VX407C
Intelligent PXI Carrier
(408216-xxxx)
User Manual

Publication No. 11028564 Rev B

Astronics Test Systems Inc.
4 Goodyear, Irvine, CA 92618
Tel: (800) 722-2528, (949) 859-8999; Fax: (949) 859-7139

atsinfo@astronics.com atssales@astronics.com
atshelpdesk@astronics.com http://www.astronicstestsystems.com

THANK YOU FOR PURCHASING THIS ASTRONICS TEST SYSTEMS PRODUCT

For this product, or any other Astronics Test Systems product that incorporates software drivers, you may access our web site to verify and/or download the latest driver versions. The web address for driver downloads is:

http://www.astronicstestsystems.com/support/downloads

If you have any questions about software driver downloads or our privacy policy, please contact us at:

atsinfo@astronics.com

WARRANTY STATEMENT

All Astronics Test Systems products are designed to exacting standards and manufactured in full compliance to our AS9100 Quality Management System processes.

This warranty does not apply to defects resulting from any modification(s) of any product or part without Astronics Test Systems express written consent, or misuse of any product or part. The warranty also does not apply to fuses, software, non-rechargeable batteries, damage from battery leakage, or problems arising from normal wear, such as mechanical relay life, or failure to follow instructions.

This warranty is in lieu of all other warranties, expressed or implied, including any implied warranty of merchantability or fitness for a particular use. The remedies provided herein are buyer's sole and exclusive remedies.

For the specific terms of your standard warranty, contact Customer Support. Please have the following information available to facilitate service.

1. Product serial number
2. Product model number
3. Your company and contact information

You may contact Customer Support by:

E-Mail:	atshelpdesk@astronics.com	
Telephone:	+1 800 722 3262	(USA)
Fax:	+1 949 859 7139	(USA)

RETURN OF PRODUCT

Authorization is required from Astronics Test Systems before you send us your product or sub-assembly for service or calibration. Call or contact Customer Support at 1-800-722-3262 or 1-949-859-8999 or via fax at 1-949-859-7139. We can also be reached at: atshelpdesk@astronics.com.

If the original packing material is unavailable, ship the product or sub-assembly in an ESD shielding bag and use appropriate packing materials to surround and protect the product.

PROPRIETARY NOTICE

This document and the technical data herein disclosed, are proprietary to Astronics Test Systems, and shall not, without express written permission of Astronics Test Systems, be used in whole or in part to solicit quotations from a competitive source or used for manufacture by anyone other than Astronics Test Systems. The information herein has been developed at private expense, and may only be used for operation and maintenance reference purposes or for purposes of engineering evaluation and incorporation into technical specifications and other documents which specify procurement of products from Astronics Test Systems.

TRADEMARKS AND SERVICE MARKS

All trademarks and service marks used in this document are the property of their respective owners.

- Racal Instruments, Talon Instruments, Trig-Tek, ActivATE, Adapt-A-Switch, N-GEN, and PAWS are trademarks of Astronics Test Systems in the United States.

DISCLAIMER

Buyer acknowledges and agrees that it is responsible for the operation of the goods purchased and should ensure that they are used properly and in accordance with this document and any other instructions provided by Seller. Astronics Test Systems products are not specifically designed, manufactured or intended to be used as parts, assemblies or components in planning, construction, maintenance or operation of a nuclear facility, or in life support or safety critical applications in which the failure of the Astronics Test Systems product could create a situation where personal injury or death could occur. Should Buyer purchase Astronics Test Systems product for such unintended application, Buyer shall indemnify and hold Astronics Test Systems, its officers, employees, subsidiaries, affiliates and distributors harmless against all claims arising out of a claim for personal injury or death associated with such unintended use.

FOR YOUR SAFETY

Before undertaking any troubleshooting, maintenance or exploratory procedure, read carefully the **WARNINGS** and **CAUTION** notices.

This equipment contains voltage hazardous to human life and safety, and is capable of inflicting personal injury.

If this instrument is to be powered from the AC line (mains) through an autotransformer, ensure the common connector is connected to the neutral (earth pole) of the power supply.

Before operating the unit, ensure the conductor (green wire) is connected to the ground (earth) conductor of the power outlet. Do not use a two-conductor extension cord or a three-prong/two-prong adapter. This will defeat the protective feature of the third conductor in the power cord.

Maintenance and calibration procedures sometimes call for operation of the unit with power applied and protective covers removed. Read the procedures and heed warnings to avoid “live” circuit points.

Before operating this instrument:

1. Ensure the proper fuse is in place for the power source to operate.
2. Ensure all other devices connected to or in proximity to this instrument are properly grounded or connected to the protective third-wire earth ground.

If the instrument:

- fails to operate satisfactorily
- shows visible damage
- has been stored under unfavorable conditions
- has sustained stress

Do not operate until performance is checked by qualified personnel.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DOCUMENT CHANGE HISTORY

Revision	Date	Description of Change
B	10/26/09	Initial Astronics Test Systems Release

This manual describes the operation and use of the VX407C Intelligent PXI/cPCI Carrier Module (Astronics Test Systems part number 408216-xxxx).

Contained within this manual are the physical and electrical specifications, installation and startup procedures, functional description, and configuration guidelines to adequately use the product.

The product part numbers covered by this manual are:

Part Number	Description
408216-0001	VX407C Single Wide Module
408216-0002	VX407C Double Wide Module

1.0 GENERAL DESCRIPTION

The VX407C is an intelligent VXI carrier that allows PXI and CompactPCI (cPCI) modules to be used in VXI systems. The carrier supports two 3U or one 6U PXI or cPCI modules. It features an on-board PowerPC processor that can perform command translation, data analysis, and many other data processing or process control functions.

1.1 PURPOSE OF EQUIPMENT

The VX407C was designed for Automated Test Equipment (ATE) applications requiring on-board instrument intelligence or data processing. Some of the more common applications include: legacy instrument emulation, data intensive signal acquisition and control, high speed signal analysis, and control processing.

1.2 SPECIFICATIONS OF EQUIPMENT

1.2.1 Key Features

- ☐ 300MHz Motorola MPC8245 Integrated Processor
- ☐ Two 33 MHz, 5V or 3.3V PXI/cPCI slots
- ☐ One 33 MHz, 5V or 3.3V PMC slot
- ☐ 128 megabytes P133 SDRAM
- ☐ 16 kilobytes dual-ported SRAM accessible by both the processor and VXI
- ☐ 8 megabytes flash memory
- ☐ Message based or register based VXI interface
- ☐ VXI A24/A32 access to shared memory

- ☐ VXI block transfers to/from shared memory
- ☐ DMA transfers between PowerPC, PCI devices and shared memory
- ☐ Direct access to PXI/cPCI and PMC modules from VXI
- ☐ On-board system utilities supports application development
- ☐ Supports common off-the-shelf real-time operating systems

1.2.2 Specifications

Processor:
- Motorola 300MHZ MPC8245
- MPC603e core
- 16KB/16KB L1 Integrated Cache

Local PCI Bus:
- 33MHZ 32-bit

Main Memory:
- 128MB SDRAM
- 8MB Flash, VXI programmable
- 32KB Boot ROM, socketed

Shared Memory:
- 16 KB Dual-ported SRAM
- Four 32 deep 32-bit FIFO's
- DMA/Burst support
- Internal arbitration
- Fully accessible by both VXI and PowerPC

cPCI/PXI Interface:
- Two 3U modules or one 6U module
- 33MHz 32-bit
- PXI triggers map to VXI TTL triggers
- cPCI/PXI interrupt to PowerPC supported
- On-board PXI CLK10 source

PMC Interface:
- Support for one PMC module
- IEEE P1386.1 32-bit compliant
- 33MHz 32-bit
- PMC I/O connected to 64-pin header

External Relay Control:
- Darlington relay driver, 8-channels
- Controlled by PowerPC
- 5V 500mA (single channel)
- 16-pin header

Interrupts:
- PCI to PowerPC interrupt support
- PowerPC to VXI interrupt level 1-7 (programmable)
- VXI Host to PowerPC interrupt support

Temperature:

Operating:	0°C to 50°C
Storage:	-40°C to 70°C

Direct Access:
- Direct VXI access of cPCI/PXI modules
- Up to 8K of local PCI address space can be directly mapped to VXI A24 or A32 space

Debugging Interface:
- Common On-Chip Processor (COP)/JTAG
- Standard COP header
- Third-party development tools supported

On-Board System Utilities:
- Boot-up and initialization
- VXI word serial protocol support
- Firmware download to Flash memory via VXI
- PCI bus enumeration

RTOS Support:

Architecture supports common real-time operating systems, such as VxWorks, OS-9, Linux, and others.

1.2.3 Electrical

The VX407C requires the +5V, ±12V, and ±24V power from the VXI back plane. The +5V supply drives a DC to DC converter supplying +3.3V to carrier components and the PXI/cPCI and PMC positions. The VXI backplane can provide a total of 7.2 amps of +5

volts, of which, the VX407C uses a maximum of 2.3 amps (11.5W) for internal purposes. The remaining 4.9 amps (24W) are available to the PXI/cPCI and PMC through a combination of the +5V and +3.3V supplies. Of that 4.9 amps, a maximum of 4.4 amps (14.5W) can be provided via the +3.3V supply.

The ±12 volt supply is not used internally by the carrier, but may be required by an installed PXI/cPCI or PMC module. The carrier can provide up to 1.5 amps (18W) each of both +12 volts and -12 volts to the PXI/cPCI and PMC positions.

The ±24V is neither used by the carrier nor the PXI/cPCI and PMC modules. However, it is available at an external connector for special purpose use. The carrier can supply a maximum of 1 amp (24W) each to the +24V and the -24V pins on the connector.

An external connection to the +5V supply is also provided. A maximum of 2.5 amps (12.5W) can be sourced or sink-ed to/from the carrier. As an output, the power drawn from this connector reduces the total power available to the PXI/cPCI and PMC positions. As an input, the power provided to this connector increases the available power to the PXI/cPCI and PMC positions. Even with an external +5V source an absolute maximum of 6 amps of +5 volts and 4.4 amps of +3.3 volts can be provided to the PXI/cPCI and PMC positions.

For electrical information on individual PXI/cPCI modules, please reference each module's documentation. The power requirements for each PXI/cPCI or PMC module installed must be added to the VX407C's requirements for the total module's requirements.

1.2.4 Mechanical

The mechanical dimensions of the VX407C are in conformance with the VXI bus specification for the height and width of Size-C modules. The nominal dimensions are 233.35 mm (9.187 in) high x 189.0 mm (7.441 in) deep. With the shield and a PXI/cPCI module installed, the total dimensions are 233.35 mm (9.187 in) high x 340.0 mm (13.386 in) deep. The module is designed for a standard mainframe with 30.48 mm (1.2 in) width between slots. The double-wide option is 60.96 mm (2.4 in) wide and will occupy two slots. The triple wide option is 91.44 mm (3.6 in) wide and will occupy three slots.

1.2.5 Environmental

The environmental specifications of the module are:

Operating Temperature:	0°C to +50°C
Storage Temperature:	-40°C to +70°C
Humidity:	<95% without condensation

1.2.6 Bus Compliance

The module complies with the VXIbus Specification Revision 1.4 for C-Size VXI modules and with VMEbus Specification ANSI/IEEE STD 1014-1987, IEC 821.

Manufacturer ID:	$FC1_{16}$ (can also be set by PowerPC)
Model Code:	$FE4_{16}$ (can also be set by PowerPC)
VXI Access Type:	Register Based or Message Based
VXI Addressing:	A16/A24/A32
VXI Data Transfer:	D16/D32
VXI Sysfail:	supported
VXI Interrupts:	ROAK, programmable levels
VXI Local Bus:	Available
TTL Triggers	SYNC trigger protocol supported
Memory Requirements:	32 Kilobytes

The module's on-board PXI/cPCI bus complies with PCI Spec. 2.2 and PXI Spec. 2.0 for cPCI and PXI 3U or 6U modules.

PXI Bus Data Width:	32-bit
PXI Bus Speed	33 MHz
PXI Bus Voltage	5V or 3.3V (jumper selectable)
PXI Triggers	supported
PXI Clock 10	On-board 10MHz source

The modules on-board PMC bus complies with the PMC Specification IEEE P1386.1 for 32-bit PMC modules.

PMC Bus Data Width:	32-bit
PMC Bus Speed	33 MHz
PMC Bus Voltage	5V or 3.3V (jumper selectable)
PMC I/O:	64-pin Header

2.0 INSTALLATION

2.1 UNPACKING AND INSPECTION

In most cases the VX407C is individually sealed and packaged for shipment. Verify that there has been no damage to the shipping container. If damage exists then the container should be retained as it will provide evidence of carrier caused problems. Such problems should be reported to the carrier immediately as well as to Customer Support. (Contact information is available in the front few pages of this manual.) If there is no damage to the shipping container, carefully remove the module from its box and anti static bag and inspect for any signs of physical damage. If damage exists, report immediately to Customer Support.

2.2 HANDLING PRECAUTIONS

The VX407C contains components that are sensitive to electrostatic discharge. When handling the module for any reason, do so at a static-controlled workstation, whenever possible. At a minimum, avoid work areas that are potential static sources, such as carpeted areas. Avoid unnecessary contact with the components on the module.

2.3 INSTALLATION OF PXI/cPCI MODULES

PXI/cPCI modules must be installed before the VX407C is installed into the VXI system. To install modules, remove the VX407C's top shield and front panel covers as needed. ***There is never a need to remove the VX407C's bottom shield.*** Install PXI/cPCI modules by carefully sliding the module through the opening in the front panel of the VX407C and firmly pressing the connector on the PXI/cPCI module together with the connector on the carrier.

There are two mounting locations on the carrier: A and B. 3U PXI/cPCI modules may be installed into either location. 6U modules must be installed into location A. The mounting locations are illustrated in Figure 1.

WARNING: For most VXI systems it is required to remove the injector/ejector of the PXI/cPCI module in slot A. The guide pin on the PXI/cPCI injector/ejector prevents the module from properly fitting in most VXI chassis.

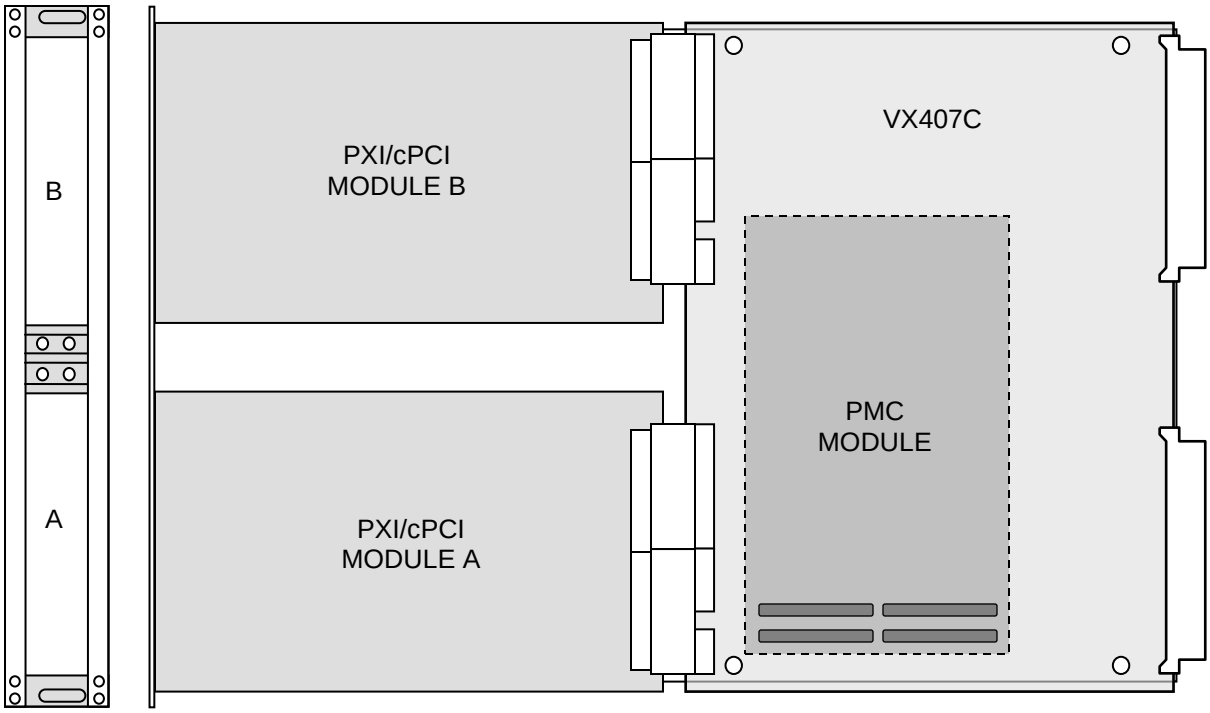


Figure 1. Front Panel and Top View (Top Shield Not Shown)

2.4 INSTALLATION OF PMC MODULES

PMC modules must be installed into the carrier before the carrier is installed into the host system. To install modules, remove the VX407C's top shield. ***There is never a need to remove the VX407C's bottom shield.*** Firmly press the connector on the PMC module together with the connector on the carrier as shown in Figure 2. Secure the module through the holes in the bottom shield using the original screws.

