot ID

Plug&Play

The MIDAS supports the Auto-Slot ID mechanism as defined by the VME64 specification. By use of the daisy-chained IACK signal, CR/CSR space accesses are enabled for one VME slot at a time. Thus, the "Monarch" (host for initialization) is able to recognize installed modules and initialize them to achieve Plug&Play VMEbus systems.

Enabling of the Auto-Slot ID feature causes IRQ2* to be asserted, this may cause problems in systems containing VMEbus boards which do not support the Auto-Slot ID feature. In such cases, the Auto-Slot ID feature must be disabled.

Configuration ROM

To support the Auto-slot ID protocol, the MIDAS board implements a configuration ROM. This ROM is implemented as a PCI bus slave-only device, which responds in PCI I/O space using subtractive decoding.

Note that parity is not generated when reading the MIDAS configuration ROM. Parity errors should therefor be disregarded when reading these locations. In the power-up state of the Universe II VME-PCI bridge, parity errors are disregarded.

The configuration ROM is disabled by any write, in PCI Configuration Space, with address bit 30 set to 1. This way boot software may “remove” the CROM from PCI I/O space to allow another subtractive decoding agent, or to avoid parity errors on reads from “non present devices”.

MIDAS-x20 Configuration ROM

CROM	**Offset: 03 (VME CR/CSR Space)**

Offset	ROM Value	Description
03		Checksum. Eight bit 2s complement binary checksum (CR bytes 03-7F).
07	00	Length of ROM to be checksummed. (MSB)
0B	00	Length of ROM to be checksummed. (NMSB)
0F	1F	Length of ROM to be checksummed. (LSB)
13	81	CR Data access width (0x81 = D08(EO), every forth byte)
17	81	CSR Data access width (0x81 = D08(EO), every forth byte)
1B	01	CR/CSR Space Specification ID (0x01 = VME64 - 1994 version)
1F	43	'C'. Used to identify valid CR.
23	52	'R'. Used to identify valid CR.
27	00	24 bit IEEE Assigned Manufacturers ID.
2B	60	0x006046 = VMETRO
2F	46	
33	00	Board ID (VMETRO Assigned)
37	00	0x00020020 = MIDAS-20R/50R
3B	02	0x00000120 = MIDAS-120xx/150xx
3F	20	0x00000220 = MIDAS-220xx/250xx
43	00	Revision ID (VMETRO Assigned)
47	00	Example 0x0000B001 = PCB Rev: B, ECO-level:1
4B	B0	
4F	01	
53	00	Pointer to null terminated ASCII string. Revision ID (VMETRO Assigned)
57	00	0x000000 = No string
5B	00	
5F-7B	00	Reserved for future use
7F	01	Program ID Code. 0x01 = No program, ID ROM only.

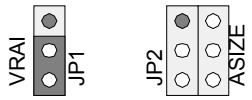
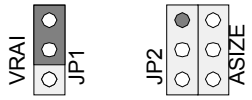
Table 3. MIDAS Configuration ROM

Switch & Jumper Descriptions

MIDAS has a DIPswitch and a number of jumpers for board configuration. For easy identification, their functions are indicated with silk-screen text on the printed circuit board.

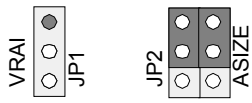
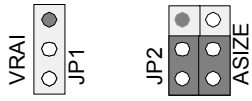
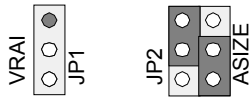
VME Register Access Image - ENABLE/DISABLE

The *VME Register Access Image (VRAI)* permits accesses from VMEbus to the VME-PCI bridge internal registers at power-up. Unless the Auto-Slot ID protocol (which uses its own slave image) is used, this slave image must be enabled to allow initialization from VMEbus.

Jumpers:	**Function:**
	JP1 - DOWN: VME Register Access Image DISABLED.
	JP1 - UP: VME Register Access Image ENABLED.

VME Register Access Image - ADDRESS SIZE

The *VME Register Access Image* can accept A16, A24 or A32 AM codes, depending on the positioning of the 'A SIZE' jumpers (JP2 & JP3). At power-up, this slave image accepts AM codes for: Supervisor, User, Data and Code.

Jumpers:	**Function:**
	JP2 - UP/UP: VME Register Access Image A24.
	JP2 - DOWN/DOWN: VME Register Access Image A32.
	JP2 - UP/DOWN: VME Register Access Image A16.

VME Register Access - BASE ADDRESS

The DIPswitch is used to define the base address for the *VME Register Access Image.* The size of this image is fixed to 4KB.

VRAI Addr. Size	BS[31:24]	BASE ADDRESS BS[23:16]	BS[15:12]
A16	N.A.	N.A.	From switch (4 MSB)
A24	N.A.	From switch	0x00
A32	From switch	0x00	0x00

Table 4. VRAI Base Address Definition

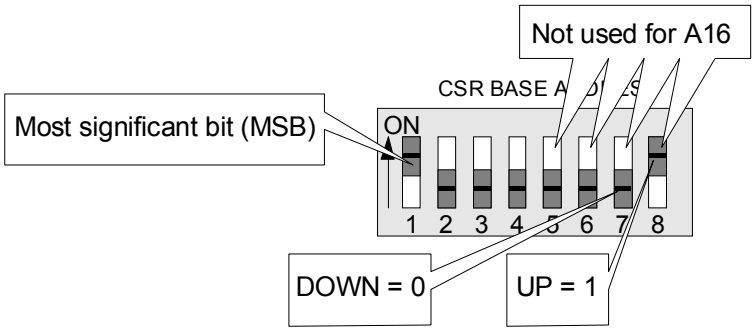
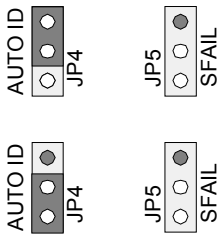


Figure 13. DIP Switch Details. (Shown switch value: 0x81)

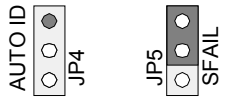
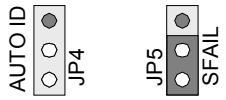
VME64 Auto-Slot ID - ENABLE/DISABLE

By using new address space for CR/CSR accesses, the VME64 specification defines a method for implementing Plug&Play on the VMEbus, called Auto-Slot ID. 'JP6' controls the enabling of the VME64 Auto-Slot ID mechanism.

Jumpers:	**Function:**	
	JP4 - UP:	VME64 AUTO-SLOT ID ENABLED.
	JP4 - DOWN:	VME64 AUTO-SLOT ID DISABLED.

SYSFAIL Assertion - ENABLE/DISABLE

The VMEbus interface provides the option to assert the VMEbus signal, SYSFAIL, during power up. When this feature is enabled, the MIDAS board will assert SYSFAIL on the VMEbus until the boot/initialization software has completed the power-up selftest and configuration routines. SYSFAIL is then de-asserted to signal to the VMEbus system that the board is ready to operate. 'JP5' controls the enabling of this mechanism.

Jumpers:	Function:	
	JP5 - UP:	SYSFAIL ASSERTION DISABLED
	JP5 - DOWN:	SYSFAIL ASSERTION ENABLED

SYSRST Assertion - ENABLE/DISABLE*

If this jumper is removed, pushing of the reset button will perform both a local and a global reset. In order to perform only a local reset (without assertion of the SYSRST* signal on the VMEbus), the jumper should be inserted.

Jumpers	Function:	
	JP7 - INSERTED:	SYSRST* ASSERTION DISABLED
	JP7 - REMOVED:	SYSRST* ASSERTION ENABLED

When disabling SYSRST* assertion, the Universe II chip is partly reset by the front panel reset button. This may have undesired side effects. The RST DIS jumper is included to support reset isolation in certain development systems.

Reset Button

Pushing the reset button will perform both a local and a global reset.

In order to perform only a local reset (without assertion of the SYSRST* signal on the VMEbus), the soft reset mechanism must be utilized. This can be done from the host, by setting the bit 23 of the Universe II MISC_CTL register (offset 404).

RACEway Interface (PXB RACEway-PCI Bridge)

Some MIDAS models (with -R option) incorporate the PXB chip, a PCI-RACEway bridge developed by Mercury Computer Systems.

Jumper Descriptions

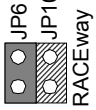
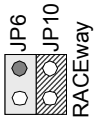
MIDAS has two jumpers for configuration of the RACEway interface. They are located close to the bottom edge of the board, indicated with silk-screen text: RACEway. As explained in the description below, both RACEway jumpers are removed during normal operation.

No EEPROM

When the *No EEPROM* jumper is removed, the PCI-RACEway bridge chip loads its internal registers from a serial EEPROM. This setting should always be used, except for cases where the PROM is blank or corrupted.

If the PROM is corrupted, and this jumper is removed, the PCI-RACEway bridge may reset to a state which causes the MIDAS board (in worst case the entire system) to hang.

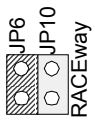
If the *No EEPROM* jumper is inserted, the PCI-RACEway bridge reset state is independent of PROM contents. This setting is normally used for the initial programming of the PROM or if the board is plugged into a non-RACEway slot. Note that before programming the PROM, the PXB must be set in *bridge mode* (ref. PXB description).

Jumpers:	Function:	
	JP6 - Inserted:	PXB registers not loaded from EEPROM.
	JP6 - Removed:	PXB registers loaded from EEPROM.

Reset from X

When the *Reset from X* jumper is removed, the PCI-RACEway bridge receives reset from PCI bus (i.e. MIDAS reset circuitry), and drives reset to the RACEway interlink. This setting should always be used.

Inserting this jumper may be destructive for the MIDAS board, and is likely to cause system malfunction.

Jumpers:	Function:
	JP10 - Removed: This setting should always be used.

PCI Bus Details

Arbitration

Secondary PCI bus.

The PCI bus arbitration unit in the i960®RD # 1 is responsible for arbitration on the secondary PCI bus, for all MIDAS models.

The secondary bus arbiter of the i960®RD supports up to six secondary bus masters, plus its own secondary bus interface. Each request can be disabled or programmed to one of three priority levels. A memory mapped control register, programmed by the application software, sets the priorities for each of the bus masters. Each priority level is handled in a round-robin fashion. The three levels define a low, medium and high priority. Using the round-robin mechanism ensures there is a winner for each priority level. To enforce the concept of fairness, a slot is reserved for the winner of each priority level (except the highest) in the next highest priority. When the winner of a priority level is not granted the bus, during that particular arbitration sequence, it is promoted to the next highest level of priority. Once its bus ownership is removed, the device is reset to its initially programmed priority and may start arbitration once again.

For more information on the arbitration scheme, and on the programming of the arbiter, please refer to the i960®RD User's Manual.

Assignments to Arbiter (RD#1) Device Numbers

PCI DEVICE	DEVICE NUMBER
PMC # 2b	Device 0
PMC # 3	Device 2
PMC # 4	Device 4
PMC # 5	Device 0
i960®RD # 2	Device 1

Table 5. Secondary PCI Bus

Primary PCI bus

The arbitration of the primary PCI bus is handled in three different ways depending on MIDAS model.

MIDAS-200 Series

The PCI bus arbitration unit in the i960®RD # 2 is responsible for arbitration on the primary PCI bus for all MIDAS-200 series models. For more information, refer to the brief description *Secondary PCI Bus (*page 25) or the i960 User's manual.

Assignments to Arbiter (RD#2) Device Numbers

PCI DEVICE	DEVICE NUMBER
PMC # 1	Device 1
PMC # 2	Device 2
i960®RD # 1	Device 0
PXB (PCI-RACEway)	Device 4
VME-PCI BRIDGE	Device 3

Table 6. Primary PCI Bus - Arbiter: RD#1

MIDAS-100 Series with RACEway Interface

For MIDAS models with RACEway interface, and with only one or none i960®RDs, the PCI bus arbitration unit in the PXB is responsible for arbitration on the primary PCI bus. This arbiter is programmable, and uses a round-robin arbitration scheme with two priority levels.

Assignments to Arbiter (PXB) Req/Gnt Pairs

PCI DEVICE	Req/Gnt Pairs
PMC # 1	Req1/Gnt1
PMC # 2	Req2/Gnt2
i960®RD # 1	Req4/Gnt4
VME-PCI BRIDGE	Req3/Gnt3

Table 7. Primary PCI Bus - Arbiter: PXB

Other MIDAS-1x0 Models

For MIDAS-120 and -150 models without RACEway interface, the arbiter for the primary PCI bus is implemented in a PLD. This arbiter uses a single level round-robin arbitration scheme.

'IDSEL' Generation

PCI bus uses a separate address space for initialization called *Configuration Space*. This address space uses a geographic addressing signal, IDSEL, to select target for all transactions. The standard way of assigning IDSEL to PCI devices & boards is to connect the IDSEL pin of each device/board to a unique AD[ ] bit.

This is also how IDSEL is generated on MIDAS. Table 8 & Table 9 shows IDSEL assignments.

PCI DEVICE/BOARD	IDSEL	PCI ADDRESS FOR CONFIG. CYCLE	VME BASE ADDRESS FOR PCI CONFIG. CYCLE
PMC # 1	pAD[16]	0x00010XXX	0xYYZZ2800
PMC # 2	pAD[17]	0x00020XXX	0xYYZZ3000
i960®RD # 1	pAD[18]	0x00040XXX	0xYYZZ3800
PXB (PCI-RACEway)	pAD[19]	0x00080XXX	0xYYZZ4000
i960®RD # 2	pAD[20]	0x00100XXX	0xYYZZ4800
VME-PCI BRIDGE	pAD[31]	0x80000XXX	

'ZZ' = PCI Bus Number (as defined in Universe II MAST_CTL register)
'YY' = VME base address for slave image

Table 8. IDSEL Assignments for Primary PCI Bus

PCI DEVICE/BOARD	IDSEL	PCI ADDRESS FOR CONFIG. CYCLE	VME BASE ADDRESS FOR PCI CONFIG. CYCLE
PMC # 3	sAD[16]	0x00010XXX	0xYYWW0000
PMC # 4	sAD[17]	0x00020XXX	0xYYWW0800
PMC # 5	sAD[18]	0x00040XXX	0xYYWW1000
PMC # 2b	sAD[19]	0x00080XXX	0xYYWW1800
i960®RD # 1	sAD[21]	0x00200XXX	0xYYWW2800
i960®RD # 2	sAD[20]	0x00100XXX	0xYYWW2000

'WW' = PCI Bus Number for secondary bus (as defined in i960®RD)
'YY' = VME base address for slave image

Table 9. IDSEL Assignments for Secondary PCI Bus

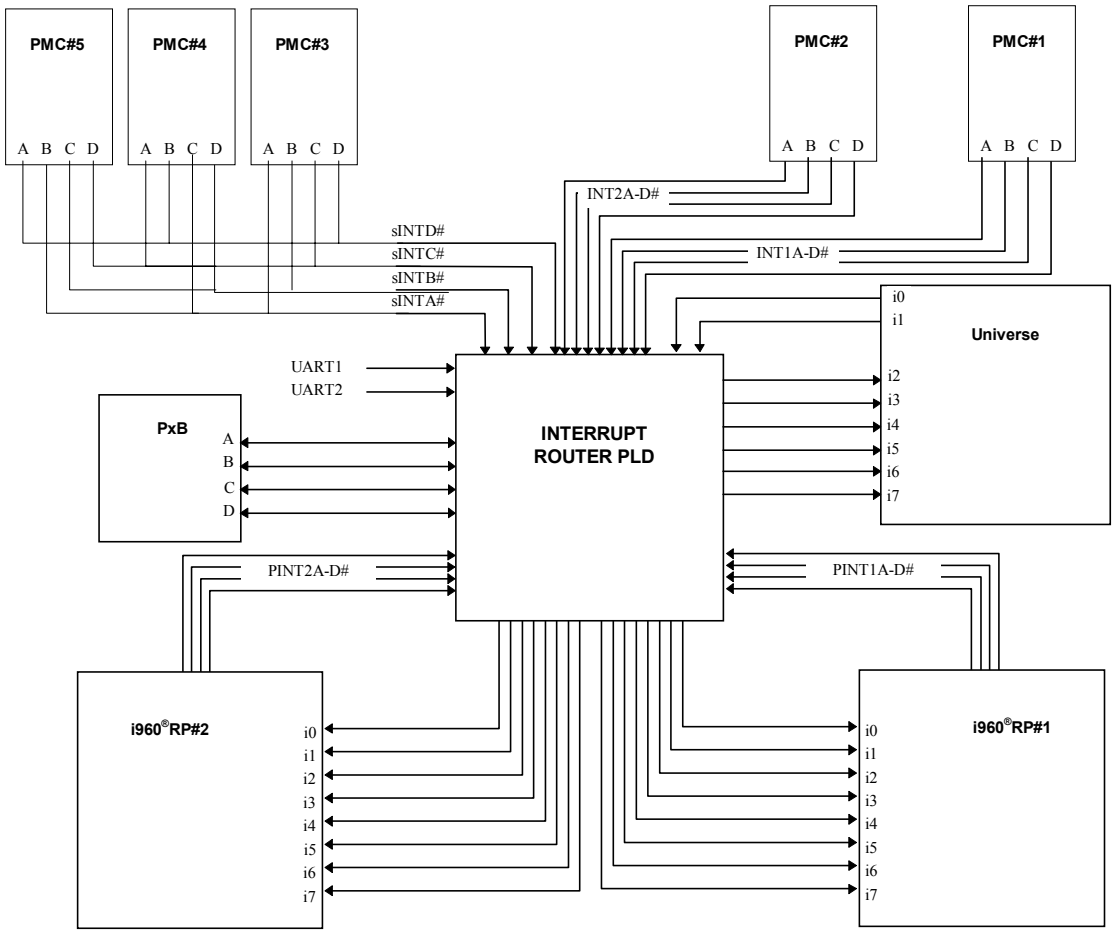
Subtractive Decoding Agent

The CR/CSR PLD utilizes subtractive decoding in the PCI bus *I/O Space*. No other PCI devices are allowed to do the same. Subtractive decoding in *Memory Space* may be used.

For further information, please refer to description of Configuration ROM.

Interrupt Routing

The MIDAS board has a number of interrupt sources and destinations. In order to provide a flexible interrupt routing scheme, which also allows customization, all interrupts are routed through a JTAG programmable PLD.