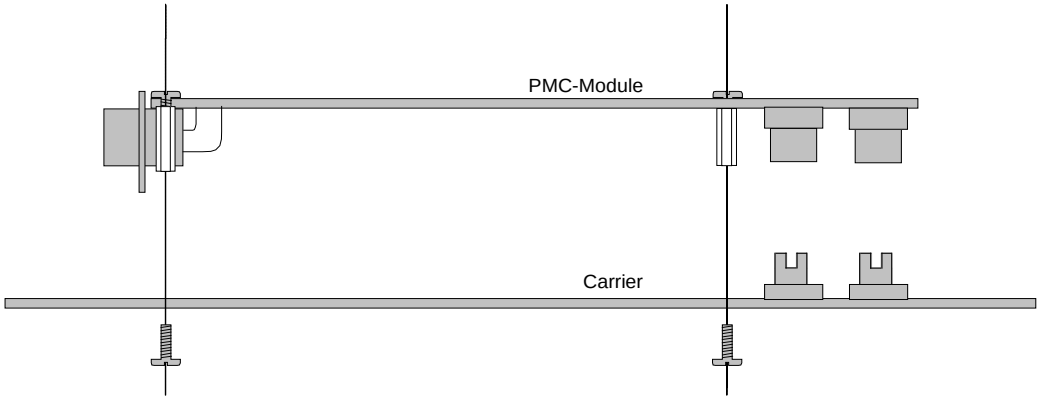


Figure 2. PMC Module Installation

2.5 INSTALLATION OF VX407C CARRIER

CAUTION: Read the entire User's Manual before proceeding with the installation and application of power.

If necessary, remove the top shield from the VX407C and configure the switches and jumpers. Set the module's logical address and addressing mode as described in section 3.4. Replace the shield and insert the carrier into the appropriate slot according to the desired priority and apply power. If no obvious problems exist, proceed to communicate with the module as outlined throughout the rest of this manual.

2.6 PREPARATION FOR RESHIPMENT

If the module is to be shipped separately it should be enclosed in a suitable water and vapor proof anti static bag. Heat seal or tape the bag to insure a moisture-proof closure. When sealing the bag, keep trapped air volume to a minimum.

The shipping container should be a rigid box of sufficient size and strength to protect the equipment from damage. If the module was received separately from a system, then the original module shipping container and packing material may be re-used if it is still in good condition.

3.0 FUNCTIONAL OVERVIEW

3.1 GENERAL

The VX407C provides an intelligent interface between the VXI bus and two 3U or one 6U PXI or CompactPCI(cPCI) modules. It features an embedded processor system powered by a Motorola MPC8245 integrated processor. The on-board PCI bus provides an interface to the two PXI/cPCI positions as well as one PMC module position and 16 kilobytes of shared memory. VXI interface logic provides an interface between the VXI bus and the PowerPC via the shared memory and the PowerPC's local bus.

The software architecture provides a flexible platform for the user applications to perform necessary tasks. On-board system utilities include: boot-up and initialization routines, system configuration routines, VXI communications routines, and various hardware interface routines to provide a basic interface to the carrier and installed PXI/cPCI modules and to assist application development. The system architecture also supports various commercially available real time operating systems.

3.2 HARDWARE OVERVIEW

The VX407C is powered by a highly integrated Motorola MPC8245 microprocessor with a PowerPC 603e core, a built-in Peripheral Component Interconnect (PCI) interface, and an advanced memory controller. The processor along with flash memory, ROM memory, and SDRAM form a complete embedded processing system with all the peripherals necessary for flexible application development.

Dual-ported shared memory and VXI interface logic allow for seamless communication between the VXI host and the PowerPC. Interrupts and handshaking logic is also provided to assist communications between the host and PowerPC.

A single PMC positions and two PXI/cPCI positions allow a variety of instruments and peripherals to be added to the system. Both the VXI host and the PowerPC can communicate with and control the modules.

Finally, relay driver logic allows special control hardware to be easily added to the overall integrated system. Figure 3 illustrates the system hardware architecture.

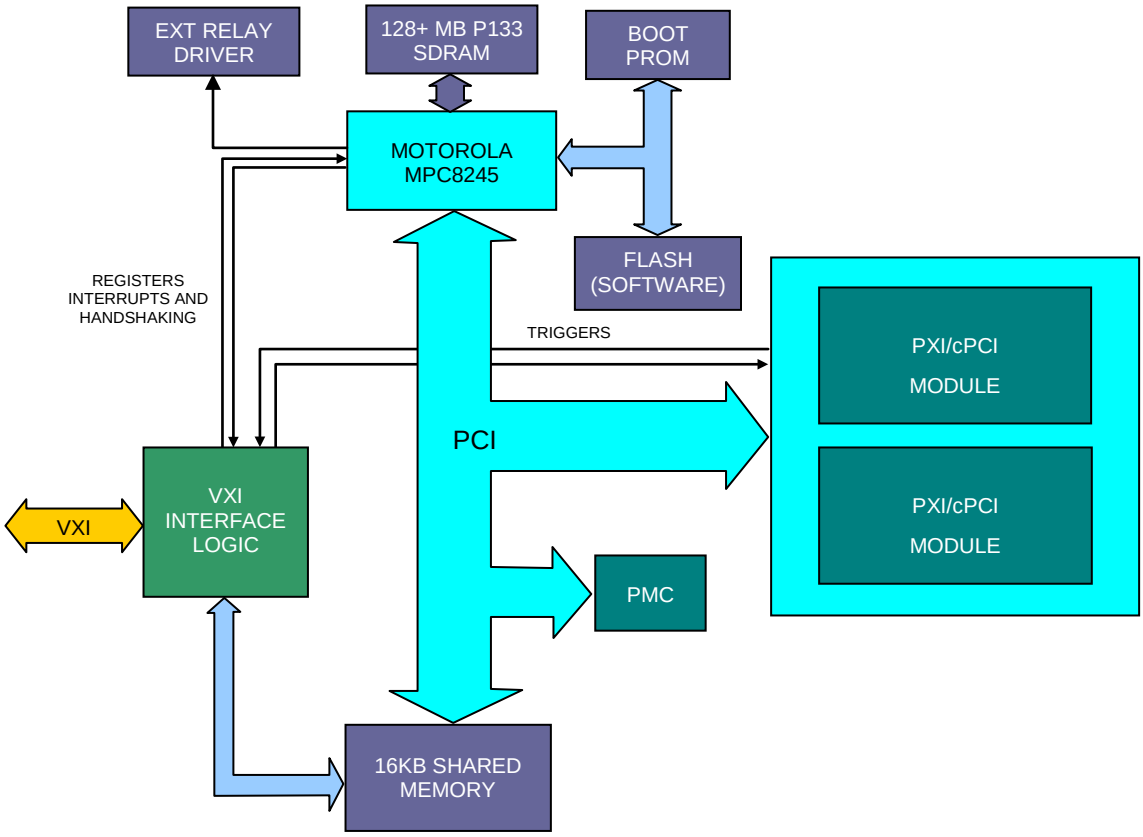


Figure 3. System Hardware Architecture

3.2.1 PXI/cPCI Modules

The PXI/cPCI modules provide the measurement and control functionality for the given application. The carrier can support up to two 3U or one 6U PXI or CompactPCI (cPCI) modules. The modules can be controlled over the on-board PCI bus by both the PowerPC application and the VXI host via the shared memory device. A variety of modules are commercially available from numerous manufacturers.

3.2.2 Shared Memory

The 16 kilobyte shared memory device acts as a buffer between the VXI bus and the on-board PCI bus. The device provides 16 kilobytes of dual-port SRAM and various other communications utilities such as general purpose FIFO's. It connects to the VXI bus through a local bus interface controlled by the VXI interface logic and to the PowerPC through the PCI bus. The device performs on-chip memory arbitration allowing the 16 kilobytes of memory to be accessed at the same time from both the VXI bus and the PowerPC. It also contains an embedded PCI bus controller allowing the VXI bus to directly access the on-board PCI bus and thus directly access the PXI/cPCI modules.

3.2.3 PowerPC and Peripherals

The PowerPC architecture was designed as a standard embedded processor system. It consists of a Motorola MPC8245 PowerPC, a boot ROM device, 128 megabytes of P133 SDRAM, and a flash memory device. This architecture allows the developer to select from standard off-the-shelf development tools and real-time operating systems for application development.

The PowerPC acts as the PCI bus master and can access both PXI/cPCI modules as well as the PMC module and the shared memory device. It also can access the VXI interface logic to perform handshaking between itself and the VXI bus.

3.2.4 VXI Interface Logic

The VXI interface logic acts as a transparent interface between the VXI bus and the shared memory device. It translates VXI bus accesses into shared memory local bus accesses by managing all local bus address and control lines. It maps all of the shared memory device's address space to VXI A24/A32 space.

The VXI interface also handles handshaking between the PowerPC and the VXI bus. It includes a set of registers that are mapped to VXI A16 space and are accessible by the PowerPC to handle host to device communications and handshaking.

Finally the interface logic provides VXI bus trigger and interrupt capabilities. The carrier has extensive mapping capabilities between the PXI trigger lines and the VXI bus trigger lines controllable by the VXI host. VXI interrupts can be generated by the PowerPC application on any of the 8 VXI interrupt levels.

3.2.5 PMC Slot

The PMC slot on the PCI bus can be used to add additional functionality not provided by the carrier. For example, a mass storage device could be added for on-board data collection by installing a PMC disk drive controller. The PMC position is accessible by both the PowerPC and the VXI host.

3.2.6 External Drivers

The PowerPC can control a Darlington sink driver device residing on its local memory bus. The device's outputs are available at a 16 pin header for external use. The device is intended to drive external relays, display LED's, or other high current devices.

3.2.7 JTAG/COP Interface

The JTAG interface to the PowerPC provides a debug and development interface supported by many standard off-the-shelf developments tools. The interface is used by development tools to communicate with the processor. It provides the developer with the ability to view system registers, view memory, set breakpoints, and use other standard debugging practices.

3.3 SOFTWARE OVERVIEW

The embedded software on the carrier as well as the host software are very application dependant and thus, must be developed specifically to suit the needs of the particular application. However, on-board system utilities are provided to assist application development and to provide basic functionality when no application exists.

When no application exists, the system software provides basic functionality allowing the user to communicate with the carrier and the PXI/cPCI and the PMC modules. A limited set of VXI message based commands are available as well as the ability to access all defined registers and the shared memory. In this capacity, the carrier can operate as a fully functional instrument without the existence of a user application.

If a user application is to be provided, the on-board system utilities assist the developer in performing several tasks that require advanced knowledge of the carrier architecture and the devices that make up that architecture. For example a system routine is provided to program the flash memory so that the developer does not need to refer to the flash device's data sheet to learn the programming protocol. Also, the on-board system utilities automatically handle the communications required for VXI message passing so that the application can concentrate on performing high level tasks and not on the details of the VXI word serial protocol. The on-board system utilities are completely independent and fully interrupt and exception driven so they only take up a very small amount of processor resources and so the user application can be independently compiled and linked without knowledge of the system utilities' memory organization.

3.4 HARDWARE CONFIGURATION

There are several switch and jumper selectable settings that configure the VX407C for operation. Configuration options include: the VXI logical address, PowerPC options, programming modes, and operational voltages. Figure 4 shows the layout of all the switches and jumpers on the VX407C.

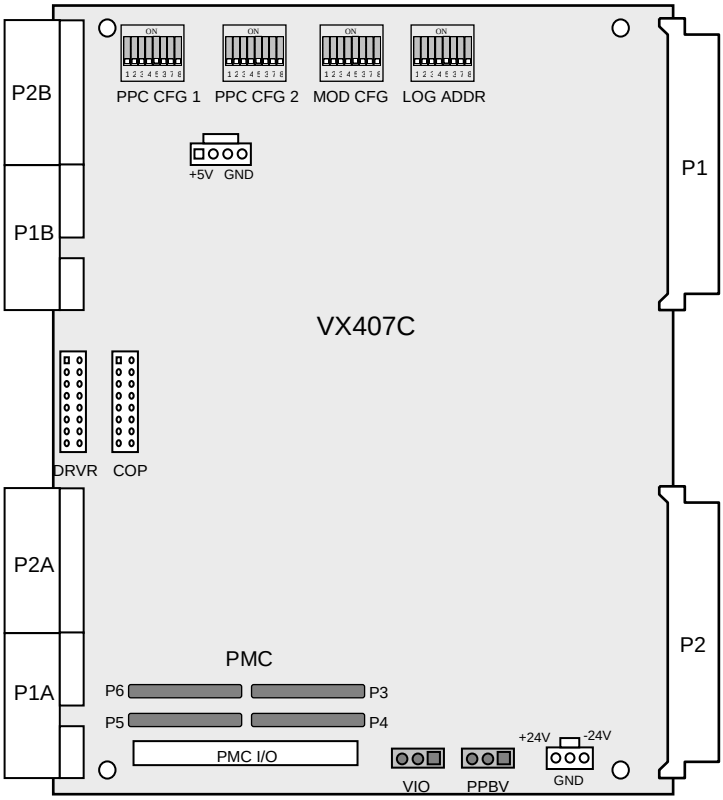


Figure 4. Hardware Layout

3.4.1 Logical Address Switch

The logical address switch specifies the logical address for the VX407C. The switches form a binary weighted decimal value that sets the logical address of the module. The OFF position for each switch represents a binary one in that bit position. For example, the switch settings shown in Figure 5 would result in a logical address of 36.

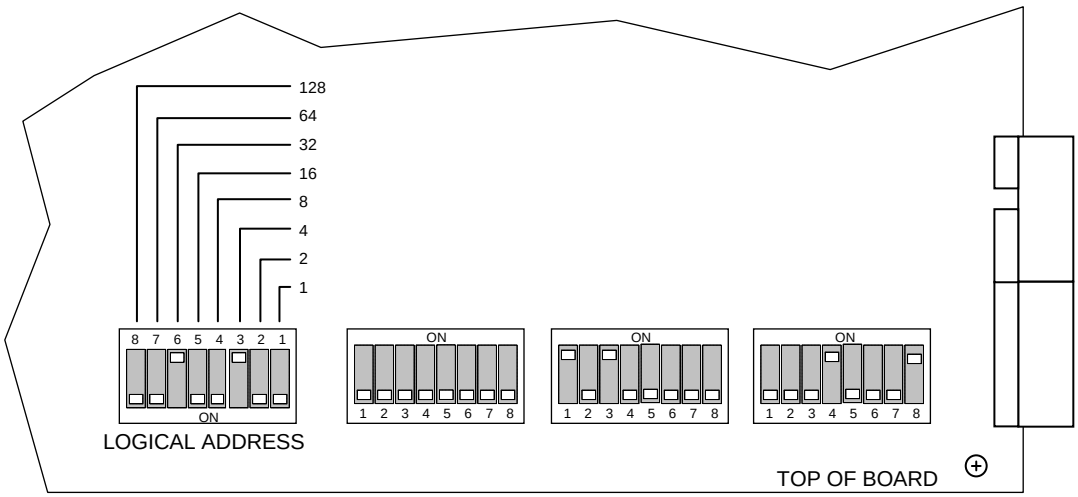


Figure 5. Logical Address Configuration Switch

3.4.2 Module Configuration Switch

The module configuration switch is used to set some of the miscellaneous options on the VX407C. Figure 6 shows the options that are configurable with this switch.

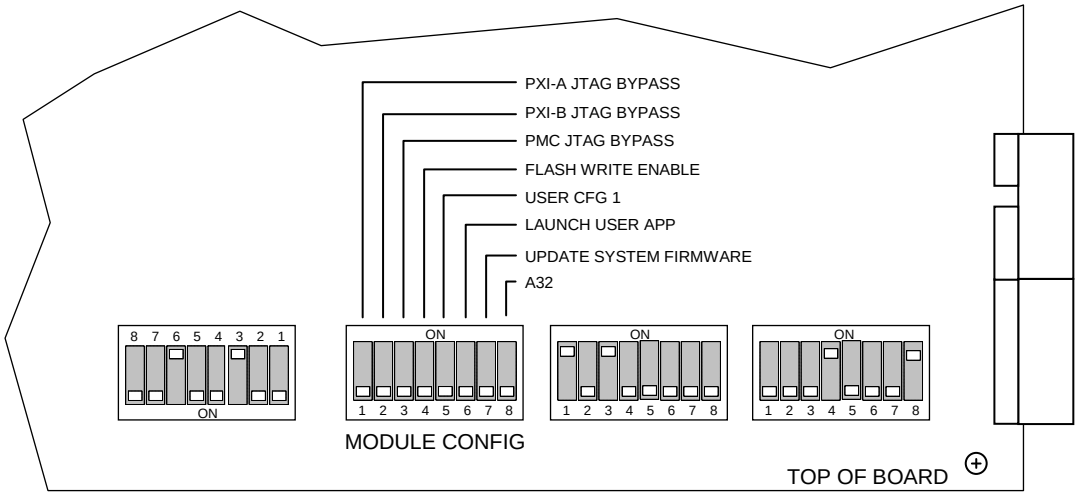


Figure 6. Module Configuration Switch

JTAG Bypass Switches: These switches specify whether or not to bypass a PCI device's JTAG interface. The on-board JTAG bus is daisy chained between the PowerPC, the JTAG/COP debug header, the PXI devices, and the PMC device. The daisy chain must not be broken in order for JTAG operation to work correctly, including the JTAG debug operation. Therefore, if a PXI or PMC slot is vacant or the installed PXI or PMC device

does not provide the JTAG bypass, the corresponding switch must be set to ON as to not break the daisy chain.