Interrupt Mode Selection

Interrupt mode is selected by i960®RD #1 through two register bits in its UART.

Register Name: MCR	Size: 8 bits	Address: 0xDF00.0004

Bits	Function						
7:0	0	0	AFE	Loop	INT_MODE	RTS	DTR

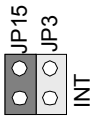
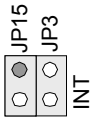
MCR Description

Name	Type	Reset State	Function
INT_MODE	R/W	00	Interrupt Mode: 00 = Interrupt Mode 0 01 = Interrupt Mode 1 10 = Interrupt Mode 2

Table 10. Interrupt Mode Selection

Interrupt Jumpers

Currently these jumpers are only used for MIDAS-xxxR boards to control whether the PCI-RACEway bridge should be treated as an interrupt source or a destination.

Jumpers:	Function:
	JP15/3 - IN/OUT: PXB Treated as interrupt destination.
	Other combinations: PXB Treated as interrupt source.

This jumper setting must be used if a MIDAS board with RACEway (-R) option is plugged into a non-RACEway slot.

Interrupt Routing Tables

The tables below show how the interrupts are routed for the three defined interrupt modes. All interrupt sources and destinations are present for a MIDAS-250R only. For other models, the routing scheme is the same, but sources (and destinations) not present should be ignored.

Interrupt Mode 0:

i960®RD #1&2	Interrupt Pin - Source																													
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				Universe II		UART	PXB				i960RD (other)			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1		A	B	C	D	A	B	C	D
XINT0#									A					C				B									D			C	
XINT1#										B					D				C						A					D	
XINT2#											C		A						D							B		A			
XINT3#												D		B			A										C		B		
XINT4#	A		C			B		D																							
XINT5#		B		D	A		C																								
XINT6#																					0										
XINT7#																						1	X								

Universe II	Interrupt Pin - Source																													
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				i960RD#1				i960RD#2				PXB			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
LINT2#	A							D																								
LINT3#				D	A																											
LINT4#			C						A					C				B			A					C					C	
LINT5#							C			B					D				C			B					D		A			
LINT6#		B									C		A						D				C		A						D	
LINT7#						B						D		B			A						D			B				B		

PXB	Interrupt Pin - Source																													
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				i960RD#1				i960RD#2				Universe II			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1		
INTA#			C					D	A					C				B			A					C			0			
INTB#	A						C			B					D				C			B					D					
INTC#		B			A						C		A						D				C		A						
INTD#				D		B						D		B			A						D			B				1		

Interrupt Mode 1:

i960®RD#1	Interrupt Pin - Source																											
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				Universe II		UART	PXB				RD#2			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1		A	B	C	D	A	B	C	D
XINT0#	A		C																												
XINT1#					A		C																								
XINT2#		B		D					A					C				B						A							
XINT3#						B		D			C		A						D						B						
XINT4#												D		B			A						X							
XINT5#										B					D			C			0										
XINT6#	*Reserved i960®RD DMA Controller*																														
XINT7#	*Reserved i960®RD Message Unit*																														

i960®RD#2	Interrupt Pin - Source																											
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				Universe II		UART	PXB				RD#1			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1		A	B	C	D	A	B	C	D
XINT0#					A		C																								
XINT1#	A		C																												
XINT2#						B		D			C		A						D						C						
XINT3#		B		D					A					C				B									D				
XINT4#												D		B			A						X							
XINT5#										B					D			C			1										
XINT6#	*Reserved i960®RD DMA Controller*																														
XINT7#	*Reserved i960®RD Message Unit*																														

Universe II	Interrupt Pin - Source																															
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				i960RD#1				i960RD#2				PXB			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
LINT2#																													A			
LINT3#																														B		
LINT4#																					A						C					
LINT5#																						B						D				
LINT6#																						C			A							
LINT7#																							D			B						

PXB	Interrupt Pin - Source																									
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				i960RD#1				i960RD#2				Universe II	
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1
INTA#																					A						C			
INTB#																					B						D			
INTC#																					C				A					
INTD#																					D				B					

Interrupt Mode 2:

i960®RD#1	Interrupt Pin - Source																													
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				Universe II		uart		PXB				RD#2	
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1		A	B	C	D	A	B	C	D
XINT0#									A					C			B								D			C		
XINT1#									B					D			C							A				D		
XINT2#									C				A				D							B				A		
XINT3#									D				B				A							C				B		
XINT4#																					0									
XINT5#																					1									
XINT6#	A		C																											
XINT7#		B		D																			X							

i960®RD#2	Interrupt Pin - Source																													
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				Universe II		uart		PXB				RD#1	
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0	1		A	B	C	D	A	B	C	D
XINT0#									A					C			B								D			C		
XINT1#									B					D			C							A				D		
XINT2#									C				A				D							B				A		
XINT3#									D				B				A							C				B		
XINT4#																					1									
XINT5#																					0									
XINT6#					A		C																							
XINT7#						B		D															X							

UNIVERSE II	INTERRUPT PIN - SOURCE																															
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				I960RD#1				I960RD#2				PXB			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
LINT2#	A							D																								
LINT3#				D	A																											
LINT4#			C						A						C				B		A						C				C	
LINT5#							C			B						D			C			B						D	A			
LINT6#		B									C		A							D			C		A						D	
LINT7#						B						D		B			A							D		B				B		

PXB	INTERRUPT PIN - SOURCE																															
	PMC#1				PMC#2				PMC#3/ PMC#2b				PMC#4				PMC#5				I960RD#1				I960RD#2				UNIVERSE II			
Interrupt pin	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	0		1	
INTA#									A						C				B													
INTB#										B						D			C													
INTC#											C		A							D												
INTD#												D		B			A															

Appendix I: PMC I/O Routing

PMC I/O Routing Scheme

PMC2 J4	PMC1 J4	P2 row A
34	2	1
36	4	2
38	6	3
40	8	4
42	10	5
44	12	6
46	14	7
48	16	8
50	18	9
52	20	10
54	22	11
56	24	12
58	26	13
60	28	14
62	30	15
64	32	16
	34	17
	36	18
	38	19
	40	20
	42	21
	44	22
	46	23
	48	24
	50	25
	52	26
	54	27
	56	28
	58	29
	60	30
	62	31
	64	32

P2 row C	PMC1 J4	PMC2 J4
1	1	33
2	3	35
3	5	37
4	7	39
5	9	41
6	11	43
7	13	45
8	15	47
9	17	49
10	19	51
11	21	53
12	23	55
13	25	57
14	27	59
15	29	61
16	31	63
17	33	
18	35	
19	37	
20	39	
21	41	
22	43	
23	45	
24	47	
25	49	
26	51	
27	53	
28	55	
29	57	
30	59	
31	61	
32	63	

Appendix II: Universe II Configuration Examples

General Information

Note that the 'Universe II' PCI-VME Bridge, performs byte swapping of the data lanes on all transactions between VMEbus and PCI bus. This is also the case for accesses to the internal registers. The internal register bank is located on the 'PCI side' of the byte swapping. This means that when registers are read or written from the VMEbus, all bytes are shuffled (compared to the bit numbering used in the Universe II User Manual).

VMEbus Slave Images

PCI Master Enable

In addition to the configuration registers for the VMEbus slave images, one control register bit is essential for mapping VMEbus cycles to PCI bus cycles through the Universe II. The PCI master enable ('BM') bit located in the PCI_CSR register space (offset: 0x004). This bit is set as default after power up.

Some VMEbus Slave Image examples are shown in Figure 14.

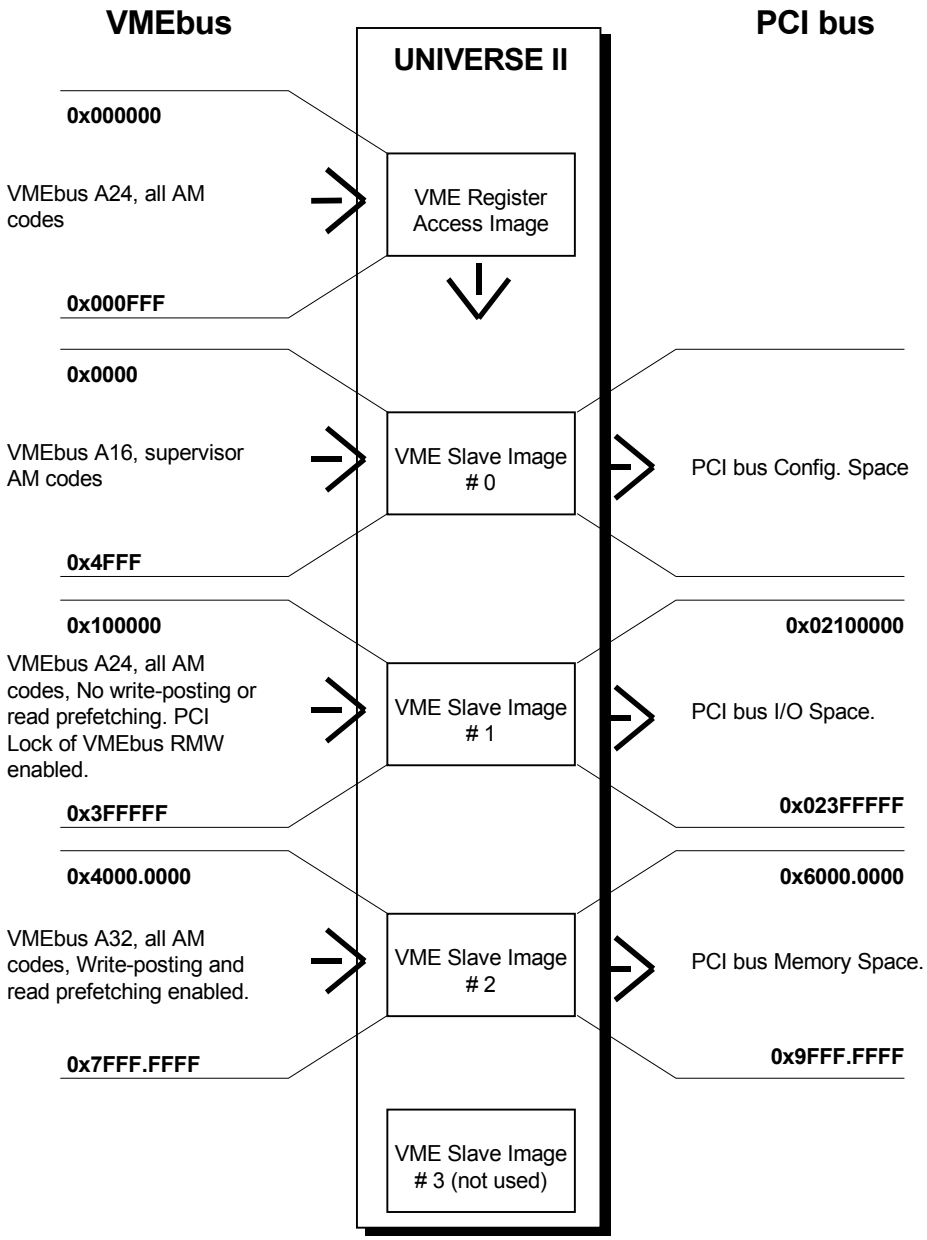


Figure 14. Configuration Example for VMEbus Slave Images

VMEbus Register Access Image

In this configuration example, the VMEbus Register Access Image is set up by use of the DIPswitch and jumpers.

Action:	Result:
Locate JP1 in its upper position.	*VME_RAI is enabled.*
Locate JP2&3 in their upper positions.	*VME_RAI is mapped in A24 address space.*
Set the DIP switch with all switches pointing down.	*VME_RAI base address is set to 0x000000.*
Locate JP4 in its lower position.	*Disable Auto-slot ID protocol.*

Table 11. VME_RAI Setup.

With the jumper settings described above, the Universe II will power up in a state where VMEbus accesses with A24 AM codes, in the address range 0x0-0xFFF, will map into Universe II registers.

The VME_RAI will be utilized to set up the VMEbus slave images described below.

VMEbus Slave Image 0

In this configuration example, the VMEbus Slave Image 0 is set up to map A16 supervisory accesses, in the address range 0x0-0x4FFF, from VMEbus to Configuration Cycles on the PCI bus.

Write from VME	PCI Data [1)]	Result:
D:0x0000.0000 to A:0x000F04	0x0000.0000	*Base Address set to 0x000000*
D:0x0050.0000 to A:0x000F08	0x0000.5000	*Bound Address set to 0x005000*
D:0x0000.0000 to A:0x000F0C	0x0000.0000	*Translation Offset set to 0x000000*
D:0x0200.E080 to A:0x000F00	0x80E0.0002	*Enable Image, VAS=A16, LAS=Config Space, PGM=both, SUPER=Supervisor, other options disabled.*

[1)] This column shows write data for configuration from PCI

Table 12. VME Slave Image 0 - Setup

VMEbus Slave Image 1

The VMEbus Slave Image 1 is set up to map A24 accesses, in the address range 0x100000-0x3FFFFF from VMEbus to I/O Cycles on the PCI bus, with PCI addresses starting from 0x02100000.

Write from VME	PCI Data [1)]	Result:
D:0x0000.1000 to A:0x000F18	0x0010.0000	*Base Address set to 0x100000*
D:0x0000.4000 to A:0x000F1C	0x0040.0000	*Bound Address set to 0x400000*
D:0x0000.0002 to A:0x000F20	0x0200.0000	*Translation Offset set to 0x2000000*
D:0x4100.F180 to A:0x000F14	0x80F1.0041	*Enable Image, VAS=A24, LAS=I/O Space, PGM=both, SUPER=both, LLRMW=enabled, other options disabled.*

[1)] This column shows write data for configuration from PCI

Table 13. VME Slave Image 1 - Setup.

VMEbus Slave Image 2

VMEbus Slave Image 2 is set up to map A32 accesses, in the address range 0x4000.0000-0x7FFF.FFFF, from VMEbus to Memory Cycles on the PCI bus, with PCI addresses starting from 0x6000.0000. Write posting and read prefetching is enabled.

Write from VME	PCI Data [1)]	Result:
D:0x0000.0040 to A:0x000F2C	0x4000.0000	*Base Address set to 0x4000.0000*
D:0x0000.0080 to A:0x000F30	0x8000.0000	*Bound Address set to 0x8000.0000*
D:0x0000.0020 to A:0x000F34	0x2000.0000	*Translation Offset set to 0x2000.0000*
D:0x0000.F2E0 to A:0x000F28	0xE0F2.0000	*Enable Image, VAS=A32, LAS=Mem. Space, PGM=both, SUPER=both, PWEN&PREN= enabled, other options disabled.*

[1)] This column shows write data for configuration from PCI

Table 14. VME Slave Image 2 - Setup.

Initialization Sequence

By performing the list of cycles shown in the table below, the mapping for this configuration example is achieved.

Write from VME	PCI Data [1)]	Result:
D:0x0000.0000 to A:0x000F04	0x0000.0000	*VSI_0: Base Address set to 0x000000*
D:0x0050.0000 to A:0x000F08	0x0000.5000	*VSI_0: Bound Address set to 0x005000*
D:0x0000.0000 to A:0x000F0C	0x0000.0000	*VSI_0: Translation Offset set to 0x000000*
D:0x0200.E080 to A:0x000F00	0x80E0.0002	*VSI_0: Enable Image, VAS=A16, LAS=Config Space, PGM=both, SUPER=Supervisor, other options disabled.*
D:0x0000.1000 to A:0x000F18	0x0010.0000	*VSI_1: Base Address set to 0x100000*
D:0x0000.4000 to A:0x000F1C	0x0040.0000	*VSI_1: Bound Address set to 0x400000*
D:0x0000.0002 to A:0x000F20	0x0200.0000	*VSI_1: Translation Offset set to 0x2000000*
D:0x4100.F180 to A:0x000F14	0x80F1.0041	*VSI_1: Enable Image, VAS=A24, LAS=I/O Space,*

		PGM=both, SUPER=both, LLRMW=enabled, other options disabled.
D:0x0000.0040 to A:0x000F2C	0x4000.0000	*VSI_2: Base Address set to 0x4000.0000*
D:0x0000.0080 to A:0x000F30	0x8000.0000	*VSI_2: Bound Address set to 0x8000.0000*
D:0x0000.0020 to A:0x000F34	0x2000.0000	*VSI_2: Translation Offset set to 0x2000.0000*
D:0x0000.F2E0 to A:0x000F28	0xE0F2.0000	*VSI_2: Enable Image, VAS=A32, LAS=Mem. Space, PGM=both, SUPER=both, PWEN&PREN= enabled, other options disabled.*

[1)] This column shows write data for configuration from PCI

Table 15. Initialization Sequence for VMEbus Slave Image Config. Example.

PCI Slave Images

The VME_RAI, described in the ’VMEbus Slave Images’ section, is also utilized to set up PCI slave images in the examples below.

PCI Target Enable - Memory & I/O Space

In addition to the configuration registers for the PCI slave images, two control register bits are essential for mapping PCI bus cycles to VMEbus cycles through the Universe II. The PCI Target Memory Enable ('MS') and Target IO Enable ('IOS') bits, located in the PCI_CSR register (offset: 0x004), must be set to allow the Universe II to respond to PCI memory and I/O commands.