Flash Write Enable Switch: This switch will enable or disable the ability for software to program flash memory. Setting this switch to ON will enable the flash programming capability. When the flash programming capability is disabled, the carrier configuration options cannot be modified. See section 5.2.2 for details on configuration options.

User Configuration Switch: The user configuration switch is available for use by a user application. The switch has not effect on the hardware operation of the VX407C. The value of the switch is copied to the operations registers so that the user application can read the value and use it however it wishes. Setting the switch to ON results in the corresponding bit of the operations register being set to a binary '0'.

Launch User Application Switch: This switch determines whether or not the initialization firmware will attempt to launch a user application or will launch the on-board system application. The on-board system application will allow the user to perform configuration routines and basic instrument communications. If this switch is set to OFF the firmware will attempt to launch the user application. Should a situation arise where the user application will not launch or the VX407C crashes on boot up this switch should be set to ON so basic configuration and debug can be performed.

Update System Firmware Switch: This switch puts the VX407C into a mode where the on-board system utilities can be updated. This is the only function that can be performed in this mode. If this switch is set to OFF at power-up, the carrier will go into the firmware update mode and wait for an update to complete. If this switch is ON at power-up, the carrier will initialize normally.

A32 Switch: This switch selects whether the VX407C performs VXI A24 or A32 address decoding. This address space is used to access the shared memory device. If this switch is set to ON, the carrier requests memory in the systems A32 address space, otherwise A24 address space is requested.

3.4.3 PowerPC Configuration Switches

The PowerPC configuration switches determine the value of the corresponding signals during reset. Each signal connected to these switches is a reset configuration signal for the PowerPC. The values of these signals at reset determine the configuration of the processor. Figure 7 shows all available PowerPC reset configuration options. Table I briefly describes each option and all possible settings. ***These switches are preset during manufacturing to the optimal settings for the VX407C. Modifying these settings is rarely necessary and in some cases may cause the VX407C to not function correctly.*** For details on reset configuration refer to the MPC8245 User's Manual.

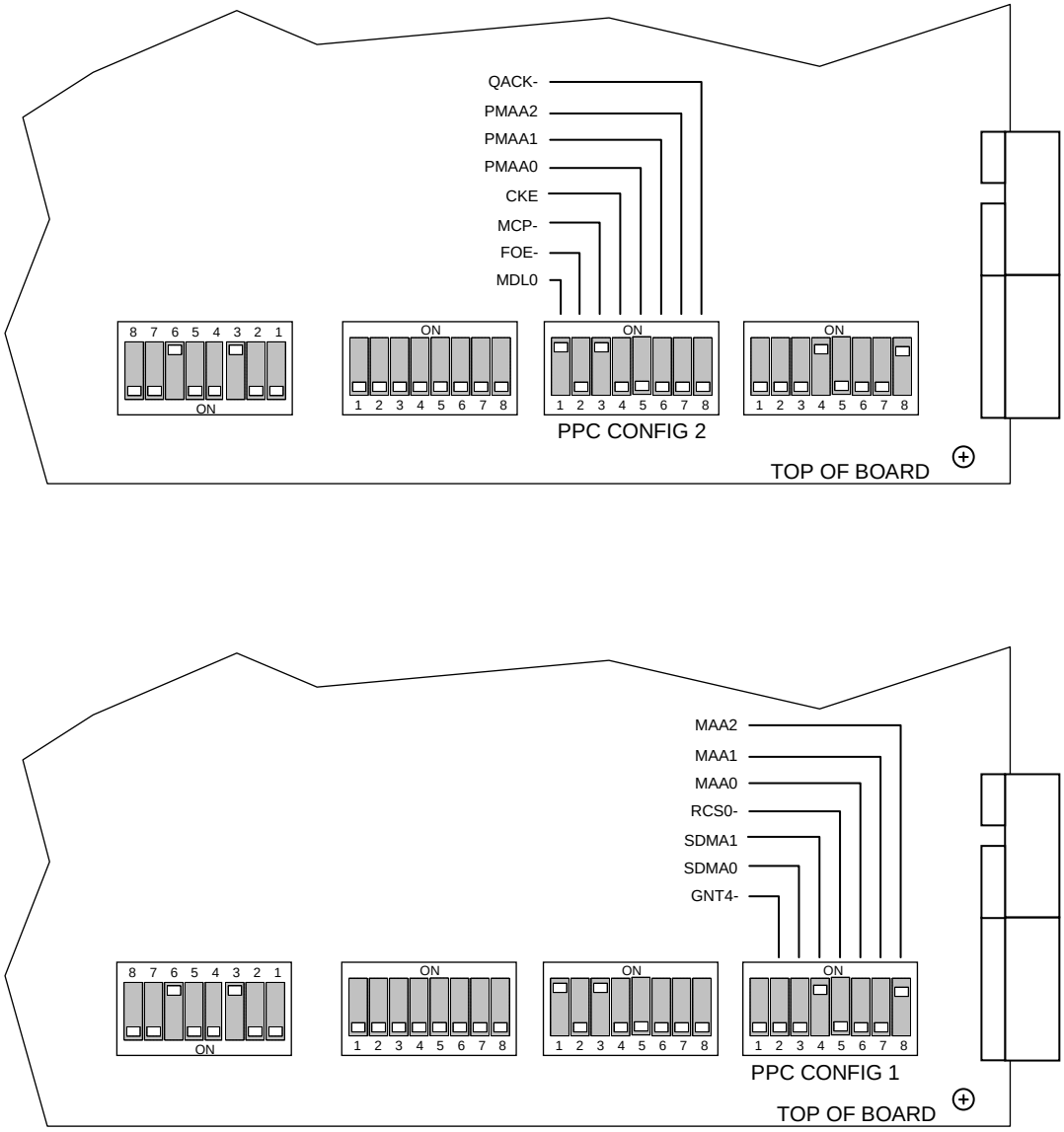


Figure 7. PowerPC Configuration Switches

Table I. PowerPC Configuration Signals

Signal	Description	Settings
MDL0, FOE-	Selects the data bus width for ROM bank 0 and SDRAM	(MDL0 = 0, FOE_ = 0) = ROM 32-bit, SDRAM 32-bit **(MDL0 = 0, FOE_ = 1) = ROM 8-bit, SDRAM 32-bit[1,2]** (MDL0 = 1, FOE_ = 0) = ROM 64-bit, SDRAM 64-bit (MDL0 = 1, FOE_ = 1) = ROM 8-bit, SDRAM 64-bit
MCP-, CKE	Sets the PCI output hold delay value (in nanoseconds) relative to PCI_SYNC_IN. Refer to the MPC8245 documentation for details on each setting.	(MCP_ = 0, CKE = 0) **(MCP_ = 0, CKE = 1) Recommended for 33 MHz PCI[1,2]** (MCP_ = 1, CKE = 0) (MCP_ = 1, CKE = 1) Recommended for 66 MHz PCI
PMAA0, PMAA1	Memory signal driver capabilities.	(PMAA0 = 0, PMAA1 = 0) = reserved (PMAA0 = 0, PMAA1 = 1) = 40 Ω drive capability (PMAA0 = 1, PMAA1 = 0) = 20 Ω drive capability **(PMAA0 = 1, PMAA1 = 1) = 6 Ω drive capability[1,3]**
PMAA2	PCI and EPIC controller driver capabilities	**0 = 40 Ω drive capability[1,2,3]** 1 = 20 Ω drive capability (except for IRQ2/S_RST and IRQ3/S_FRAME- signals which have 6 Ω drive capability)
QACK-	Clock Flip Disable	0 = Clock flip enabled **1 = No clock flip[1,2]**
GNT4-	Debug Address Disable	0 = Debug address enabled **1 = Debug address disabled[1,2]**
SDMA0	DUART Signals Disabled	0 = DUART signals enabled **1 = PCI_CLK[0:3] signal used instead of DUART[1,2]**
SDMA1	Extended Addressing Mode	**0 = Extended addressing mode enabled[1,2]** 1 = Extended addressing mode disabled
RCS0-	Boot Memory Location	0 = Boot ROM is located on the PCI bus **1 = Boot ROM is located on the local bus[1,2]**
MAA0	Address Map Setting. The MPC8245 only supports address map B.	0 = Invalid **1 = MPC8245 is configured for address map B[1,2]**
MAA1	PCI Host Mode	0 = MPC8245 is a PCI agent device **1 = MPC8245 is a PCI master device[1,2]**
MAA2	PCI Arbiter Disable	**0 = PCI arbiter enabled[1,2]** 1 = PCI arbiter disabled

Notes: 1. Bold indicates the recommended setting for the VX407C
2. 1=Switch OFF, 0=Switch ON (except for PMAA2 see note 3)
3. For the PMAA2 switch, 1=Switch ON, 0=Switch OFF

3.4.4 VIO Configuration Jumper

The VIO configuration jumper selects the voltage level supplied to the VIO pins on the PXI/cPCI connectors. The VIO power signals are used by universal CompactPCI modules that can operate in both +5V and +3.3V systems. On these boards, the power

for the I/O buffers is provided by the VIO pins instead of directly from the +3.3V or +5V power pins. Set the jumper according to the PXI/cPCI modules installed on the carrier as shown in Figure 8.

Figure 8. VIO Configuration Jumper

3.4.5 PPBV Configuration Jumper

The Processor PCI Bus Voltage (PPBV) configuration jumper selects the PCI bus voltage level. This is the active level at which the PowerPC will drive its PCI bus signals. The voltage level should be set according to the PXI/cPCI and PMC modules installed on the carrier. This voltage level will normally correspond to the VIO voltage level setting described in section 3.4.4. Figure 9 shows the PPBV configuration jumper settings.

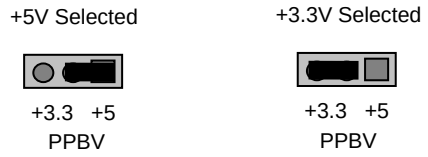


Figure 9. PPBV Configuration Jumper

3.5 CONNECTORS

The VX407C incorporates several connectors to provide a physical connection to its various interfaces. Figure 4 shows the general location of each connector on the VX407C. Detailed pin-out information can be found in Appendix A. A short description of each connector is provided in the following sections.

3.5.1 External Power Connectors

Two connectors are provided to connect +5V, +24V and -24V externally. The +5V connection is provided at a Molex 70543 male 4-pin connector. The +24V and -24V connections are provided by a Molex 70543 male 3-pin connector. Refer to Appendix A for details on the header pin-outs.

3.5.2 External Drivers Connector

The external driver's output signals are available at a 16-pin header (8x2 with 0.100 inch centers). Refer to Appendix A for details on the header pin-outs.

3.5.3 JTAG/COP Connector

Connection to the JTAG/COP debug interface is provided through a keyed 16-pin header (8x2 with 0.100 inch centers). This header is the standard size and employs the standard pin-out used by most JTAG based emulators. The pin-out details of the JTAG/COP header can be found in Appendix A.

3.5.4 PMC Connectors

The four PMC connectors provide the physical interface to a PMC module. The connectors are configured in accordance with the PMC specification. Refer to Appendix A for pin-out details.

3.5.5 PMC I/O Connector

Some PMC modules provide 64 bits of I/O to the PMC carrier board though the PMC rear connectors. On the VX407C these 64 bits of I/O are available at the PMC I/O connector. The connector is a standard 64 pin header (32x2 with 0.100 inch centers). Refer to Appendix A for pin-out details.

3.5.6 VXI Connectors

The rear connectors, labeled P1 and P2, provide the physical interface to the VXI system. They are configured in accordance with the VXI specification. Refer to Appendix A for pin-out details.

3.5.7 PXI/cPCI Connectors

The four PXI/cPCI connectors provide the physical interface for two 3U or one 6U PXI/cPCI modules. The connectors are configured in accordance with the PXI specification. Refer to Appendix A for pin-out details.

4.0 SYSTEM ARCHITECTURE

4.1 OVERVIEW

The system architecture illustrated in Figure 3 is viewed differently from an application running on the embedded PowerPC than from an application running on the VXI host. Most of the carrier's hardware can be accessed by both applications but, the methods for doing so differ. The system architecture is best described by viewing the host-side and the device-side separately. However, it is also important to understand how the resources shared between both applications are used for host to device communications.

4.2 DEVICE-SIDE ARCHITECTURE

The device-side architecture is anchored by a standard embedded processor system powered by the MPC8245 PowerPC. The architecture provides on-board RAM, boot ROM, and flash memory to support the software application. The PowerPC acts as the PCI bus master and has full access to all devices on the PCI bus. A set of operational registers and the external relay driver are available to the application via the processor's local bus. Figure 10 illustrates the device-side architecture.

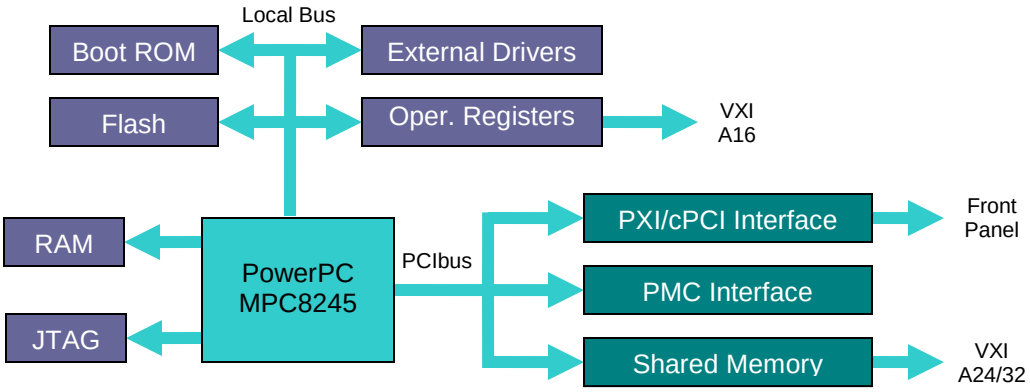


Figure 10. Device-Side Architecture

4.2.1 PowerPC Memory Map

Being a 32-bit processor, the MPC8245 can address up to 4 Gigabytes of physical memory. On the VX407C, the processor maps this 4 Gigabytes of memory into a configuration designated as Address Map B. The address map B configuration divides the memory space into sections that, when accessed, translate the operation to a local memory, PCI memory, PCI I/O, or ROM access. Figure 11 shows the general layout of address map B.