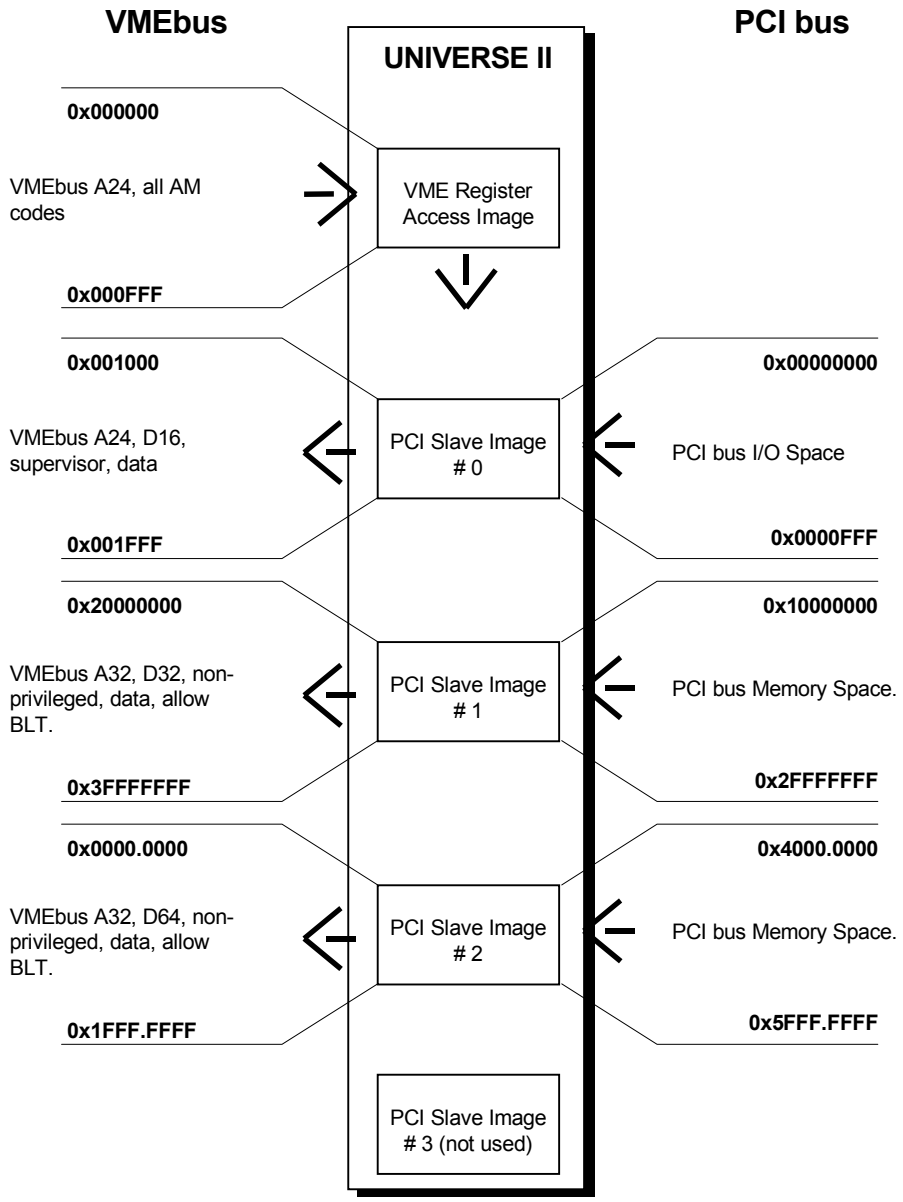


Figure 15. Configuration Example for PCI Slave Images

PCI Slave Image 0

In this configuration example, the PCI Slave Image 0 is set up to map PCI I/O Space transactions, in the address range 0x0-0xFFF, to A24, D16 VMEbus cycles, in the address range 0x1000-0x1FFF.

Write from VME	PCI Data [1)]	Result:
D:0x0000.0000 to A:0x000104	0x0000.0000	*Base Address set to 0x0000.0000*
D:0x0010.0000 to A:0x000108	0x0000.1000	*Bound Address set to 0x0000.1000*
D:0x0010.0000 to A:0x00010C	0x0000.1000	*Translation Offset set to 0x0000.1000*
D:0x0110.4180 to A:0x000100	0x8041.1001	*Enable Image, VAS=A24, VDW=D16, LAS=I/O Space, PGM=data, SUPER=supervisor, other options disabled.*

[1)] This column shows write data for configuration from PCI

Table 16. PCI Slave Image 0 Setup.

PCI Slave Image 1

In this configuration example, the PCI Slave Image 1 is set up to map PCI Memory Space transactions, in the address range 0x1000.0000-0x2FFF.FFFF, to A32, D32 VMEbus cycles, in the address range 0x2000.0000-0x3FFF.FFFF.

Write from VME	PCI Data [1)]	Result:
D:0x0000.0010 to A:0x000118	0x1000.0000	*Base Address set to 0x1000.0000*
D:0x0000.0030 to A:0x00011C	0x3000.0000	*Bound Address set to 0x3000.0000*
D:0x0000.0010 to A:0x000120	0x1000.0000	*Translation Offset set to 0x1000.0000*
D:0x0001.82C0 to A:0x000114	0xC082.0100	*Enable Image, VAS=A32, VDW=D32, LAS=Mem. Space, PGM=data, SUPER=non-priv, Posted Write enabled, BLT allowed.*

[1)] This column shows write data for configuration from PCI

Table 17. PCI Slave Image 1 Setup.

PCI Slave Image 2

PCI Slave Image 2 is set up to map PCI Memory Space transactions, in the address range 0x4000.0000-0x5FFF.FFFF to A32, D64 VMEbus cycles, in the address range 0x0000.0000-0x1FFF.FFFF.

Write from VME	PCI Data [1)]	Result:
D:0x0000.0040 to A:0x00012C	0x4000.0000	*Base Address set to 0x4000.0000*
D:0x0000.0060 to A:0x000130	0x6000.0000	*Bound Address set to 0x6000.0000*
D:0x0000.00C0 to A:0x000134	0xC000.0000	*Translation Offset set to 0xC000.0000*
D:0x0001.C2C0 to A:0x000128	0xC0C2.0100	*Enable Image, VAS=A32, VDW=D64, LAS=Mem. Space, PGM=data, SUPER=non-priv, Posted Write enabled, BLT allowed.*

[1)] This column shows write data for configuration from PCI

Table 18. PCI Slave Image 2 Setup.

Initialization Sequence

By performing the list of cycles shown in the table below, the mapping for this configuration example is achieved.

Write from VME	PCI Data [1)]	Result:
D:0x0700.8002 to A:0x000004	0x0200.0007	*PCI Target Enable bits set. (this write cycle also sets the PCI master enable bit if it is disabled, ref. VMEbus Slave Image section).*
D:0x0000.0000 to A:0x000104	0x0000.0000	*LSI_0: Base Address set to 0x0000.0000*
D:0x0010.0000 to A:0x000108	0x0000.1000	*LSI_0: Bound Address set to 0x0000.1000*
D:0x0010.0000 to A:0x00010C	0x0000.1000	*LSI_0: Translation Offset set to 0x0000.1000*
D:0x0110.4180 to A:0x000100	0x8041.1001	*LSI_0: Enable Image, VAS=A24, VDW=D16, LAS=I/O Space, PGM=data, SUPER=supervisor, other options disabled.*
D:0x0000.0010 to A:0x000118	0x1000.0000	*LSI_1: Base Address set to 0x1000.0000*
D:0x0000.0030 to A:0x00011C	0x3000.0000	*LSI_1: Bound Address set to 0x3000.0000*
D:0x0000.0010 to A:0x000120	0x1000.0000	*LSI_1: Translation Offset set to 0x1000.0000*
D:0x0001.82C0 to A:0x000114	0xC082.0100	*LSI_1: Enable Image, VAS=A32, VDW=D32, LAS=Mem. Space, PGM=data, SUPER=non-priv, Posted Write enabled, BLT allowed.*
D:0x0000.0040 to A:0x00012C	0x4000.0000	*LSI_2: Base Address set to 0x4000.0000*
D:0x0000.0060 to A:0x000130	0x6000.0000	*LSI_2: Bound Address set to 0x6000.0000*
D:0x0000.00C0 to A:0x000134	0xC000.0000	*LSI_2: Translation Offset set to 0xC000.0000*
D:0x0001.C2C0 to A:0x000128	0xC0C2.0100	*LSI_2: Enable Image, VAS=A32, VDW=D64, LAS=Mem. Space, PGM=data, SUPER=non-priv, Posted Write enabled, BLT allowed.*

[1)] This column shows write data for configuration from PCI

Table 19. Initialization Sequence for PCI Slave Image Config. Example.

Appendix III: PXB Information

PXB Register Descriptions

P-Side Register Descriptions

If no valid PROM is present, the PXB powers up in 'endpoint mode', but should always be used in bridge mode. This is done by clearing bit 6 in register +0x40.

Configuration Space Registers				Offset
Device ID *(PXB=0001)*		Vendor ID *(MC=1134)*		00
Status *(0480=default)*		Command		04
Class Code *(060000)*			Revision *(00)*	08
00	Header *(01)*	Latency	Cache	0C
Prefetchable Memory BAR *(Endpoint Mode Only)*				10
Memory Mapped I/O BAR *(internal registers)*				14
Sec Latency	Sub Bus#	Sec Bus#	Pri Bus#	18
Secondary Status		I/O Limit	I/O Base	1C
Memory Mapped I/O Limit		Memory Mapped I/O Base		20
Pref. Memory Limit		Pref. Memory Base		24
0000.0000				28
0000.0000				2C
I/O Limit (upper 16)		I/O Base (upper 16)		30
0000.0000				34
EEPROM Address Register		*0000*		38
Bridge Control		Int. Pin	Int. Line	3C
PXB Misc.	IO/MIO Shift	Memory Window	PXB Bridge Control	40
Memory Internal Base				44
Misc. BAR Size				48
				4C
Mailbox Vector				50
Memory Mask				54
I/O Mask				58
MI/O Mask				5C
Subsystem Vendor Information				60
PCI Miscon				64
Arbitration Control				68
				6C
EEPROM Data Register				70

Table 20. PXB P-side CSR Registers

'PCI Command Register'

Register Name: PCMDR	Size: 16 bits	Offset: 0x04

Bits	Function							
15:8	*RESERVED (000000).*						*0*	SERR_EN
7:0	*0*	PERR_EN	*0*	*0*	*0*	BM	MS	IOS

PCMDR Description

Name	Type	Reset State	Function
SERR_EN	R/W	0	SERR Enable
PERR_EN	R/W	0	PERR Generation enable
BM	R/W	1	PCI Master Enable
MS	R/W	1	Memory Space Target Enable
IOS	R/W	1	I/O Space Target Enable

Table 21. PCI Command Register

'PCI Status Register'

Register Name: PSR	Size: 16 bits	Offset: 0x06

Bits	Function						
15:8	DPE	SSE	RMAB	RTAB	STAB	DEVSEL	DPD
7:0	*RESERVED (000000).*						

PSR Description

Name	Type	Reset State	Function
DPE	R/W		Detected parity error
SSE	R/W		Signaled SERR
RMAB	R/W		Received Master Abort
RTAB	R/W		Received Target Abort
STAB	R	0	Signaled Target Abort. Always 0.
DEVSEL	R	10	DEVSEL Timing. 10=slow.
DPD	R/W	0	Data Parity Detected.

Table 22. PCI Status Register

'Secondary Status Register'

Register Name: SSR	Size: 16 bits	Offset: 0x1E

Bits	Function						
15:8	DPE	RSE	RMAB	RTAB	STAB	DEVSEL	DPD
7:0	*RESERVED (000000).*						

SSR Description

Name	Type	Reset State	Function
DPE	R/W		Detected parity error
RSE	R/W		Received SERR
RMAB	R/W		Received Master Abort
RTAB	R/W		Received Target Abort
STAB	R	0	Signaled Target Abort. Always 0.
DEVSEL	R	10	DEVSEL Timing. 10=slow.
DPD	R/W	0	Data Parity Detected.

Table 23. Secondary Status Register

'Memory Mapped I/O Base Address Register'

Register Name: MIOBAR	Size: 16 bits	Offset: 0x20

Bits	Function	
15:8	MIOBA(31:24)	
7:0	MIOBA(23:16)	*0000*

MIOBAR Description

Name	Type	Reset State	Function
MIOBA	R/W		Base Address (inclusive) for Memory Mapped I/O (20 lsb assumed 0). Alignment 1MB.

Table 24. Memory Mapped I/O Base Address Register

'Memory Mapped I/O Limit Address Register'

Register Name: MIOLAR	Size: 16 bits	Offset: 0x22

Bits	Function	
15:8	MIOLA(31:24)	
7:0	MIOLA(23:16)	*0000*

MIOLAR Description

Name	Type	Reset State	Function
MIOLA	R/W		Base Limit (inclusive) for Memory Mapped I/O (20 lsb assumed 1). Alignment 1MB.

Table 25. Memory Mapped I/O Limit Address Register

'Prefetchable Memory Base Address Register'

Register Name: PMBAR	Size: 16 bits	Offset: 0x24

Bits	Function	
15:8	PMBA(31:24)	
7:0	PMBA(23:16)	*0000*

PMBAR Description

Name	Type	Reset State	Function
PMBA	R/W		Base Address (inclusive) for Prefetchable Memory space (20 lsb assumed 0). Alignment 1MB.

Table 26. Prefetchable Memory Base Address Register

'Prefetchable Memory Limit Address Register'

Register Name: PMLAR	Size: 16 bits	Offset: 0x26

Bits	Function	
15:8	PMLA(31:24)	
7:0	PMLA(23:16)	*0000*

PMLAR Description

Name	Type	Reset State	Function
PMLA	R/W		Base Limit (inclusive) for PrefetchableMemory space (20 lsb assumed 1). Alignment 1MB.

Table 27. Prefetchable Memory Limit Address Register

'Bridge Control Register'

Register Name: BCR	Size: 16 bits	Offset: 0x3E

Bits	Function							
15:8	*RESERVED (0000.0000).*							
7:0	FBTB	SBRES	MAM		VGAEN	ISAEN	SERREN	PERREN

BCR Description

Name	Type	Reset State	Function
FBTB	R/W	0	Fast Back to Back Enable
SBRES	R/W	0	Secondary Bus Reset. 0 = Do not assert RST 1 = Assert RST on secondary bus
MAM	R/W		Master Abort Mode. 0= Do not report master abort (all 1) 1= Report master abort with target abort.
VGAEN	R/W	0	VGA Enable
ISAEN	R/W	0	ISA Enable.
SERREN	R/W	0	Enable for system errors detected on secondary bus and reported to primary bus.
PERREN	R/W	0	Enable for parity errors response on secondary bus.

Table 28. Bridge Control Register

'PXB Bridge Control Register'

Register Name: PBCR	Size: 8 bits	Offset: 0x40

Bits	Function							
7:0	EIBAR	MODE	WSWAP	BSWAP	*0*	PRIM	RPRIM	EPBAR

PBCR Description

Name	Type	Reset State	Function
EIBAR	R/W		Enable memory internal BAR. (1=enable, 0=disable)
MODE	R/W		Operating Mode. 0 = Bridge Mode 1 = Endpoint Mode *This bit should always be cleared (use Bridge Mode !).*
WSWAP	R/W	1	Word Swap (1=enable, 0=disable)
BSWAP	R/W	0	Byte Swap (1=enable, 0=disable)
PRIM	R/W		Prim/Sec Mode 0= Secondary Mode 1 = Primary Mode
RPRIM	R		Same as above, read
EPBAR	R/W		Enable prefetchable memory BAR. (1=enable, 0=disable)

Table 29. PXB Bridge Control Register

'Memory Window Register'

Register Name: MWR	Size: 8 bits	Offset: 0x41

Bits	Function	
7:0	PPAGE	MWSHIFT

MWR Description

Name	Type	Reset State	Function
PPAGE	W		Primary Page. Used by secondary PXB as index into page ram when out-of-bounds accesses occurs; should be set to unused page register. Returns zeros when read.
MWSHIFT	R/W		Memory window shift. Selects which address bits are used to index page ram during memory accesses. Bits "Big Window" "Small Window" 0000 31:28 4GB 256 MB 0001 30:27 2 GB 128 MB 0010 29:26 1 GB 64 MB 0011 28:25 512 MB 32 MB 0100 27:24 256 MB 16 MB 0101 26:23 128 MB 8 MB 0110 25:22 64 MB 4 MB 0111 24:21 32 MB 2 MB 1xxx 23:20 16 MB 1 MB

Table 30. Memory Window Register

'IO/MIO Window Register'

Register Name: IOMIOWR	Size: 8 bits	Offset: 0x42

Bits	Function	
7:0	IOSHIFT	MIOSHIFT

MWR Description

Name	Type	Reset State	Function
IOSHIFT	R/W		I/O window shift. Selects which address bits are used to index page ram during I/O accesses. Bits — "Big Window" — "Small Window" 0000 23:20 16 MB 1 MB 0001 22:19 8 MB 512 KB 0010 21:18 4 MB 256 KB 0011 20:17 2 MB 128 KB 0100 19:16 1 MB 64 KB 0101 18:15 512 KB 32 KB 0110 17:14 256 KB 16 KB 0111 16:13 128 KB 8 KB 1xxx 15:12 64 KB 4 KB
MIOSHIFT	R/W		Memory Mapped I/O window shift. Selects which address bits are used to index page ram during memory mapped I/O accesses. Bits — "Big Window" — "Small Window" 0000 29:26 1 GB 64 MB 0001 28:25 512 MB 32 MB 0010 27:24 256 MB 16 MB 0011 26:23 128 MB 8 MB 0100 25:22 64 MB 4 MB 0101 24:21 32 MB 2 MB 0110 23:20 16 MB 1 MB 0111 22:19 8 MB 512 KB 1xxx 21:18 4 MB 256 KB

Table 31. IO/MIO Window Register

'PXB Misc. Register'

Register Name: PMR	Size: 8 bits	Offset: 0x43

Bits	Function				
7:0			UNAL	NAL	PPAGE

PMR Description

Name	Type	Reset State	Function
UNAL	R/W	0	Unaligned (1=unaligned, 0=aligned). *For most normal operations "aligned" operation is used.*
NAL	R/W		No auto load. Used to control if the PCI mask registers are auto loaded for each RACEway-to-PCI transaction. *Should always be 0 for secondary PXBs.*
PPAGE	R		Primary Page. Read only.