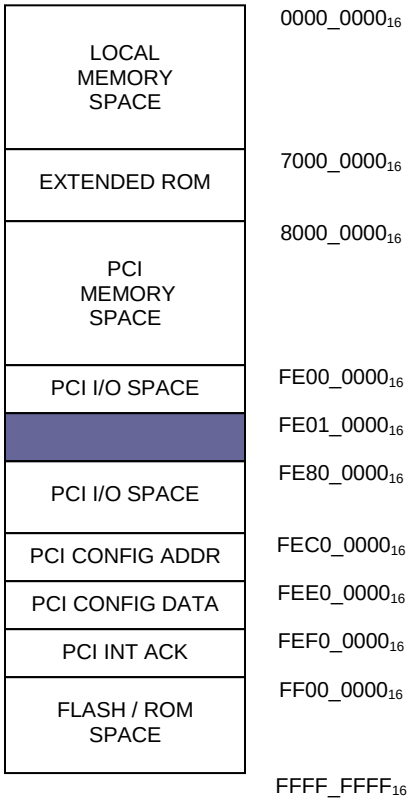
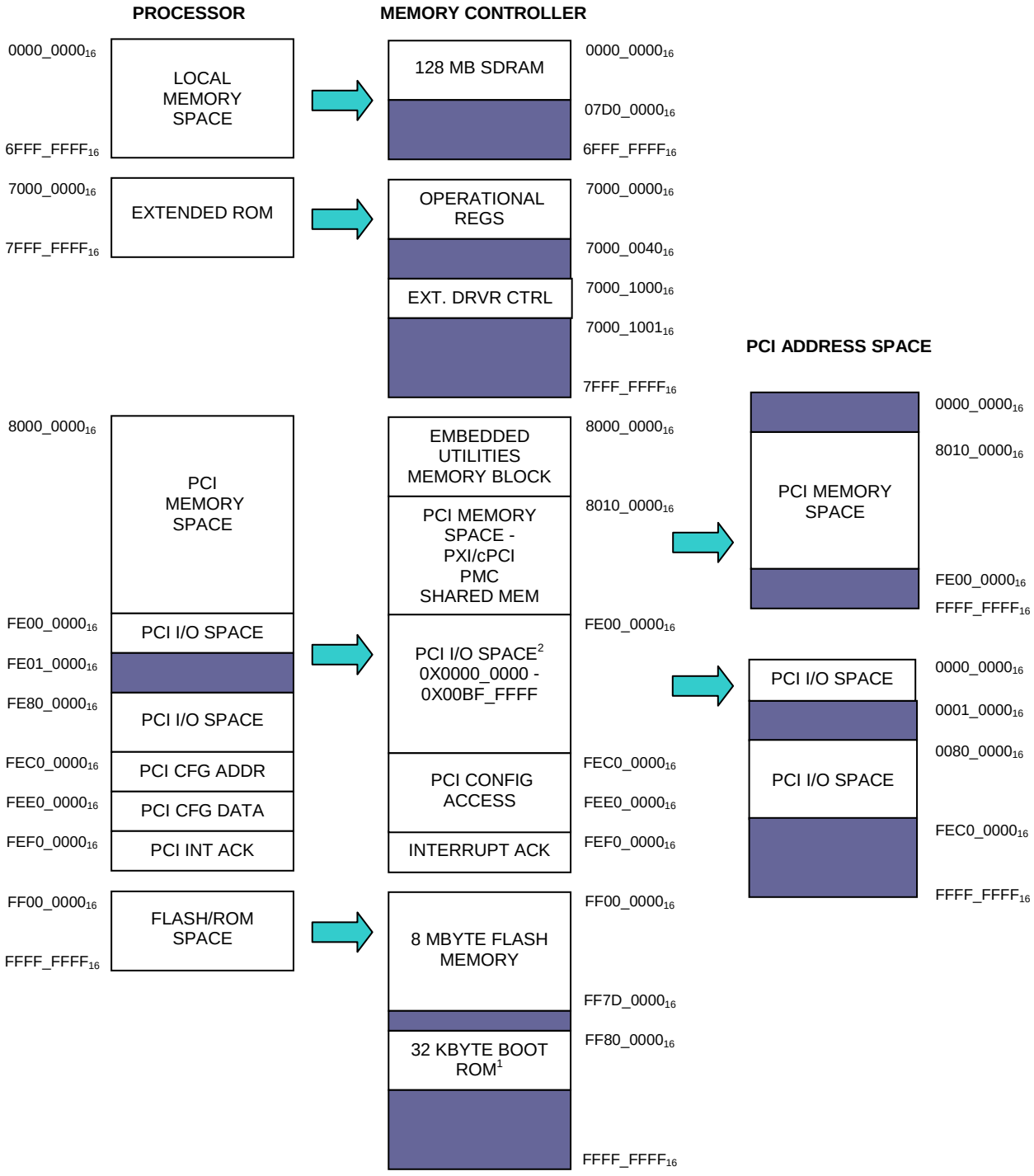


Figure 11. Address Map Overview

Details of the address map B implementation for the VX407C architecture are shown in Figure 12. Each section in the address map directly addresses a resource in the system architecture. Further details of each address block are provided throughout this document. Other address map B options and settings are also available but generally not used on the VX407C. Refer to the MPC8245 User's Manual for details.



Notes:

1. The boot ROM device only decodes 15 address lines. Therefore, the boot ROM is repeated throughout the address space. For example, address $FF80\_0000_{16}$ is the same location as $FFF0\_0000_{16}$.
2. PCI I/O accesses are forwarded to the PCI bus with the 8 most significant bits of the address cleared. (i.e. processor address $FE80\_0000_{16}$ = PCI I/O address 0080_0000_{16})

Figure 12. Detailed PowerPC Address Map

4.2.2 SDRAM

The SDRAM provides 128 Megabytes of temporary storage for the application. The memory is organized in a 13 rows x 10 columns x 4 banks configuration. It has a 10ns access time and a 32 bits wide data bus. It is accessed through the PowerPC's addresses space starting at offset 0.

4.2.3 Boot ROM

The boot ROM provides 64 kilobytes of non-volatile, read-only memory. It is normally programmed during the manufacturing process to contain boot code and initialization routines. It can not be reprogrammed in circuit. The boot ROM is mapped to PowerPC address $FF80\_0000_{16}$ and has an 8-bit data bus. Only 15 address bits are decoded so that the 64 kilobytes are repeated throughout the PowerPC's ROM/Flash space between addresses $FF80\_0000_{16}$ and $FFFF\_FFFF_{16}$. Consequently, the default exception vector table starting at address FFF0_0000h resides in the boot ROM device.

4.2.4 Flash Memory

The flash device provides 8 megabytes of non-volatile storage for code and data. Unlike the boot ROM, flash is programmable in circuit and may be used by a user application. However, the first sector is normally reserved for use by the system firmware. The flash device is accessed starting at PowerPC address $FF00\_0000_{16}$ and has an 8-bit wide data bus. Reads from flash are performed as standard PowerPC memory accesses. Programming and erasing the device, however, requires a sequence of commands to be sent to the device. System utilities are provided with the on-board system routines for programming the flash device. For details on using the on-board system utilities refer to section 5.2.

4.2.5 PCIbus Architecture

The on-board PCI bus can contain up to 5 devices including the PowerPC which acts as the bus master. If a PCI to PCI Bridge is added at any of the PCI interfaces, more devices are available and can be accessed by the PowerPC. The devices on the primary bus include the PowerPC, the shared memory device, a PMC device, and the PXI/cPCI devices. The bus operates at 33 MHz and 5V or 3.3V (jumper selectable).

PCI memory, configuration, and I/O space is memory mapped directly into the PowerPC's address map as shown in Figure 12. Approximately 2 gigabytes of PCI memory space is mapped starting at address 8010_0000_{16}. Each device requiring memory will have a base address within this mapped area. About 4 megabytes of PCI I/O space is mapped to PowerPC addresses $FE00\_0000_{16}$. When performing a PCI I/O access, the processor clears the upper 8 bits of the address before forwarding the transaction to the

PCI bus. So, for example, accessing processor address $FE80\_0000_{16}$ will read or write PCI I/O address 0080_0000_{16}. The base address of each device is determined by the PCI enumeration routines during initialization. The base address of a particular device can be determined by reading its Base Address Register (BAR) register in PCI configuration space for that device.

To perform a single PCI configuration write or read, two processor accesses are required. First, the PCI configuration address register at PowerPC address $FEC0\_0000_{16}$ must be set to point to the correct device and offset. Then the data can be read from or written to the PCI configuration data register at PowerPC address $FEE0\_0000_{16}$. The PCI configuration address register value is determined by the device number, IDSEL signal routing, device function number, and the register offset. For details on performing PCI configuration accesses refer to the MPC8245 User's Manual. System routines are provided that an application can use to perform configuration reads and writes. For details on using the on-board system utilities for configuration accesses, refer to section 5.2.

4.2.5.1 PCIbus Enumeration

During initialization, the boot-up firmware will search the PCI bus for devices, determine the resources needed for the device, and allocate the resources accordingly. This procedure determines where in the PowerPC memory map a particular PCI device's memory space is located. PCI device mapping is not guaranteed from one carrier configuration to another or even from one firmware version to another. The application software should always check a device's configuration registers for memory mapping information prior to accessing the device.

4.2.5.2 IDSEL Signal Routing

Each device on the PCI bus has a unique ID select line used to specify the destination of a configuration access. The PCI specification does not stipulate the source of each ID select line; however, the upper 16-bits of the address bus are normally used. On the VX407C each device has its IDSEL line tied to a specific address line as shown in Table II. The device number, normally provided to software routines, is also system dependant. Table II also shows the device numbering used on the VX407C. This information must be incorporated into a configuration access by the application when performing a write or read. The system routines for configuration access automatically incorporate this information into the access.

Table II. IDSEL Signal Routing

Device	IDSEL	DevNum
• Shared Memory	AD16	16_{10}
• PowerPC	AD17	17_{10}
• PMC	AD18	18_{10}
• PXI Slot A (bottom slot)	AD31	10_{10}
• PXI Slot B (top slot)	AD30	30_{10}

4.2.5.3 PCI Interrupts

The PowerPC's Embedded Programmable Interrupt Controller (EPIC) acts as the PCI interrupt controller. The interrupt lines from the PCI devices are routed to the EPIC controller's five interrupt inputs as shown in Table III. The PCI specification requires that single function devices only use interrupt pin INTA#. Thus, in most cases, each device on the VX407C's PCI bus will have a unique interrupt line to the EPIC controller. Only in the rare case where a multi-function device is to be used will interrupt line sharing be required. Refer section 6.5.1 of this document and the MPC8245 User's Manual for information on programming the EPIC controller to handle interrupts.

Table III. PCI Interrupt Signal Routing

Device	PCI Interrupt Pin	EPIC IRQ
PXI-A (Bottom Slot)	INTA# INTB# INTC# INTD#	**IRQ1** IRQ2 IRQ3 IRQ0
PXI-B (Top Slot)	INTA# INTB# INTC# INTD#	**IRQ2** IRQ3 IRQ0 IRQ1
PMC	INTA# INTB# INTC# INTD#	**IRQ3** IRQ0 IRQ1 IRQ2
Shared Memory	INTA#	**IRQ4**

Note: Bold indicates the EPIC interrupt line unique to the device when no multi-function devices are in use.

4.2.5.4 Shared Memory Device

The shared memory's entire address space is mapped to PCI memory space including all operations registers, the I_2O messaging unit, and the general purpose shared memory. The offset into PowerPC memory space is determined at boot up by the PCI enumeration software.

Figure 13 shows the shared memory device's address map. All addresses are offsets from the devices base address. To determine the shared memory's base address perform a PCI configuration read of offset 10_{16} (BAR0 Register) of the shared memory's configuration space. This value is the base address of the shared memory device. The general purpose shared memory begins at an offset of 4000_{16} from this address.

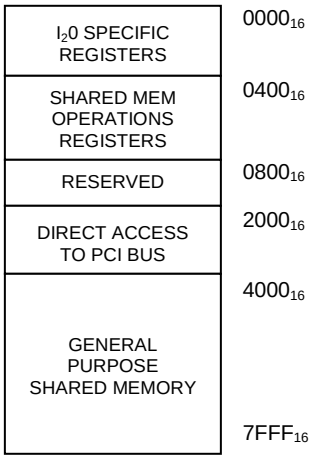


Figure 13. Shared Memory Organization

4.2.5.5 PXI/cPCI Devices

The two PXI/cPCI positions reside on the on-board PCI bus. Any memory or I/O space required by a device connected to one of these two slots is mapped to the PowerPC's address space. These devices may also contain a PCI to PCI Bridge in which case devices on the PXI/cPCI module's secondary bus will also be mapped to the PowerPC's address space.

PXI defined signals not specified by the PCI or CompactPCI specifications are also provided by the carrier. The PXI TTL triggers can be mapped directly to any of the VXI TTL triggers as discussed in section 4.2.6. The carrier also includes a 10 MHz clock source to provide the PXI_CLK_10 signal to the PXI connectors. Start trigger operation is supported with slot B (top slot) pin-outs configured to accept a PXI start trigger module.

4.2.5.6 PMC Device

The PMC position resides on the on-board PCI bus. Its address space is mapped directly to the PowerPC's address map. The bus mode signals are implemented to inform the PMC module of the PCI bus configuration. The PMC module may contain a PCI to PCI Bridge whose secondary bus is fully accessible by the PowerPC. Interrupts from the PMC device are supported as described in section 4.2.5.3.

4.2.6 Triggers

The carrier includes two programmable switching matrices for mapping PXI triggers to VXI triggers. The matrices are illustrated in Figure 14. The input matrix operates in the direction from the VXI system to the PXI module while the output matrix operates in the direction from the PXI module to the VXI system. In each direction the trigger can be enabled or disabled and inverted.

This architecture provides enormous flexibility allowing a large number of trigger mapping combinations. Programming of the matrices is provided via the operations registers discussed in section 4.2.7 and detailed in section 4.4.1.

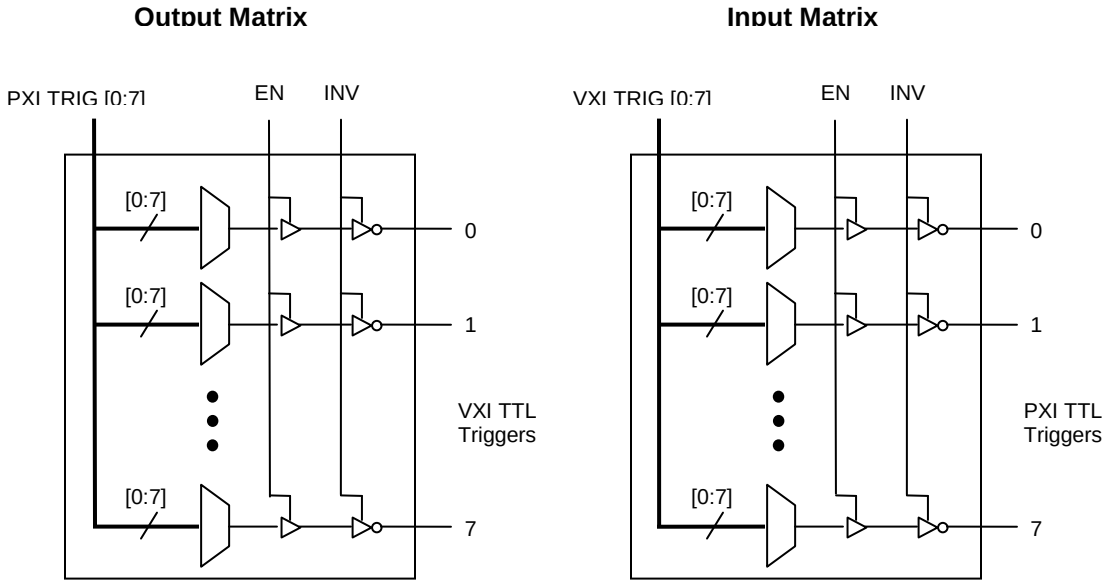


Figure 14. Trigger Architecture

4.2.7 Operations Registers

A set of operations registers are defined to allow the user application or the host application to perform certain operations on the VX407C such as PXI to VXI trigger mappings. The standard set of VXI defined registers are also part of the operations registers. The PowerPC has access to these operations registers via the local bus. These registers are mapped to the PowerPC's extended ROM space starting at address 7000_0000_{16}. The data bus width between the PowerPC and the operations registers is 8-bits wide and reads and writes are performed as standard memory accesses. For register definitions in this space refer to section 4.4.1.

4.2.8 External Drivers

The architecture includes an 8-bit Darlington sink driver device residing on the PowerPC's local memory bus. The device is intended to drive external relays, display LED's, or other devices with high current requirements. The device's outputs are available at a 16 pin header for external use. Refer to section 3.5.2 and Appendix A for details on the header

Access to the device is provided at address 7000_1000_{16}. The data bus width to the device is 8-bits wide and each bit corresponds to one of the 8 channels. The device can only be written to. Figure 15 shows the external driver control register.

External Driver Control

7000_1000 Bit	7	6	5	4	3	2	1	0
Write	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Read	(write-only)							

CHx ⇨ Channel value (1 = driven, 0 = not drivern)

Note: On VX407C Revision A modules, reading this register will cause random data to be written to the relay driver.

Figure 15. External Driver Control Register

4.2.9 JTAG/COP Interface

The JTAG interface to the PowerPC provides support for several standard off-the-shelf developments tools. Most development environments for the PowerPC support JTAG based communications with the processor. It provides the developer with the ability to view system registers, view memory, set breakpoints, and use other standard debugging practices.

Connection to the JTAG/COP interface is provided through a standard 16 pin header. Refer to section 3.5.3 and Appendix A for details on the header.

4.3 HOST-SIDE ARCHITECTURE

The host-side architecture is anchored by the VXI system including a VXI chassis and a host computer. Standard off-the-shelf VXI controllers from several different manufacturers are available to interface the carrier to the host computer, including high performance embedded controllers. The VXI host has access to the entire address space of the shared memory device as well as to a set of the operations registers. The shared memory device provides a utility to directly access devices on the PCI bus from the VXI host. Therefore, the VXI host application can control on-board PCI devices without the assistance of the PowerPC. The standard VXI registers required by the VXI specification are implemented as part of the operations registers. These include the registers required to implement the VXI word serial protocol. The VXI host has the ability to fully access all devices on the on-board PCI bus and to fully utilize all host to device communications utilities. Figure 16 illustrates the intelligent carrier architecture as viewed from the VXI host computer.

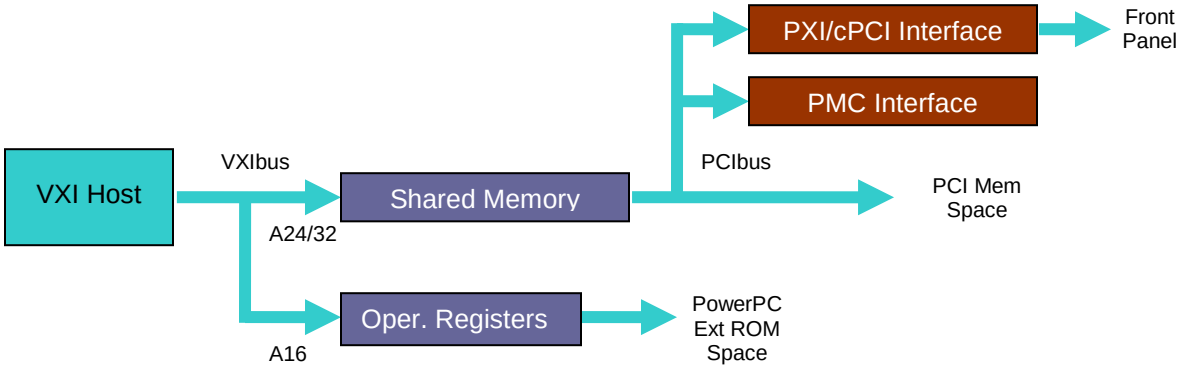


Figure 16. Host-Side Architecture

4.3.1 VXI Memory Map

Figure 17 shows the host-side memory organization for the intelligent carrier. The operations registers are accessed via VXI A16 space. These registers include the VXI required registers and the VXI message based communication registers as defined by the VXIbus specification.