Table 32. PXB Misc. Register

X-Side Register Descriptions

X-side Internal Registers	Offset
MailBox Write	400
Generate/Clear Interrupts	408
MISCON	410
Force Test Increment (test only)	418
Interrupt Mask Load/ Status Read	420
Unused	428
Unused	430
Load RTC/Performance #1&#2 From Modulus	438
Load RTC/Performance #1 Modulus	440
Load RTC/Performance #1 From Modulus / Read RTC/Performance #1	448
Load RTC/Performance #2 Modulus	450
Load RTC/Performance From Modulus / Read RTC/Performance #2	458
Load Free Running Counter	460
Write Mailbox Base Address	470
DMA Remote Address (diagnostics)	478
DMA Nest Pointer	480
DMA Local Address (diagnostics)	488
DMA Word Count (diagnostics)	490
DMA Link Address (diagnostics)	498
DMA Last Pointer (diagnostics)	4A0
Clear Remote Bus Error Interrupt	4B0
Clear RTC-1 Interrupt	4E8
Clear RTC-2 Interrupt	4F0
Clear Mailbox Interrupt (primary side)	4F8
Route for Page - 0	504
Return Route for Page - 0	50C
Route for Page - 1	514

Return Route for Page - 1	51C
Route for Page - 2	524
Return Route for Page - 2	52C
Route for Page - 3	534
Return Route for Page - 3	53C
Route for Page - 4	544
Return Route for Page - 4	54C
Route for Page - 5	554
Return Route for Page - 5	55C
Route for Page - 6	564
Return Route for Page - 6	56C
Route for Page - 7	574
Return Route for Page - 7	57C
Route for Page - 8	584
Return Route for Page - 8	58C
Route for Page - 9	594
Return Route for Page - 9	59C
Route for Page - 10	5A4
Return Route for Page - 10	5AC
Route for Page - 11	5B4
Return Route for Page - 11	5BC
Route for Page - 12	5C4
Return Route for Page - 12	5CC
Route for Page - 13	5D4
Return Route for Page - 13	5DC
Route for Page - 14	5E4
Return Route for Page - 14	5EC
Route for Page - 15	5F4
Return Route for Page - 15	5FC

Table 33. PXB X-side CSR Registers

Miscellaneous PXB Information

Configuration Serial EEPROM

- In typical P2P applications, the configuration PROM is normally used only by the primary PXB. Secondary PXBs are set up from the host using config. cycles type 1, through the primary PXB.

- In an embedded application using a fixed predetermined address map, where no 'off-the-shelf' POST initialization code is running, all PXBs may be initialized almost completely from the configuration PROM. The only bit field which must be set is the 'Routes_to_Primary' bit, in the Miscon register (+0x410).
- All configuration registers inside the PXB may be initialized from either side of the chip.
- In order for a primary PXB to do Type1-to-Type0 configuration operations, or virtual Type1-to-Type0 operations (accessing CSRs in secondary PXBs), the lookup tables for config-ops in the PROM must be initialized. The contents of these tables are used to index the route table for configuration type 1 accesses. The tables are located in PROM address ranges 0x80-0xBF, and 0x220-→.

PCI-to-RACEway Addressing

- Three bits of PCI address are used to index the Route Table. This table holds the routes used to set up a connection through the crossbars.
- With Big Mem enabled, the least significant portion of the Return Route holds the most significant bits of the RACEway address. Bit 0 of the Return Route will replace the least significant PCI bus address bit used to index the Route Table. The most significant bit replaced is always bit 27 of the PCI address. Big Mem is only used for prefetchable memory space. **Big Mem is not used for P2P applications.**

Configuration Cycles

- Lookup table, in EEPROM, is used to index the route table for each PCI bus device.

PXB Route Format

Bits	Function				
31:24	Routes				
23:16	Routes				
15:8	Routes				
7:0		BigMem	Not Splitable	Priority	B'cast

PXB Route Description

Name	Function
Routes	Concatenated three-bit route codes, one per crossbar hop. Route codes: 111 = Port A, 110 = Port B, 101 = Port C, 100 = Port D 011 = Port E, 010 = Port F, 001 = Adaptive Route (E first), 000 = Adaptive Route (F first). *Example: Route entry: 0xFACx.xxxA is used to route a transfer through 4 crossbars, ports A, B, C, and D, with splitable bit set, and priority 01.*
BigMem	Enables addition of high order address bits when addressing RACEway from PCI. **Not to be used for P2P operation.**
Split Disable	A 1 disables split transactions on RACEway. **Split should always be enabled.**
Priority	Two bit priority code. Ref. RACEway specification. Note: Never use 11b as priority for I/O space.
Broadcast	Set for broadcast operations on RACEway. In broadcast mode the meaning of route codes change. Ref. RACEway specification. Requires split disabled and no BigMem.

Table 34. PXB Route Format

PXB Return Route Format

Bits	Function
31:24	Return Routes
15:8	Return Routes
15:8	Return Routes
7:0	Big Address

PXB Return Route Description

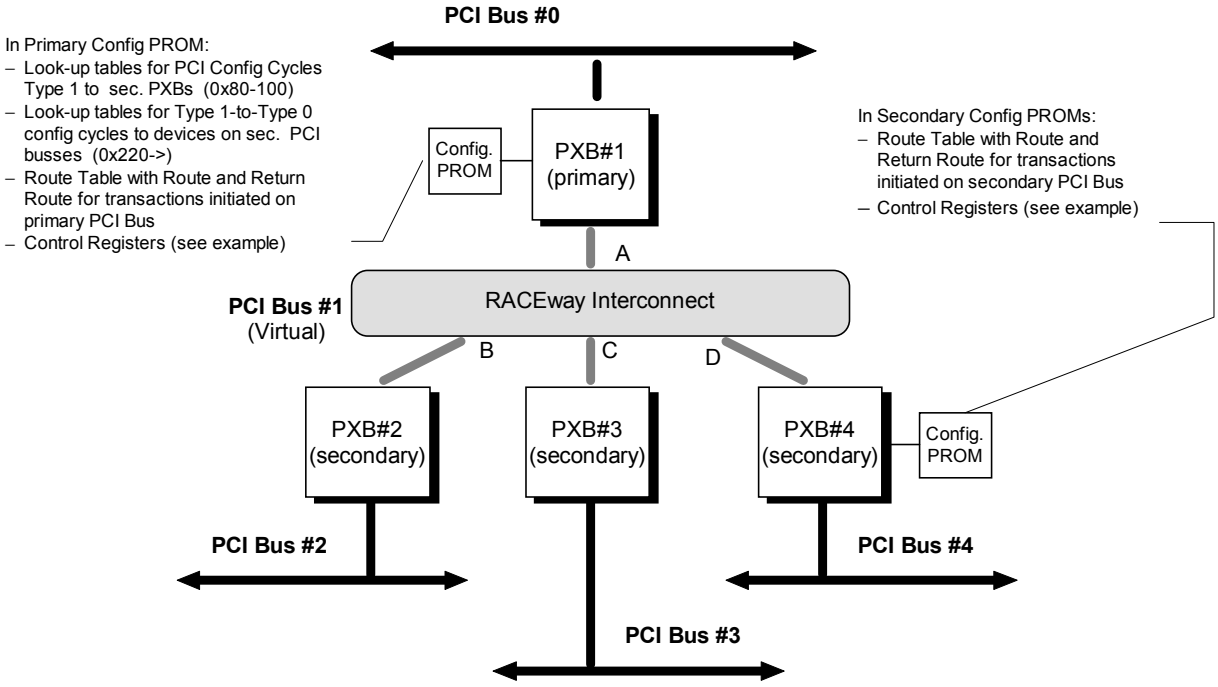
Name	Function
Return Routes	Concatenated three-bit route codes used in the response of split RACEway transactions, one three-bit field per hop. Return Route codes: 111 = Port A, 110 = Port B, 101 = Port C, 100 = Port D 011 = Port E, 010 = Port F, 001 = Adaptive Route (E first), 000 = Adaptive Route (F first). **If 'Split Disable' in the route is set, the return route is not used.**
BigAddress	In BigMem Mode a number of bits (number is depending on prefetchable memory window size) from this field replaces the most significant address bits when going from PCI into RACEway. Bit 0 of this field replaces the least significant bit used to index the page table. The most significant bit replaced is always bit 27 of the PCI address. With 256MByte windows no bits are replaced. **If BigMem Mode is not used (always the case for P2P applications), this field is not used.**

Table 35. PXB Return Route Format

RACEway-to-PCI Addressing

- PCI Mask registers (one for each address window) are always used by secondary PXB to generate the high-order PCI address bits. All bits from the least significant '1' and up to 3 are used as PCI bus addresses. These registers are normally loaded automatically from the BARs for each transaction (i.e. as long as the no-autoload bit in register +0x40 is cleared). **The register is not used by primary PXB.**

PCI-to-PCI Bridge Operation



- Both a memory mapped I/O window and a prefetchable memory window must be defined in order to make the bridge operation from secondary to primary side work correctly.
- BAR and limit registers of the secondary PXB are set to values corresponding to the address space on their PCI side. PXB will calculate the size of the “big window” (BAR-limit on primary PXB) based on the BAR and window size. Widow sizes set in +0x40 must be the same for primary and secondary PXBs.

PXB Initialization Example

Below is an example on how to setup registers of two PXB chips so that they provide a PCI-to-PCI bridge across a RACEway interlink (ILK4).

The PXB, in the RACEway slot A, is referred to as the primary PXB, and the PXB, in the slot B, as the secondary PXB.

The following example creates an 8 Mbytes window, at PCI Memory space addresses 0x80000000-0x807fffff, from the primary to the secondary side i.e. the primary PXB will respond to address cycles between 0x80000000-0x807fffff, and forward them to the secondary side. The secondary PXB will forward transactions in address ranges 0x0 - 0x7FFFFFFF and 0x81000000 - 0xFFFFFFFF, to the primary side.

The initialization values may easily be programmed into the PXB, using PCI Configuration Type 0 cycles. The offsets for each register are given.