A24/A32 memory space is a direct mapping of the shared memory device's memory map. This architecture gives the host full access to the shared memory and its registers including direct access to the PCI bus and other miscellaneous communications utilities.

A24 or A32 addressing is switch selectable as described in section 3.4.2. The VXI resource manager will write a base address to the offset register at address 06_{16} in A16 space. If the carrier is configured for A32 addressing the carrier will use the value of the offset register as the upper 16 bits of its 32-bit base address. If A24 addressing is selected the carrier will use the value in the offset register as the upper 16 bits of its 24-bit base address. This behavior is illustrated at the bottom of Figure 17.

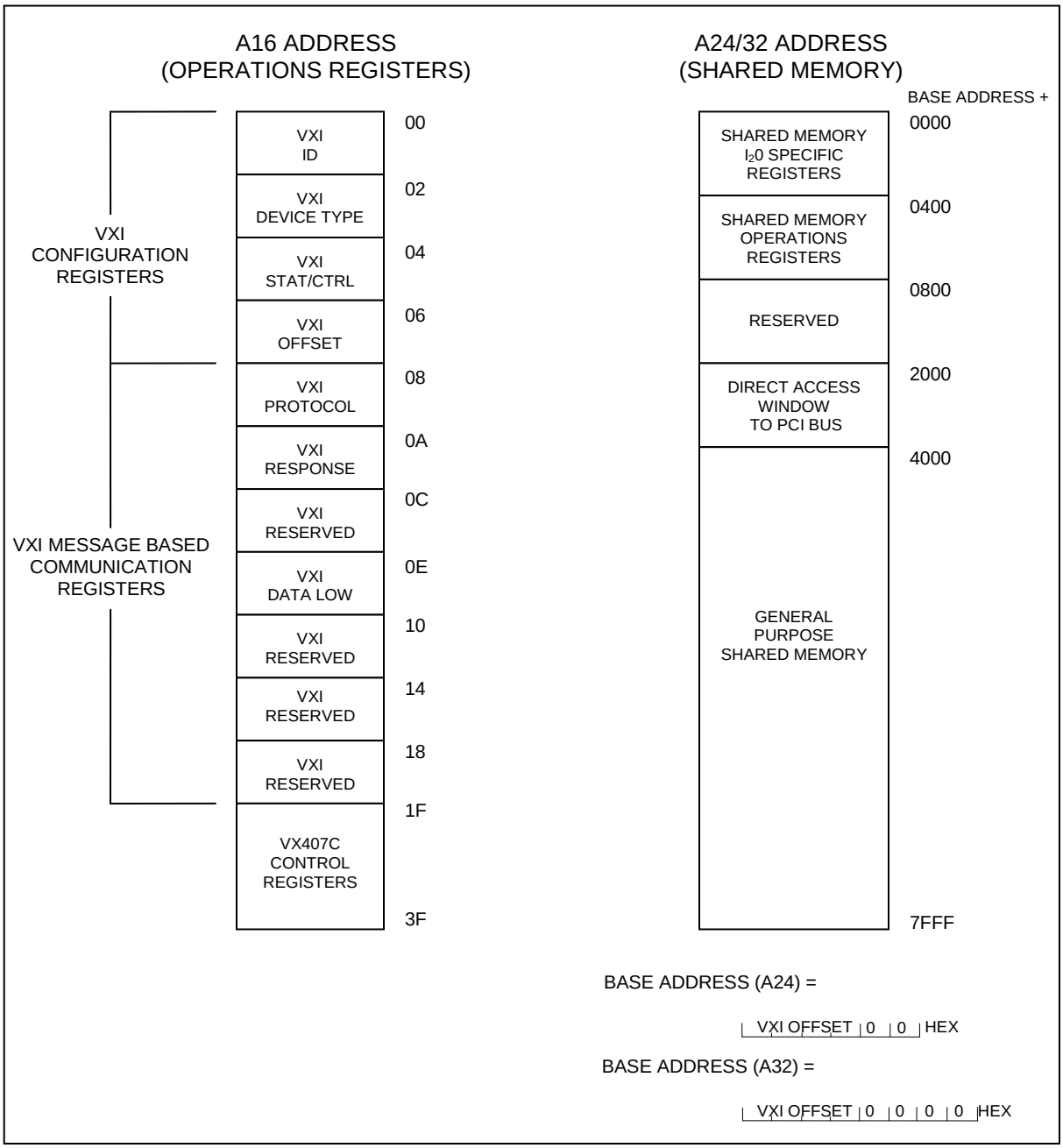


Figure 17. VXI Memory Organization

4.3.2 Data Bus Width

The intelligent carrier supports 16 and 32-bit wide data transactions to the shared memory device in the A24/A32 address space. However the device must be configured to be either 16 or 32-bits, but not both. Differences in the byte lane assignments between the VXI bus and the shared memory's local bus make dynamically switching between D16 and D32 impossible.

A system command is provided to configure the data bus width to the shared memory device. Refer to section 5.2.7 for details on the configuration command. A system configuration option is also provided allowing the data bus width setting to be initialized to the desired value at reset. Refer to section 5.2.2 for details on configuration options.

Only 16-bit accesses to A16 address space are supported.

4.3.3 PCI Bus Mastering and Direct Access

The shared memory device provides an 8 Kilobyte window directly into PCI memory space. This window is accessible by the VXI host at offset 2000_{16} in A24/A32 space. This window gives the host PC direct access to any PCI device residing on the on-board PCI bus as well as any secondary buses that may exist. To point the 8 Kilobyte window to the correct PCI bus address the host must control the direct access control register defined by the shared memory device. The register is part of the shared memory operation registers accessible by the VXI host in A24/A32 space. Refer to section 6.8 for details on using the direct access capabilities of the carrier.

4.4 SHARED RESOURCES AND DEVICE COMMUNICATIONS

Communication between the host-side application and the device-side application is accomplished using a couple of resources available to both the host and the device. Namely, these shared resources are the operations registers and the shared memory device as shown in Figure 18. These shared resources are used to perform VXI communications, device configurations, block data transfers, and other miscellaneous functions.

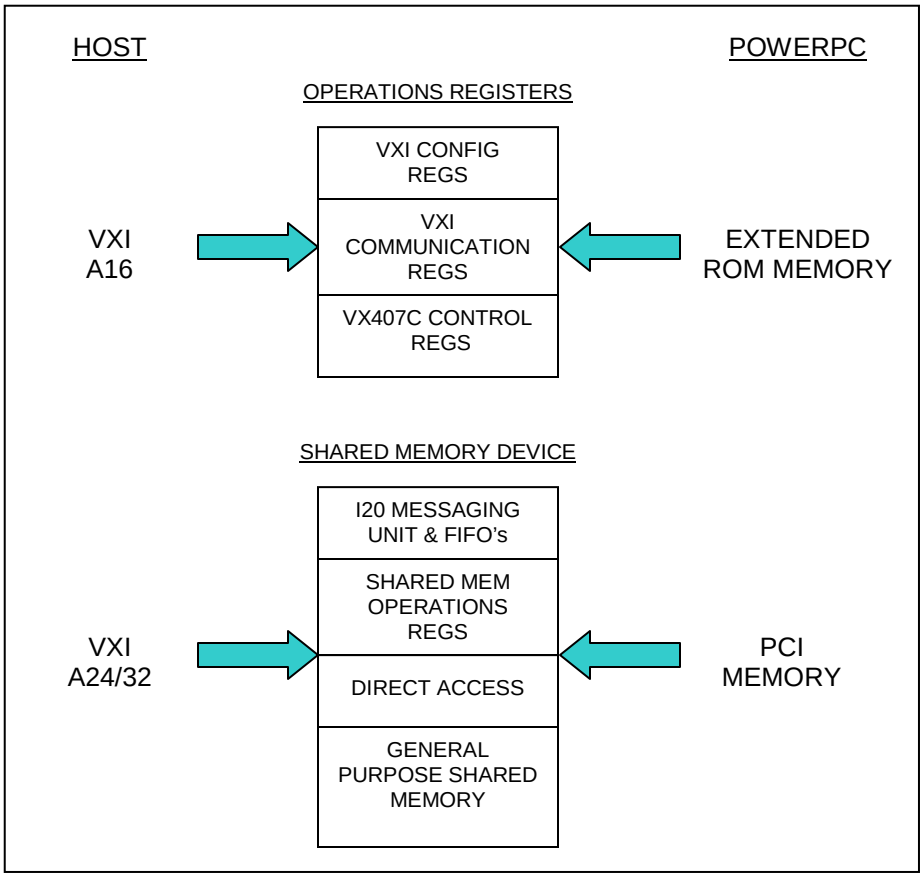


Figure 18. Shared Resources

4.4.1 Operations Registers

The operations registers combine the required VXI configuration registers, VXI communication registers and a set of carrier control registers. Table IV lists all available registers along with their offset. The VXI host can access each registers at its specified offset in the A16 address space. The PowerPC can access each register at its specified offset in its extended ROM space starting at address 7000_0000_{16}. There maybe access restrictions on individual registers or individual bits within a register depending on the whether it is being accessed by the host-side or the device-side application. Refer to the register descriptions in Figure 19, Figure 20, and Figure 21 for details on each register.

The PowerPC's interface to the operations registers is only 8-bits wide. In Figure 19, Figure 20, and Figure 21, the least significant bits reside in the low PowerPC address. For example, VXI ID bits 0-7 reside at PowerPC address $0x7000\_0000_{16}$ and bits 8-15 reside at address $0x7000\_0001_{16}$.

Table IV. Operations Registers Map

Offset	Register Description
VXI Configuration Registers	
00	VXI ID
02	VXI Device Type
04	VXI Status/Control
06	VXI Offset Register
VXI Communication Registers	
08	VXI Protocol
0A	VXI Response
0C	Reserved
0E	VXI Data Low
10 – 1F	Reserved
VX407C Control Registers	
20	VX407C Status/Control
22	Interrupt Control
24	Trigger Control
26-3F	Reserved

4.4.1.1 VXI Configuration Registers

The VXI configuration registers contain basic information needed to configure a VXI system as required by the VXIbus specification. The configuration information includes: manufacturer identification, product model code, device type, memory requirements, device status, and device control. The registers are briefly described below and are detailed in Figure 19.

VXI Identification (ID) Register (00_{16}): This register provides the manufacturer identification, device classification (i.e., register based or message based), and the addressing mode (i.e. A32 or A24). It is a read only register from the VXI host. The PowerPC can write this register however it should be done immediately after reset prior to running VXI resource manager. System configuration options are available to initialize this register automatically after reset. Refer to section 5.2.2 for details on the configuration options.

VXI Device Type Register (02_{16}): This register provides the model code identifier and required memory information. It is a read only register from the VXI host. The PowerPC can write this register however it should be done immediately after reset prior to running VXI resource manager. System configuration options are available to initialize this register automatically after reset. Refer to section 5.2.2 for details on the configuration options.

VXI Status/Control Register (04_{16}): A read of this register provides the state of the VXI MODID* line and the pass, ready, and self-test status bits. A write to this register allows

disabling of the SYSFAIL function and performing a reset of the carrier. This register is readable and writeable from both the VXI host and the PowerPC however, there are several access restrictions on individual bits depending on the source of the access.

VXI Offset Register (06_{16}): This register controls the offset value for addressing the A24/A32 address space. The VXI system resource manager or control module sets this value according to the memory requirements specified for this module and the memory requirements of the other instruments in the system. This register is readable and writeable from the VXI host. The PowerPC only has read capability of this register.

VXI ID

00_{16}

Bit	15 – 14	13 – 12	11 – 0
PPC Write	Device Class	(read only)	Manufacturer ID
VXI Write	(read only)		
Read	Device Class	Address Space	Manufacturer ID

Device Class ⇨ Device Class (10 = Message Based, 11 = Register Based, 00 & 01 = reserved)
Address Space ⇨ Address Space (00 = A16/A24, 01 = A16/A32, 10 = reserved, 11 = A16 Only))
Manufacturer ID ⇨ Manufacturer Identification (default = $FC1_{16}$)

VXI Device Type

02_{16}

Bit	15 – 12	11 – 0
PPC Write	(read only)	Model Code
VXI Write	(read only)	
Read	Required Memory	Model Code

Required Memory ⇨ 32 Kbytes required (F_{16} if A32, 8_{16} if A24)
Model Code ⇨ Model Code (default = $FE4_{16}$)

Figure 19. VXI Configuration Registers

VXI Status/Control

04_{16}

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPC Write	-								POST Result				RDY	PASS	-	
VXI Write	AAA	-													SI	RST
Read	AAA	MID	-						POST Result				RDY	PASS	-	

AAA ⇨ A24/A32 Access (0 = disabled)
MID ⇨ Module ID Status(0=MODID* line is asserted)
POST Result ⇨ Power On Self Test Result[1]

0000	Passed
0001	SDRAM Failure
0010	Shared Memory Failure
0011	Flash Memory Failure
0100	Operations Register Failure
0101	Trigger Matrix Failure
0110-1111	Reserved

RDY ⇨ Ready(1=ready)
PASS ⇨ Pass/Fail Indicator(0=executing or failed, 1=passed)
SI ⇨ Sysfail Inhibit(1=inhibit)
RST ⇨ Reset(writing a '1' to this bit resets the carrier; after a minimum of 100μs a '0' must be written to resume normal operation)

Notes:
1. Refer to section 5.2.3.3 for details on the Power On Self Test (POST).

VXI Offset

06_{16}

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPC Write	(read only)															
VXI Write	A24/A32 Offset															
Read	A24/A32 Offset															

A24/A32 Offset ⇨ Offset to the carriers A24/A32 memory space

Figure 19. VXI Configuration Registers (continued)

4.4.1.2 VXI Communication Registers

The VXI communication registers are defined by the VXI specification for message based devices. They provide all the functionality necessary to perform the VXI word serial protocol. The word serial protocol firmware in the on-board system utilities manages these registers to perform message passing. Only on very rare occasions should the user application need to access these registers directly. Figure 20 shows these registers in detail.

CAUTION: It is rarely necessary for either the host-side or the device-side application to access the VXI communications registers directly. The host-side VXI libraries and the device-side on-board system utilities automatically manage these registers when performing message passing functions. Directly accessing these registers is not advised without prior knowledge of the VXI specification for message based devices.

VXI Protocol Register (08_{16}): This register indicates which message based protocols the carrier supports and indicates additional communication capabilities of the carrier. This register is a read only register by both the host and device applications.

VXI Response Register ($0A_{16}$): This register indicates status of the carrier's communications capabilities. The register is read only from the host-side application. The PowerPC writes to this register to perform message passing. The PowerPC user application should never need to write to this register. The word serial protocol handler in the on-board system utilities automatically manages this register.

VXI Data Low Register ($0E_{16}$): This register is used to pass VXI commands and data to and from the carrier over the VXI bus. A write to this register causes the word serial protocol handler in the on-board system utilities to execute and handle the command. The status of this register is indicated in the VXI response register. This register is readable and writeable by both the host and device applications. However, the rules for message based communications as set by the VXI specification must be followed to ensure data integrity. Neither the PowerPC nor the VXI host applications should need to write directly to this register. The word serial protocol handler in the on-board system utilities automatically manages this register and the VXI host should use standard VXI libraries to communicate with the carrier.

VXI Protocol

08_{16}

Bit	15	14	13	12	11	10	9	0
PPC Write	(read only)							
VXI Write	(read only)							
Read	CMDR*	SIG*	MSTR*	INT	FHS*	SMEM*	reserved	

CMDR* ⇨ Commander (default 1=Servant only capabilities)
SIG* ⇨ Signal Register (default 1=No signal register)
MSTR* ⇨ Master (default 1=No VME bus master capabilities)
INT ⇨ Interrupter (default 1=Has interrupter capabilities)
FHS* ⇨ Fast Handshake (default 1=Does not support the Fast Handshake Mode)
SMEM* ⇨ Shared Memory (default 1=Does not support the shared memory protocol)

VXI Response

$0A_{16}$

Bit	15	14	13	12	11	10	9	8	7	6	0
PPC Write	0	rsvd	DOR	DIR	ERR*	RRDY	WRDY	FHS*	LCK*	reserved	
VXI Write	(read only)										
Read	0	rsvd	DOR	DIR	ERR*	RRDY	WRDY	FHS*	LCK*	reserved	

DOR ⇨ Data Out Ready (1 = message byte available to be read by VXI host)
DIR ⇨ Data In Ready (1 = carrier ready to receive message byte)
ERR* ⇨ Error (0 = error occurred, 1 = no error)
RRDY ⇨ Read Ready (1 = data has been place in the data low register for read by the VXI host)[1]
WRDY ⇨ Write Ready (1 = data low register is empty and ready for VXI host to write a command)[2]
FHS* ⇨ Fast Handshake Active (0 = Fast handshake mode is active)
LCK* ⇨ Locked (0 = a commander has locked the carrier from being accessed by other sources)

Notes:
1. The read ready bit automatically cleared by the VXI interface logic when the data low register is read by the VXI host.
2. The write ready bit is automatically cleared by the VXI interface logic when the data low register is written by the VXI host.

Figure 20. VXI Communications Registers

VXI Data Low

$0E_{16}$ Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPC Write	Data Low															
VXI Write	Data Low															
Read	Data Low															

Data Low ⇨ Low Data Word

Notes:
1. The VXI specification for message based devices must be followed for reading and writing the data low register
2. The read ready bit in the VXI response register is automatically cleared by the VXI interface logic when the data low register is read by the VXI host.
3. The write ready bit in the VXI response register is automatically cleared by the VXI interface logic when the data low register is written by the VXI host.

Figure 20. VXI Communications Registers (continued)

4.4.1.3 VX407C Control Registers

The VX407C control registers provide miscellaneous configuration, status, and control functionality for the carrier. Refer to the register descriptions and Figure 21 for details.

VX407C Control/Status (20_{16}): This register provides miscellaneous status information and control functionality for the carrier. Each bit has individual read/write restrictions depending on whether the host-side or the device-side application is accessing it.

Interrupt Control (22_{16}): This register is used by the PowerPC application to interrupt the VXI host. The host or PowerPC application can enable/disable the ability of the carrier to interrupt the VXI host and can configure the interrupt level. The PowerPC application can set a vector that'll be passed to the VXI host and generate the interrupt.

Trigger Control (22_{16}): This register is used to configure the trigger matrices to map PXI trigger lines to VXI trigger lines. The same address location is used to configure both matrices and each trigger line within each matrix. The trigger line and matrix to be configured is specified by setting the trigger select bits in the VX407C Control/Status register. This register is readable and writeable by both the VXI host and the PowerPC. Refer to section 4.2.6 for details on the trigger architecture.

VX407C Status/Control

20_{16} Bit	15	11	8	7	6	5	4	3	2	1	0
PPC Write	reserved	TRIGSEL		-	VXDIS	PMC RST	PXI1 RST	PXI0 RST	-	-	-
VXI Write	reserved	TRIGSEL		-	VXDIS	PMC RST	PXI1 RST	PXI0 RST	-	-	-
Read	reserved	TRIGSEL		-	VXDIS	PMC RST	PXI1 RST	PXI0 RST	CFG1	LM	USF

USF[1] ⇨ Update System Firmware(Value of the Update System Firmware configuration switch)
0 = Boot normally
1 = Boot into update system firmware mode

LM[2] ⇨ Launch Mode (Value of the Launch User Application configuration switch)
0 = Run system process loop
1 = Launch user application

CFG1[3] ⇨ User Configuration Switch (Value of user configuration switch)
0 = Switch closed
1 = Switch open

PXI0 RST ⇨ PXI Slot 0 Reset (writing a '1' to this bit resets the module in PXI slot 0; after a minimum of 100µs a '0' must be written to resume normal operation)

PXI1 RST ⇨ PXI Slot 1 Reset (writing a '1' to this bit resets the module in PXI slot 1; after a minimum of 100µs a '0' must be written to resume normal operation)

PMC RST ⇨ PMC Reset (writing a '1' to this bit resets the module in PMC slot; after a minimum of 100µs a '0' must be written to resume normal operation)

VXDIS[4] ⇨ VXI Disable (writing a '1' to this bit will disable the interrupt generated whenever a message byte is received over the VXI interface)

TRIG SEL ⇨ Trigger and matrix that is accessed whenever the Trigger Control register is read or written

	Input Matrix		Output Matrix
0000	PXI Trig 0	1000	VXI Trig 0
0001	PXI Trig 1	1001	VXI Trig 1
0010	PXI Trig 2	1010	VXI Trig 2
0011	PXI Trig 3	1011	VXI Trig 3
0100	PXI Trig 4	1100	VXI Trig 4
0101	PXI Trig 5	1101	VXI Trig 5
0110	PXI Trig 6	1110	VXI Trig 6
0111	PXI Trig 7	1111	VXI Trig 7

Notes:
1. This bit is used by the boot code to determine whether to go into firmware update mode immediately on power-up or whether to boot normally
2. This bit is used during system reset to determine whether a user application is launched or not. Refer to section 5.2 for details on the effects of the launch user application switch
3. The user configuration switch has no effect on the carrier operation. The software application can use this value in any way it desires.
4. Disabling the VXI interrupt will cause the VXI message based interface to not operate and may not be recoverable. Make sure the DIR and DOR bits in the VXI Response register are cleared before setting this bit to avoid problems.

Figure 21. VX407C Control Registers

Interrupt Control

22_{16}

Bit	15 – 8	7	6	5	4	3 – 1	0
PPC Write	Vector	PIP	-	PIE	SMIE	VXI Level	MIE
VXI Write	(read only)		-	PIE	SMIE	VXI Level	MIE
Read	Vector	PIP	SMIP	PIE	SMIE	VXI Level	MIE

Vector[1] ⇨ Upper 8 bits of the status-id value returned during an interrupt acknowledge cycle

$00_{16} – 7F_{16}$	VXI response interrupt
80_{16}	reserved for shared memory interrupt
81_{16}-BF_{16}	user defined interrupt
FC_{16}	VXI request false event interrupt
FD_{16}	VXI request true interrupt
FE_{16}	reserved
FF_{16}	no cause given

PIP ⇨ Processor interrupt pending (if PIE=1 then writing a 1 to this bit will generate an interrupt)

SMIP[2] ⇨ Shared memory interrupt pending (a value of 1 indicates that the shared memory device has asserted its interrupt line)

PIE ⇨ Processor interrupt enable (1 = a value of 1 in the PIP bit will generate an interrupt)

SMIE ⇨ Shared memory interrupt enable (1 = enable interrupts from the shared memory device)

VXI Level ⇨ VXI Interrupt Level (0= disabled, 1-7=IRQ 1-7)

MIE[3] ⇨ Master Interrupt Enable (if 1 then writing a 1 to the IP bit will generate an interrupt)

Notes:

1. The vector value specifies the type of interrupt that is pending. The 'user defined interrupt' vector range may be used by the user application when generating processor interrupts. All other interrupt vector values are defined by the VXI specification or reserved by the VX407C. The system utilities manage this register field when generating VXI defined interrupts.
2. If both the PIP and SMIP bits are set, the shared memory interrupt will have priority and a vector value of 80_{16} will be returned regardless of the value in the vector field.
3. All interrupts are Release On Acknowledge (ROAK) interrupts. This is achieved by clearing the MIE bit during the interrupt acknowledge cycle. This bit should be re-enabled prior to returning from the interrupt service routine in order for interrupts to continue to occur.

Figure 21. VX407C Control Registers (continued)

Trigger Control

24_{16}

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPC Write	-	-	-	-	-	-	-	-	-	-	EN	INV	-	MAP		
VXI Write	-	-	-	-	-	-	-	-	-	-	EN	INV	-	MAP		
Read	-	-	-	-	-	-	-	-	-	-	EN	INV	-	MAP		

MAP ⇨ Trigger that the selected output trigger is mapped to. The selection depends on the value of TRIGSEL in the VX407C Control/Status Register. If the Input matrix is selected, then this value selects the VXI trigger to map to the PXI output trigger selected. If the output matrix is selected, then this value selects the PXI trigger to map to the VXI output trigger selected.

000	VXI/PXI Trig 0	100	VXI/PXI Trig 4
001	VXI/PXI Trig 1	101	VXI/PXI Trig 5
010	VXI/PXI Trig 2	110	VXI/PXI Trig 6
011	VXI/PXI Trig 3	111	VXI/PXI Trig 7

INV ⇨ Invert the trigger (1 = inverted, 0 = non-inverted (default))
EN ⇨ Enable the trigger (1 = enabled, 0 = disabled (default))

Note: Refer to section 4.2.6 for details on the trigger architecture.

Figure 21. VX407C Control Registers (continued)

4.4.2 VXI Word Serial Protocol

Most communications between the host and the PowerPC is via the VXI word serial protocol defined by the VXI bus specification. Message passing is implemented via the VXI communication registers as described in section 4.4.1.2. The PowerPC handles the device-side of the communication protocol with the on-board system utilities.

All standard word serial commands are implemented in the on-board system utilities. System commands are defined to provide general access to the PXI/cPCI modules, PowerPC utilities, and carrier configuration options. Application dependant commands can be developed per application to communicate with the PXI/cPCI modules and PowerPC application at a higher level.

4.4.3 General Purpose Shared Memory

The 16 kilobytes of general purpose shared memory can be used to pass large amounts of data between the host application and the PowerPC application. High performance, data intensive applications can take advantage of burst access to this memory space from the host and DMA access to this memory space from the PowerPC or other device on the PCI bus. The shared memory device will provide low level arbitration for this memory space. High level handshaking can be provided through message based commands and/or VXI interrupts.

4.4.3.1 Shared Memory Arbitration

The shared memory device provides arbitration logic so that any location can be accessed at the same time by both the VXI host via the shared memory device's local bus interface and the PowerPC via the shared memory device's PCI bus interface.

The shared memory device also provides an Arbitration Utility Flag Register accessible to the host through VXI A24/A32 space and to the PowerPC through PCI memory space. Software can use this register to implement high level memory arbitration. As shown in Figure 22, the register provides four arbitration flags that can be owned by either the local bus (VXI) or the PCI bus (PowerPC) but never both at the same time.

Arbitration Utility Flag Register

$04C0_{16}$

Bit	31 … 26	25	24	23 … 18	17	16	15 … 10	9	8	7 … 2	1	0
Write	-	L3	P3	-	L2	P2	-	L1	P1	-	L0	P0
Read	-	L3	P3	-	L2	P2	-	L1	P1	-	L0	P0

Lx ⇨ Local bus ownership (This bit can only be set by the VXI host and only if the corresponding Px bit is not set)

Px ⇨ PCI bus ownership (This bit can only be set by the PowerPC and only if the corresponding Lx bit is not set)

Figure 22. Shared Memory Arbitration Utility Flag Register

4.4.3.2 DMA/Burst

The PowerPC contains an embedded DMA controller than can be used to burst data into and out of shared memory. The destination or source of the DMA transfer can be local memory or another PCI device. Details on using the embedded DMA controller are beyond the scope of this document. Refer to the MPC8245 User's Manual for further information.

The shared memory device also contains an embedded DMA controller that can burst between the shared memory device and any PCI device. The shared memory can be programmed to perform the DMA transfer then interrupt the VXI host when the transfer is complete. The shared memory's DMA controller is fully accessible without the help of a PowerPC application.

The VXI host can burst data into and out of shared memory using VXI block transfer cycles. If supported, the host's VXI library should provide functions to perform block transfers. The data width of block transfers can be 16 or 32 bits however, the data bus width must be configured as discussed in section 4.3.2.

4.4.4 I_20 Message Unit

The shared memory device has an on-board I_2O messaging unit that can be used to communicate between the host and the PowerPC. However, the I_2O messaging unit will only be used in special circumstances since the VXI Word Serial Protocol provides full message passing capabilities. Full access to the I_2O messaging unit is provided through VXI A24/A32 space and through PCI memory space. Refer to the Cypress CY7C09449PV Data Sheet for further details.

4.4.5 General Purpose FIFOs

The I_2O messaging unit contains four FIFOs that are available for general purpose use when the I_2O messaging unit is not being used. Each FIFO is 32 deep x 32 bits and can be accessed by both the host and the PowerPC applications. Access to the FIFOs is achieved through 4 registers that are part of the I_2O Specific Registers section mapped to VXI A24/A32 space and to PCI memory space. Figure 23 describes the FIFO registers. Each register is a shared port such that on a write it places data on one FIFO and on a read it reads data from a different FIFO. This way each FIFO can be accessed by both the host and the PowerPC simultaneously.

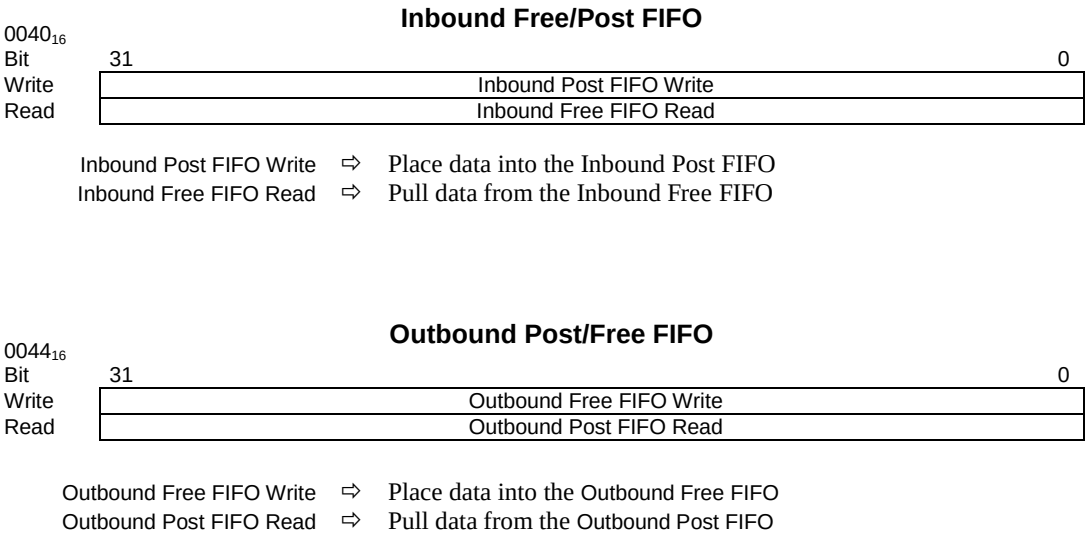


Figure 23. General Purpose FIFO Registers

Inbound Post/Free FIFO

0048_{16}

Bit	31	0
Write	Inbound Free FIFO Write	
Read	Inbound Post FIFO Read	

Inbound Free FIFO Write ⇨ Place data into the Inbound Free FIFO
Inbound Post FIFO Read ⇨ Pull data from the Inbound Post FIFO

Outbound Free/Post FIFO

$004C_{16}$

Bit	31	0
Write	Outbound Post FIFO Write	
Read	Outbound Free FIFO Read	

Outbound Post FIFO Write ⇨ Place data into the Outbound Post FIFO
Outbound Free FIFO Read ⇨ Pull data from the Outbound Free FIFO

Notes:
1. Writing to a full FIFO will result in loss of data. The contents of the FIFO will not change.
2. Reading from an empty FIFO will return $FFFFFFFF_{16}$.
3. All FIFOs are empty at reset.

Figure 23. General Purpose FIFO Registers (continued)

5.0 SOFTWARE ARCHITECTURE

For a typical application, the system software will consist of both an on-board application running on the PowerPC and a host application running on the VXI host computer. The two applications will communicate over the VXI bus using the shared resources of the VX407C described in section 4.4. Figure 24 illustrates the system architecture for a typical application.