PCI-to-PCI Bridge Configuration Space Header Registers of the Primary PXB:

Register	Offset	Value
Device and Vendor ID	0x0	0x00011134
Command and Status	0x4	0x04800007
Class Code + Revision ID	0x8	0x06000000
Misc.	0xc	0x00010000
Base Addr. 0	0x10	0x0
Base Addr. 1	0x14	0x0
Bus numbers and Sec.Lat.Timer	0x18	0x00030100
Sec.Status and IO window	0x1C	0x04800111
Memory Window	0x20	0x00000010
Pref. Memory Window	0x24	0x80f08000
Pref. Base Upper 32 bits	0x28	0x00000000
Pref. Limit Upper 32 bits	0x2c	0x00000000
IO upper	0x30	0x00000000
Reserved	0x34	0x00000000
Exp. ROM Base Address	0x38	0x00000000
Bridge and Interrupt Control	0x3c	0x00000100
PXB Bridge Control	0x40	0x10880826
Memory window	0x44	0x00000000
EEPROM register	0x48	0x0000f773
.	0x4c	0x00000000
Mailbox reg.	0x50	0x00000000
Mem. Mask. Reg.	0x54	0x00000000
IO Mask Reg.	0x58	0x00000000
MIO Mask Reg.	0x5c	0x00000000
Subsystem Identification	0x60	0x00000000
PCI Miscon Reg.	0x64	0x00000200
Arbitration Control	0x68	0x00000000

Route Table for the Primary PXB:

Register	Offset	Value
Route for Page 0	0x500	0xc0000000
Return route for Page 0	0x508	0xe0000080
Route for Page 1	0x510	0xc0000000
Return route for Page 1	0x518	0xe0000081
Route for Page 2	0x520	0xc0000000
Return route for Page 2	0x528	0xe0000082
Route for Page 3	0x530	0x c0000000
Return route for Page 3	0x538	0x e0000083
Route for Page 4	0x540	0x c0000000
Return route for Page 4	0x548	0x e0000084
Route for Page 5	0x550	0x c0000000
Return route for Page 5	0x558	0x e0000085
Route for Page 6	0x560	0x c0000000
Return route for Page 6	0x568	0x e0000086
Route for Page 7	0x570	0x c0000000
Return route for Page 7	0x578	0x e0000087
Route for Page 8	0x580	0x a0000000
Return route for Page 8	0x588	0x e0000088
Route for Page 9	0x590	0x a0000000
Return route for Page 9	0x598	0x e0000089
Route for Page 10	0x5a0	0x a0000000
Return route for Page 10	0x5a8	0x e000008a
Route for Page 11	0x5b0	0x a0000000
Return route for Page 11	0x5b8	0x e0000000
Route for Page 12	0x5c0	0x 80000000
Return route for Page 12	0x5c8	0x e0000000
Route for Page 13	0x5d0	0x 80000000
Return route for Page 13	0x5d8	0x e0000000
Route for Page 14	0x5e0	0x 80000000
Return route for Page 14	0x5e8	0x e0000000
Route for Page 15	0x5f0	0x 80000000
Return route for Page 15	0x5f8	0x e0000000

PCI-to-PCI Bridge Configuration Space Header Registers of the Secondary PXB:

Register	Offset	Value
Device and Vendor ID	0x0	0x00011134
Command and Status	0x4	0x04800007
Class Code + Revision ID	0x8	0x06040000
Misc.	0xc	0x00010000
Base Addr. 0	0x10	0x0
Base Addr. 1	0x14	0x0
Bus numbers and Sec.Lat.Timer	0x18	0x00020201

Sec.Status and IO window	0x1C	0x04800111
Memory Window	0x20	0x00000010
Pref. Memory Window	0x24	0x80708000
Pref. Base Upper 32 bits	0x28	0x00000000
Pref. Limit Upper 32 bits	0x2c	0x00000000
IO upper	0x30	0x80708000
Reserved	0x34	0x00000000
Exp. ROM Base Address	0x38	0x00000000
Bridge and Interrupt Control	0x3c	0x00000000
PXB Bridge Control	0x40	0x00800820
Memory window	0x44	0x00000000
EEPROM register	0x48	0x00005550
.	0x4c	0x00000000
Mailbox reg.	0x50	0x00000000
Mem. Mask. Reg.	0x54	0x80400000
IO Mask Reg.	0x58	0x80400000
MIO Mask Reg.	0x5c	0x00100000
Subsystem Identification	0x60	0x00000000
PCI Miscon Reg.	0x64	0x00000700
Arbitration Control	0x68	0x00000020

Route Table for the Secondary PXB:

Register	**Offset**	**Value**
Misc. Control	0x418	0x00020000
Route for Page 0	0x500	0xe0000008
Return route for Page 0	0x508	0xc0000000
Route for Page 1	0x510	0xe0FFFFFF
Return route for Page 1	0x518	0xFFFFFFFF
Route for Page 2	0x520	0xFFFFFFFF
Return route for Page 2	0x528	0xFFFFFFFF
Route for Page 3	0x530	0xFFFFFFFF
Return route for Page 3	0x538	0xFFFFFFFF
Route for Page 4	0x540	0xFFFFFFFF
Return route for Page 4	0x548	0xFFFFFFFF
Route for Page 5	0x550	0xFFFFFFFF
Return route for Page 5	0x558	0xFFFFFFFF
Route for Page 6	0x560	0xFFFFFFFF
Return route for Page 6	0x568	0xFFFFFFFF
Route for Page 7	0x570	0xFFFFFFFF
Return route for Page 7	0x578	0xFF000000
Route for Page 8	0x580	0x a0000008
Return route for Page 8	0x588	0x c0000000
Route for Page 9	0x590	0x a0000008
Return route for Page 9	0x598	0x c0000000
Route for Page 10	0x5a0	0x a0000008
Return route for Page 10	0x5a8	0x c0000000
Route for Page 11	0x5b0	0x a0000008

Return route for Page 11	0x5b8	0x c0000000
Route for Page 12	0x5c0	0x a0000008
Return route for Page 12	0x5c8	0x c0000000
Route for Page 13	0x5d0	0x a0000008
Return route for Page 13	0x5d8	0x c0000000
Route for Page 14	0x5e0	0x a0000008
Return route for Page 14	0x5e8	0x c0000000
Route for Page 15	0x5f0	0x a0000008
Return route for Page 15	0x5f8	0x c0000000

Appendix IV:

List of Tables

Table 1. Power Consumption 12
Table 2. 'Universe II' Power-Up Options 19
Table 3. MIDAS Configuration ROM 20
Table 4. VRAI Base Address Definition 22
Table 5. Secondary PCI Bus 26
Table 6. Primary PCI Bus - Arbiter: RD#1 26
Table 7. Primary PCI Bus - Arbiter: PXB 27
Table 8. IDSEL Assignments for Primary PCI Bus 27
Table 9. IDSEL Assignments for Secondary PCI Bus 28
Table 10. Interrupt Mode Selection 30
Table 11. VME_RAI Setup. 38
Table 12. VME Slave Image 0 - Setup 38
Table 13. VME Slave Image 1 - Setup. 39
Table 14. VME Slave Image 2 - Setup. 39
Table 15. Initialization Sequence for VMEbus Slave Image Config. Example. 40
Table 16. PCI Slave Image 0 Setup. 42
Table 17. PCI Slave Image 1 Setup. 42
Table 18. PCI Slave Image 2 Setup. 42
Table 19. Initialization Sequence for PCI Slave Image Config. Example. 43
Table 20. PXB P-side CSR Registers 44
Table 21. PCI Command Register 45
Table 22. PCI Status Register 45
Table 23. Secondary Status Register 46
Table 24. Memory Mapped I/O Base Address Register 46
Table 25. Memory Mapped I/O Limit Address Register 47
Table 26. Prefetchable Memory Base Address Register 47
Table 27. Prefetchable Memory Limit Address Register 48
Table 28. Bridge Control Register 48
Table 29. PXB Bridge Control Register 49
Table 30. Memory Window Register 49
Table 31. IO/MIO Window Register 50
Table 32. PXB Misc. Register 51
Table 33. PXB X-side CSR Registers 52
Table 34. PXB Route Format 54
Table 35. PXB Return Route Format 55

List of Figures

Figure 1. MIDAS-220/250 Block Diagram....3
Figure 2. MIDAS-120/150 Block Diagram....4
Figure 3. i960®RD Data Flow....5
Figure 4 PCI-VME bridge functional diagram....6
Figure 5. MIDAS Board Layout (No model has all parts mounted.)....8
Figure 6. Steps 2&3: Mount PMC modules on the MEZZ-x50 and MIDAS-x20 boards....10
Figure 7. Step 4: Mount the MEZZ-x50 with PMC modules on the MIDAS-x20 board....11
Figure 8. i960®RD with External Devices....14
Figure 9. Local Memory Address Space....15
Figure 10. FLASH Boot Address....17
Figure 11. Pin definition for the RS232 female connector mounted on the front panel....17
Figure 12. Connection of two i960®RDs on MIDAS-220....18
Figure 13. DIP Switch Details. (Shown switch value: 0x81)....22
Figure 14. Configuration Example for VMEbus Slave Images....37
Figure 15. Configuration Example for PCI Slave Images....41