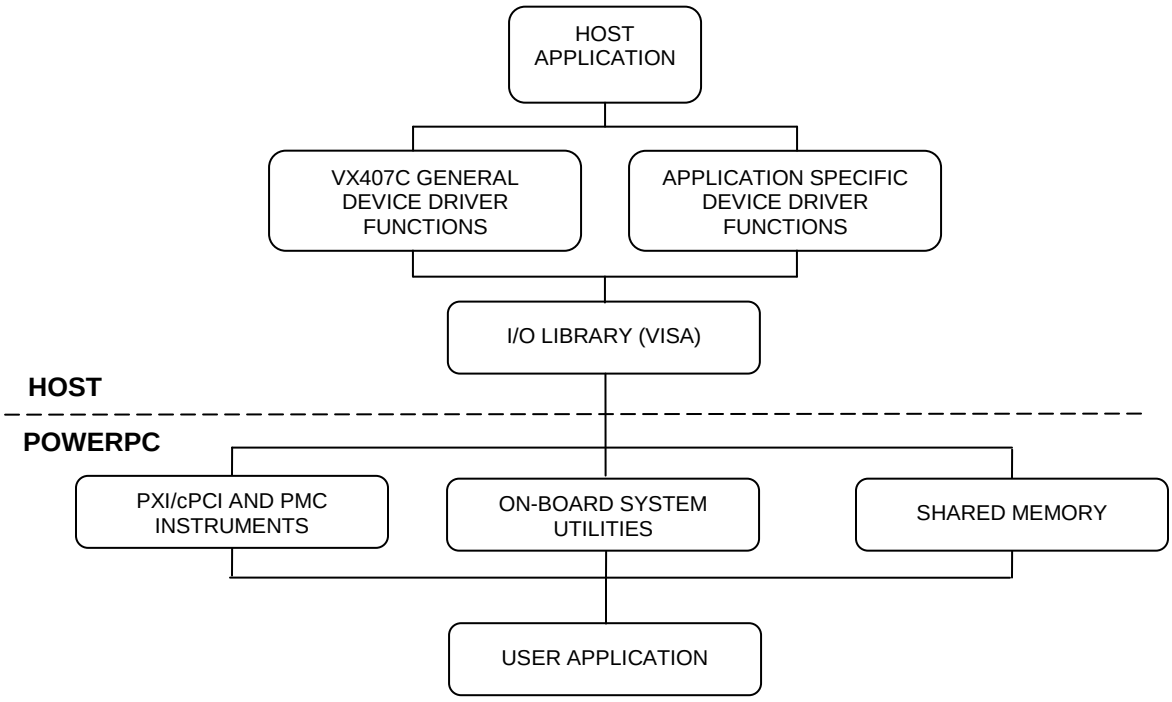


Figure 24. System Software Architecture

5.1 HOST SYSTEM SOFTWARE

The host-side application will normally run on a standard PC or an embedded VXI controller and will communicate with the device using standard off-the-shelf VXI interfaces and software libraries. The application can be part of a large automated test system responsible for controlling the VX407C along with numerous other instruments, or it can be an independent diagnostic application allowing the user to interact with the VX407C only. The typical application will be responsible for sending high level

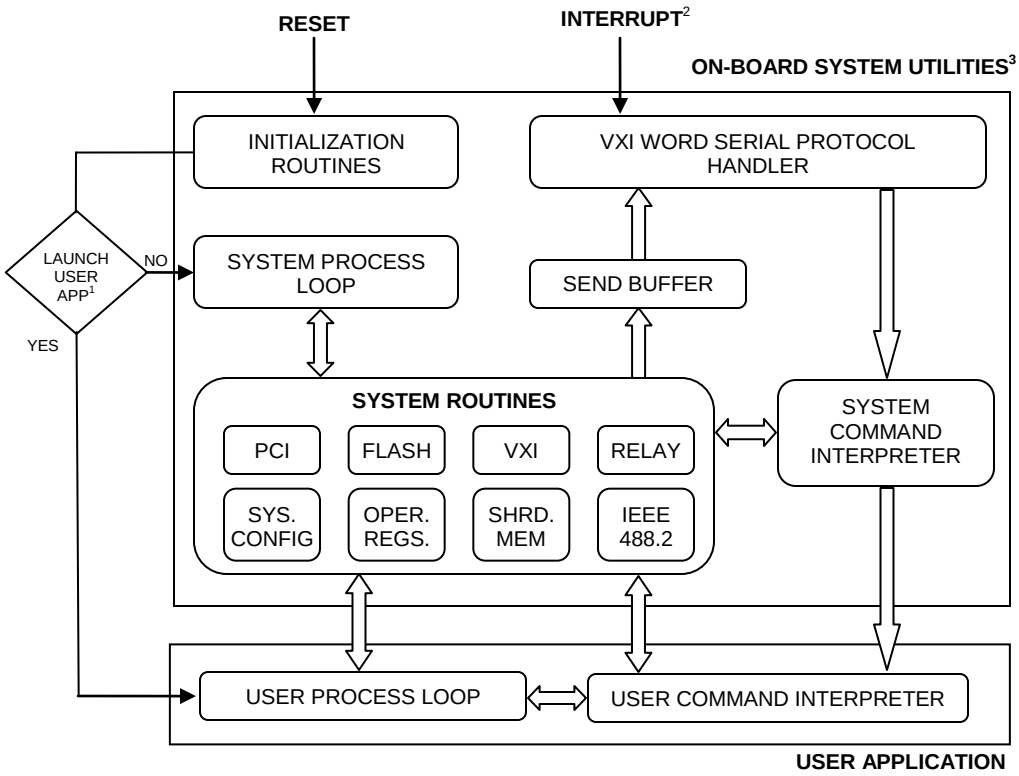
commands to the PowerPC application and retrieving, analyzing, and reporting data. It might also provide a soft front panel that a user may use to interact with the instrument.

The device driver library will provide high level functions to communicate with the instrument. The library can be separated into VX407C general carrier driver functions and application specific driver functions. The general carrier functions will provide functionality that exists regardless of the PXI/cPCI modules and PowerPC application residing on the board. The application specific functions will be developed along with the PowerPC software and will be specific to the given application.

This document does not attempt to define the scope or the architecture of the host application. It is only mentioned here to illustrate how the application would interact with the software running on-board the VX407C's PowerPC system.

5.2 ON-BOARD SYSTEM UTILITIES

On-board system utilities are provided to initialize the VX407C, assist the user application in using the various on-board interfaces, and to perform VXI word serial message passing. The utilities contain initialization routines, a system process loop, hardware interface and other system routines, a VXI word serial protocol handler, a system command interpreter, and a user application interface. Figure 25 illustrates the architecture of the on-board system utilities and how they interface to the user application.



Notes:

1. Launch User Application selection is determined by module configuration hardware switch. See section 3.4.2 for details.
2. VXI Word Serial Interrupt is automatically generated for every message byte written to the VX407C over the VXI bus.
3. The On-board System Utilities physically reside in the non-volatile Boot ROM and the first sector of flash

Figure 25. On-board System Utilities Software Architecture

On reset, the firmware will run a short self-test, initialize the hardware, initialize the word serial protocol handler, and start the user application if configured to do so. The setting of the module configuration switch determines if a user application is to be launched. If a user application is not to be used then the initialization code will launch a simple on-board process loop allowing the VX407C to respond to system commands and operate normally.

The VXI word serial protocol handler is completely autonomous so that the user application can concentrate on performing its tasks and not on instrument communication. When a command is written to the VXI message based registers, an interrupt is generated and the word serial protocol handler is launched to manage the data transfer and retrieve the command. Messages are immediately passed to the user command interpreter if one is installed. The user command interpreter is part of the user application and should be installed when the application is launched. If a user command interpreter is not installed or returns a “command not supported” error, the system utilities will assume the command is a system command and process it accordingly.

When the user application needs to send data back to the VXI host, it simply calls the appropriate system routine to place data in the send buffer to be processed by the word serial protocol handler.

The user application calls the on-board system routines by generating an exception with a *System Call* (**sc**) instruction. The PowerPC's general purpose registers are used to specify which routine to run, to pass parameters to the routine, and to receive returned data and status from the routine. The user application must configure these registers prior to performing the system call instruction.

The VX407C hardware architecture will also support several commercial off-the-shelf real-time operating systems. The software effort required to launch one of these operating system is OS dependant and beyond the scope of this document.

5.2.1 System Resource Usage

The on-board system utilities require some system resources for operation. The resources include: boot ROM, RAM, flash memory, and interrupts. These sections are used for storing code both during execution and when power is off, for storing and manipulating variables during execution, and for storing configuration options. Figure 26 shows the sections of memory reserved for use by the system utilities.

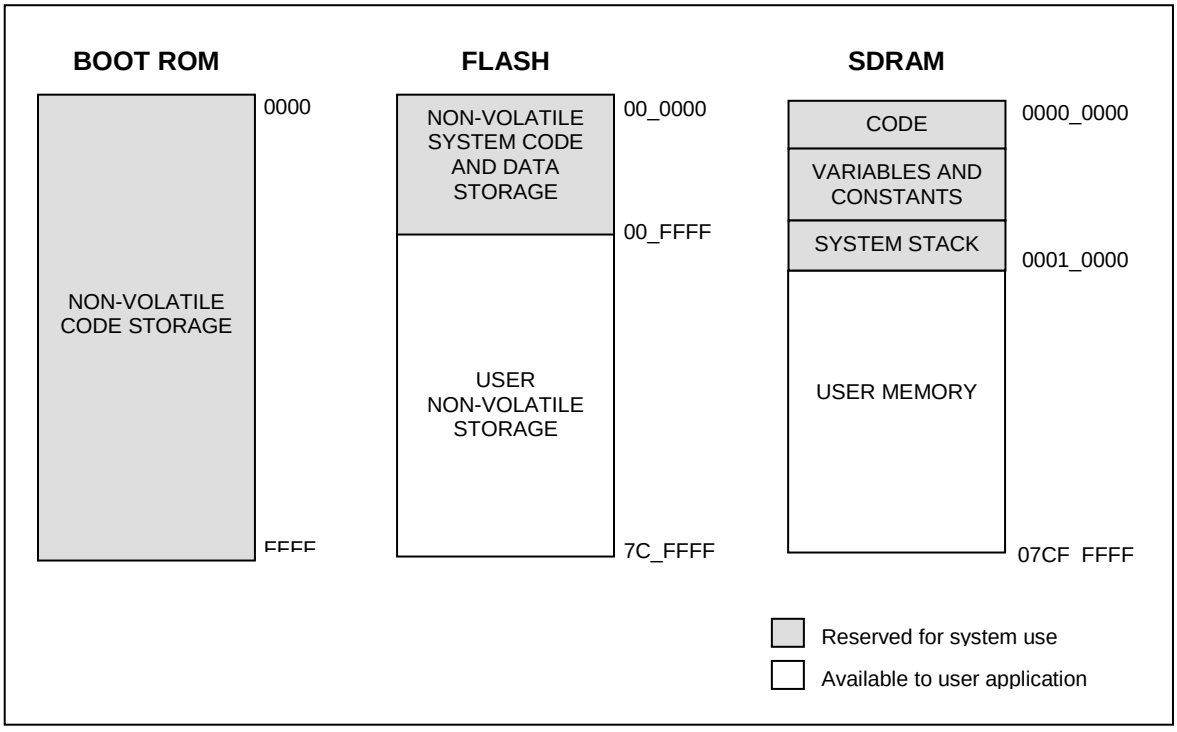


Figure 26. System Resource Usage

WARNING: The user application must not modify the sections of memory reserved for the system utilities. Doing so may result in a system failure requiring a reboot of the system.

The entire 32 kilobytes of boot ROM is reserved for non-volatile storage of processor initialization and boot code. This code is responsible for initializing the PowerPC to a state where a minimal application can run. It then attempts to launch the on-board system utilities. Also stored in the boot ROM is code to download and update the on-board system utilities via the VXI bus. The boot ROM is not in-circuit programmable; therefore, it is not possible for the user application to overwrite this space.

The first 64 kilobytes of flash are reserved for non-volatile storage of the configuration options and for storage of the on-board system utilities code and data. All code and data necessary for the system utilities to operate are programmed into this memory during the manufacturing process of the VX407C. The section occupies the entire first sector of flash since, in most cases, modifying the configuration options requires erasing the entire flash sector in which they reside. Therefore, this space can be modified without affecting any user storage and the user application can modify or erase its storage space without affecting the configuration options or the system utilities. If this section becomes corrupt or erased, the user can easily restore the system to its default configuration by updating the system firmware using the routines that reside in the boot ROM.

The first megabyte of system RAM is reserved for the system utilities execution space. During execution, certain routines of the on-board system utilities are stored in RAM for faster execution. The space is also used for storage of variables and constants and for the system stack. The user application must be written such that it does not overwrite this section of RAM. Doing so may result in a system failure requiring a reboot of the system.

The system requires the use of the system management interrupt (SMI) for proper operation of the VXI word serial protocol handler. This requires that PowerPC interrupts be enabled at all times. If the user disables PowerPC interrupts by setting the external interrupt enable bit in the processors machine state register (MSR[EE]) to 0, the instrument will not respond to VXI word serial commands.

5.2.2 Configuration Options

Configuration options allow the user to configure certain behaviors of the system initialization sequence. Options are configured using VXI system commands. The settings are stored in non-volatile flash memory, so they remain valid even after power is removed from the instrument. Factory default values, for each option, are stored in boot ROM and can easily be restored using a VXI system command. Table V shows all available configuration options.

Table V. Configuration Options

Option	**Settings**	**Description**
Boot Address	Any addressable location	Set to location of user application
Boot Type	0 = normal (default) 1 = download	Normal = use boot address, download = download code from VXI prior to boot
VXI A24/A32 Width	2=16-bit (default) 4=32-bit	Initializes the data bus width for the A24/A32 address space
VXI Manf ID	Any 12-bit number default = $FC1_{16}$ (C&H)	Sets the VXI manufacturer ID value in the VXI ID Register
VXI Model Code	Any 12-bit number default = $FE4_{16}$ (VX407C)	Sets the VXI model code value in the VXI Device Type register
VXI Type	0 = message (default) 1 = register	VXI interface type (message or register based)

Note: Flash programming must be enabled to modify the configuration options. Refer to section 3.4.2 for details on enabling flash programming.

Boot Address: The system initialization routine can launch a user application from any PowerPC addressable memory location. The boot address configuration option allows the user to specify the location of the user application. This option is only valid when the module is configured to launch a user application using the module configuration switch discussed in section 3.4.2. Also this option is ignored if the boot type configuration option is set to download.

Boot Type: The boot type configuration option allows the user to specify whether the application is to be downloaded after reset or whether it resides in non-volatile memory on the carrier. This option is only valid when the module is configured to launch a user application using the module configuration switch discussed in section 3.4.2. If the boot type is set to normal, the initialization routine will attempt to launch the application from the address specified in the boot address configuration option. It the boot type is set to download, the initialization routine will wait for the application to be downloaded over the VXI bus via shared memory. Instructions on downloading an application are provided in section 6.4.

VXI A24/A32 Width: The VXI A24/A32 width configuration option initializes the VXI data bus access width to 16 or 32 bits as described in section 4.3.2. This option only configures the initial bus width after a reset. The bus width can be changed at any time during operation using the VXI system commands.

VXI Manufacturer ID: The VXI manufacturer ID configuration option initializes the 12-bit manufacturer ID value in the VXI ID register. The default value is C&H Technologies' assigned VXI ID ($FC1_{16}$). This number can be set to any 12-bit value.

VXI Model Code: The VXI model code configuration option initializes the 12-bit model code value in the VXI Device Type register. The default value is the VX407C model code assigned by C&H Technologies ($FE4_{16}$). This number can be set to any 12-bit value.

VXI Type: The VXI type configuration option will set the carrier hardware to operate either as a message based device or as a register based device. If message based is selected, the VXI word serial protocol handler will be enabled and the carrier will be initialized to accept VXI commands. If register based is selected, then the VXI word serial protocol handler will be disabled and only VXI register based accesses can be performed. This option is only valid when the module is configured to launch a user application using the module configuration switch discussed in section 3.4.2. ***When a user application is not being used, the VX407C will always initialize as a message based device and the word serial interface will be enabled.***

5.2.3 Initialization Routines

At system reset, including a power on reset, the PowerPC generates a reset exception and jumps to the first instruction in the boot ROM to begin executing the initialization routines. The boot sequence includes initializing the PowerPC configuration registers, enumerating the PCI bus, running a short self test, initializing the operations registers, initializing the shared memory device, and launching the application. Configuration options can be set using defined VXI commands. Refer to section 5.2.2 for details on configuration options.

5.2.3.1 PowerPC Initialization

The first action of the VX407C boot sequence is an initialization of the PowerPC and all its associated peripheral interfaces. The MPC8245 is a highly integrated processor with the capability of interfacing to many different peripherals. An extensive set of configuration registers are provided to configure the processor for all the different possible architectures. During the boot sequence, these registers are initialized to values appropriate for the VX407C hardware architecture. Detailed settings for each configuration option are beyond the scope of this document. However, a list of configuration registers and the value that each is initialized to, is provided in APPENDIX B. For further details on the processor configuration, refer to the MPC8245 User's Manual.

5.2.3.2 PCIbus Enumeration

After successfully initializing the PowerPC, the firmware will scan the PCI bus, determine the resources necessary for all the devices on the bus, and allocate the PowerPC resources accordingly. This process is referred to as enumeration. The routine will cycle through each of the PCI interfaces on the PowerPC's PCI bus and use each device's configuration space to determine and allocate necessary resources. If a PCI to PCI bridge is found on the bus, the routine will scan its secondary bus for devices and allocate any resources necessary. The PCI enumeration routine will continue this process, until necessary resources have been allocated for all devices in the system.

It can be assumed that as long as no devices are added or removed from the system's PCI bus, the same resources will be allocated to a specific device every time the carrier is reset. For example, if no devices are added or removed from the on-board PCI bus then the base address of a device in the PMC position will not change. However, this cannot be guaranteed if the devices on PCI bus are changed in any way. Also it cannot be guaranteed from one version of the on-board system utilities to another. For this reason, ***it is recommended that application software be written such that it verifies each device's allocated resources (including shared memory) prior to accessing the device.***

5.2.3.3 Power-On-Self-Test (POST)

Immediately after initializing the PowerPC, the boot sequence runs the Power On Self Test (POST). The POST verifies basic operation of the carrier and all of its components. The routine includes a test of: local SDRAM, flash memory, shared memory, operations registers and basic processor functions.

The results of the self test are recorded in the VXI Control/Status register at offset 04_{16} of VXI A16 space. If the test passed, the Pass bit in the VXI Control/Status register will be set to a '1' and the POST result value will be set to '0'. If the test failed, the Pass bit will be set to '0' and the POST result value will be set to a binary number indicating the failure. Refer to the VXI Control/Status register description in Figure 19 for details on the POST result value. Failure of the POST will also cause the VXI *SysFail* signal to be asserted notifying the VXI host of the failure. The SysFail inhibit bit (SI) in the VXI Control/Status register can be used to inhibit the *SysFail* signal to perform diagnostic functions on the carrier.

5.2.3.4 Launching the Application

Once the boot sequence has completed initialization, the firmware launches either the system process loop that is part of the system utilities or a user application residing anywhere in the processors addressable space. The firmware determines which to launch by reading the Launch Mode (LM) bit in the PowerPC Control Status Register at offset 20_{16} of the carrier's operations registers space. The value of the LM bit is determined by

the module configuration switch. If the launch user application switch is in the OFF position, the LM bit will be set to 1 and the firmware will launch the user application. If it is in the ON position, the LM bit will be set to 0 and the firmware will launch the system process loop. Refer to section 3.4.2 for details on the module configuration switch.

The user application is responsible for loading itself into RAM, setting up the PowerPC's general purpose registers for use, initializing its stack and heap, and allocating memory. For example, if the application is a C compiled executable then the general purpose registers must be initialized according to the Embedded Application Binary Interface (EABI) specification. Most off-the-shelf development tools contain libraries that can be linked with the application to perform these tasks automatically.

The user can configure the initialization software to boot the user application from any address within the PowerPC's addressable space. This allows the application to be stored anywhere in on-board flash or on a storage device residing on the PCI bus such as a PMC flash memory device or on a removable drive connected to a PMC or CompactPCI disk controller. Alternatively, the initialization routine can be configured to download the application over the VXI bus into memory prior to launching it. The application must be stored in a memory image format. The initialization routines will not attempt to decode any other type of file format.

The initialization code loads an address into the processor's link register prior to jumping to the user application. If the user application returns, it should first restore the link register to this address so that the processor can cleanly go into an exception state. If the application returns with an invalid address in the link register the behavior of the carrier will be unpredictable. It is rarely necessary for the user application to return.

5.2.4 VXI Word Serial Protocol Handler

The VXI Word Serial Protocol software performs all the necessary functions to send and receive commands and data over the message based interface. It communicates directly with the VXI message based communication registers to handle the word serial protocol defined in the VXIbus specification. The protocol handler is interrupt driven so that it runs independent of the user application. Thus, the user application does not have to manage the minute details of message based communications or even monitor the VXI bus for commands.

For every VXI command or data byte that is written to the VX407C, the system management interrupt (SMI) is asserted causing the word serial protocol handler to execute. The handler communicates with the VXI communications registers to perform the data transfer and to handle the command. When a VXI message is received, the protocol handler will automatically call the user application's command interpreter, if installed. If the user command interpreter is not installed, or if it returns an error, the message will be passed to the system command interpreter. If the message is a system

command it will be handled appropriately, otherwise a VXI error will be reported. The user command interpreter is always called before the system command interpreter so that any system command can be overwritten by the user application if necessary.

The user application can define its own set of VXI message based commands or overwrite any of the system commands. The user command interpreter must handle the command and return status to the protocol handler before any other VXI commands can be received by the instrument. If the user application needs to return data over the message based interface, it simply places data in a buffer using one of the system routines. Refer to section 6.7.1 for details on defining user commands and installing a user command interpreter.

5.2.5 IEEE 488.2 Utilities

The IEEE 488.2 utilities implement the status reporting conventions and required commands defined by the IEEE 488.2 specification. The IEEE 488.2 standard status reporting model illustrated in Figure 27 is used.

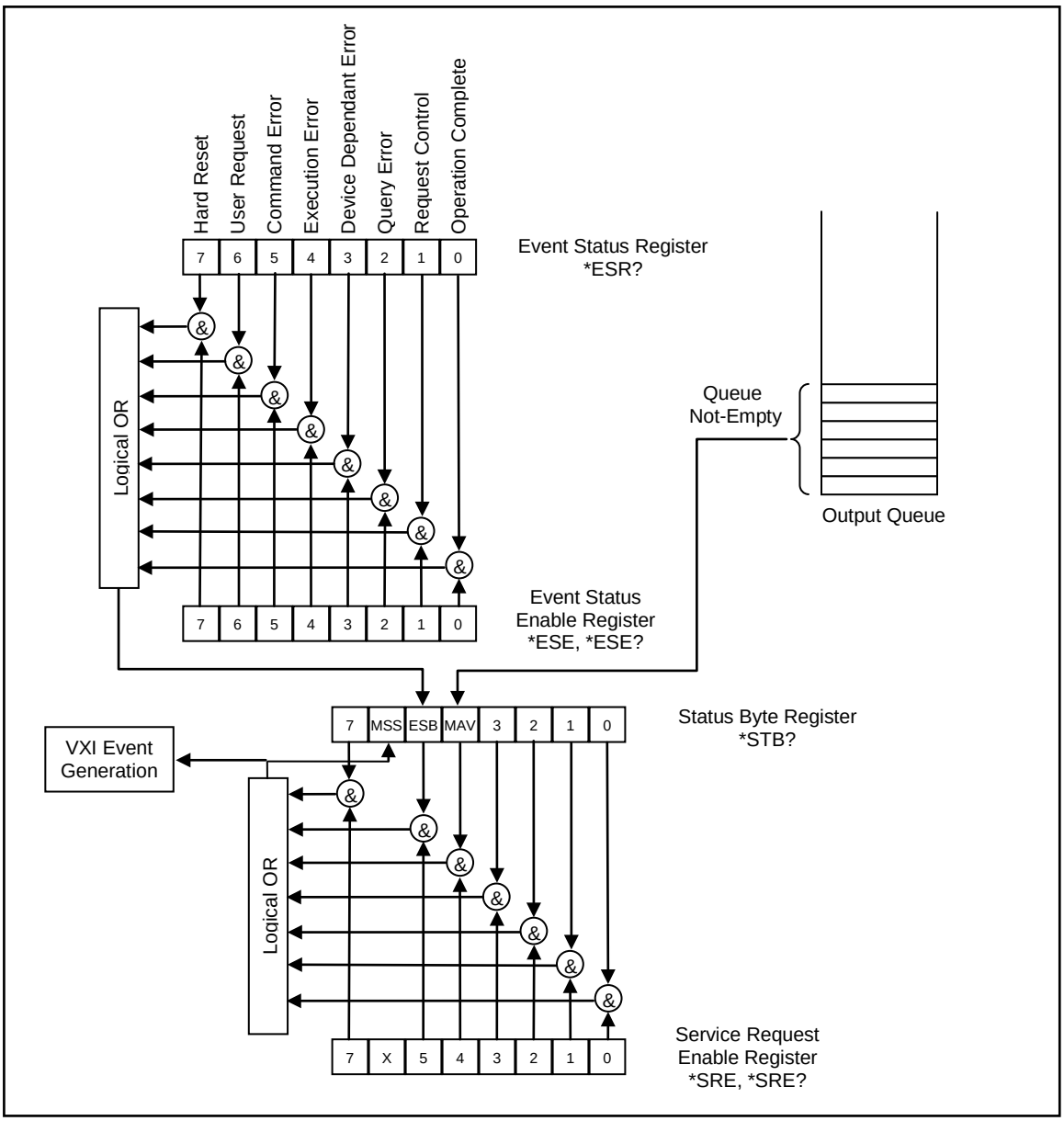


Figure 27. IEEE 488.2 Status Report Model

There are two status registers and two corresponding enable registers. The user application can use system calls to read or modify the values of these registers. The Status Byte Register contains three defined bits as described below. All undefined bits in the Status Byte Register are available as general purpose status bits for use by the user application.

Master Status Summary (MSS): This bit is set to '1' if any other bit in the status byte register is set along with its corresponding enable bit. The rising edge of this bit generates a VXI request true event which can be sent to the VXI host through a VXI interrupt.

Event Status Bit (ESB): This bit is set to '1' if any bit in the Event Status Register is set along with its corresponding enable bit.

Message Available Bit (MAV): This bit is set to '1' whenever data is available in the output queue.

The Event Status Register provides further details on the status of the VX407C. Whenever a bit in this register is set to '1' and its corresponding enable bit is set, the ESB bit in the Status Byte Register is set to a '1'. The Event Status Register is cleared whenever it is read by the VXI host using the *ESR? command. The bit definitions of the Event Status Register are as follows:

Hard Reset: This bit indicates that a hard reset of the carrier has been performed since the last time the Event Status Register was queried. It is cleared after the first read of the Event Status Register (*ESR?) by the VXI host.

User Request: This bit is used to indicate that the user has activated a device-dependant control to request service from the VXI host. The system utilities do not use this bit for any reason; however it is available for use by the user application.

Command Error: This bit indicates that an error was found in a command passed to the instrument. This error is normally the result of an invalid command or a command with incorrect syntax. This bit is automatically set by the system utilities when the user command interpreter returns a command error or when the command is not supported. Refer to the section 6.0 for details on interfacing to the IEEE 488.2 utilities from a user application.

Execution Error: This bit indicates that a valid command was received but that it could not be executed due to some device condition. This bit is automatically set by the system utilities when the user command interpreter returns an execution error. Refer to the section 6.0 for details on interfacing to the IEEE 488.2 utilities from a user application.

Device Dependant Error: This bit indicates that some unspecified device dependant error occurred. The system utilities do not use this bit for any reason; however it is available for use by the user application.

Query Error: This bit indicates that an error occurred while the device was trying to return data to the VXI host. This bit will automatically be set when either the user application or the system utilities attempts to place something in the VXI send buffer and it results in a buffer overflow. The user application can also use this bit for other query errors.

Request Control: This bit indicates that the device wants to become the active controller. This bit is not used since the VX407C does not have controller capabilities.

Operation Complete: This bit indicates that the device has completed all pending operations. The bit is only set in response to the Operation Complete command (*OPC). The user application can take control of this bit by calling the system call function "Wait for Operation Complete". For further details on the "Wait for Operation Complete" system call refer to APPENDIX C. The operation complete bit is automatically cleared whenever the Event Status Register is read by the VXI host using the (*ESR?) command.

All the required IEEE 488.2 common commands, sometimes called star (*) commands, are also implemented in the utilities. These commands perform common device functions and allow the VXI host to interface to the status reporting model described in Figure 27. They are required by any device implementing the IEEE 488.2 conventions. Any of these commands can be overridden in the user application by simply handling the command in the user command interpreter and returning the "success" status value. Table VI lists all of the commands defined in the system utilities. Further details on each command can be found in APPENDIX D.

Table VI. IEEE 488.2 Common Commands

Command	Description
*IDN?	Identification query
*RST	Reset
*TST?	Self test query
*OPC	Operation complete
*OPC?	Operation complete query
*WAI	Wait to continue
*CLS	Clear status
*ESE	Event status enable register
*ESE?	Event status enable register query
*ESR?	Event status register query
*SRE	Service request enable register
*SRE?	Service request enable register query
*STB?	Status byte register query

5.2.6 System Calls

The system calls provide an interface for the user application to the on-board system utilities. They allow the user application to perform complicated tasks without detailed knowledge of the system architecture or knowledge of how the task is performed. For example, the application can send data to the VXI host by simply using a system routine to place data into the send buffer. The user does not need to have any knowledge of how to manipulate the VXI communications registers, as required by the VXIbus specification, to perform the data transfer.

Systems routines are called using the PowerPC's *System Call* (**sc**) instruction, which generates a system call exception (vector $00C00_{16}$). A specific system routine is

specified by programming the general purpose register r10 to the function code of the desired routine prior to executing the system call instruction. Table VII lists all available system calls and their corresponding function codes. Parameters are passed to system call functions using general purpose registers r3-r9 for scalar values and floating point registers f1-f8 for floating point values. The system call functions return data using registers r3, r4, and f1. The system calls are described in detail in APPENDIX C. Refer to each system call's description for details on parameter passing to that particular function.

Table VII. System Calls

System Call	Function Code (hex)	System Call	Function Code (hex)
Get Shared Memory Base Address	0001	End of Interrupt	0044
Peek	0002	Set Current Task Priority	0045
Poke	0003	Interrupt Pending	0046
Self Test	0004	Install Interrupt Handler	0047
		Un-Install Interrupt Handler	0048
VXI Send Message	0010		
VXI Send Buffer Status	0011	Firmware Version	0050
Enable/Disable VXI Word Serial	0012		
Install User Command Interpreter	0013	488.2 Set Event Status Register	0100
Enable/Disable Command Interpreter	0014	488.2 Get Event Status Register	0101
Generate VXI Event Interrupt	0015	488.2 Set Event Status Bit	0102
		488.2 Clear Event Status Bit	0103
Read PCI Config. Offset	0020	488.2 Set Event Status Enable Register	0104
Write PCI Config. Offset	0021	488.2 Get Event Status Enable Register	0105
		488.2 Set Event Status Enable Bit	0106
Flash Write	0030	488.2 Clear Event Status Enable Bit	0107
Flash Read	0031	488.2 Set Status Register	0108
Flash Erase Sector	0032	488.2 Get Status Register	0109
Flash Block Write	0033	488.2 Set Status Bit	0110
		488.2 Clear Status Bit	0111
Configure Interrupt	0040	488.2 Set Status Enable Register	0112
Enable Interrupt	0041	488.2 Get Status Enable Register	0113
Disable Interrupt	0042	488.2 Set Status Enable Bit	0114
Acknowledge Interrupt	0043	488.2 Clear Status Enable Bit	0115

5.2.7 System Commands

The system commands allow the VXI host to configure the VX407C and to perform basic communications with the instrument regardless of the PXI/cPCI modules installed. The system command interpreter will handle these commands as they arrive through the word serial protocol handler. Table VIII lists all available system commands. Each command is described in detail in APPENDIX D.

Table VIII. System Commands

Command	Parameters
SYSTem	
:PEEK?	<address>, <width>
:POKE	<address>, <width>, <data>
:FLASH	
:WRITE	<offset>, <data>
:READ?	<offset>
:ERASE	<sector>
:BLOCK	<offset>,<data ptr>,<num bytes>
:PCI	
:CONFIGure	
:READ?	<bus>, <device>, <func>, <width>, <offset>
:WRITE	<bus>, <device>, <func>, <width>, <offset>, <data>
:SCAN?	
:CONFIGure	
:BOOT	
:TYPE[?]	<type>
:ADDR[?]	<address>
:VXI	
:TYPE[?]	<type>
:MANF[?]	<manufacturer id>
:MODEL[?]	<model code>
:WIDTH[?]	<A24/A32 width>
:DEFAULT	
:VXI	
:WIDTH[?]	<width>
:VER?	
:DOWNLOAD	
:ERR?	

5.3 USER APPLICATION

The user application, if present, provides the central control for all the software running on the PowerPC. The responsibilities of the user application will vary greatly form one application to the next. Common task might include command interpretation, interrupt handling, data manipulation, and instrument control. The application should make system calls to access the on-board system utilities when performing necessary tasks.

The application can be stored anywhere in memory or can be downloaded over the VXI bus after reset. The application is responsible for loading itself into RAM, setting up the general purpose registers for use, initializing its stack and heap, and allocating memory. Most off-the-shelf development tools contain libraries that can be linked with the application to perform these tasks automatically.

The application can define a set of VXI commands for communication between VXI host and the device. To receive commands, the application must install a user command interpreter. This function will be automatically called whenever a message is received over the VXI bus. The user command interpreter must handle the command and return status before another VXI command can be received. Refer to section 6.7.1 for details.

6.0 PROGRAMMING INSTRUCTIONS

6.1 MAKING SYSTEM CALLS

System calls are designed around the Embedded Application Binary Interface (EABI) specification. Conventions defined by this specification include, among others: register usage, parameter passing, and stack organization. Specifically, the conventions used for parameter passing allow applications built using an EABI compliant development tool to easily interface to the system calls as they would any other library function. Most off-the-shelf development tools for the PowerPC are EABI compliant.

To make a system call, first program the general purpose registers r3-r9 and f1-f8 with the appropriate parameter values. Second, program the general purpose register r10 to the desired function code. Finally, perform the *System Call* (sc) instruction. When the system routine completes it will return to the instruction following the system call instruction and the application can continue normally. Any returned data will be placed in the general purpose registers r3, r4, or f1. APPENDIX C discusses each system routine in detail including register usage for parameter passing and returning data. Figure 28 shows the assembly level code necessary to generate a system call.

```
_system_call:
     addi  r3, 0, 0xNNNN     Initialize Function Parameters
     ...
     addi  r10, 0, 0xNNNN    Set Function Code in r10
     sc                      Generate System Call
```

Figure 28. System Call Example Code

6.2 FLASH PROGRAMMING

Programming a single byte in flash requires sending a stream of commands to the device. System routines are provided in the on-board system utilities to perform flash programming so that the user does not need to know the specifics of the device protocol.

The device is organized into sectors that are 64 kilobytes each. The first sector is reserved for system utilities and should never be programmed or erased. The sectors are organized sequentially in memory so that sector 0 is from address 00_0000_{16} – 00_FFFF_{16}, sector 1 is from address 01_0000_{16} – 01_FFFF_{16} and so forth. The 64 kilobytes are mapped to the PowerPC address space starting at address $FF00\_0000_{16}$.

Programming operations can be performed on any address in the flash device. Only 8-bit accesses are supported. ***The programming operation can only toggle a bit from '1' to '0'***. To set a bit back to '1' an erase operation must be performed. Erase operations can only be performed on a sector by sector basis. Therefore, in most cases it is necessary to

erase an entire sector and rewrite it to change a single byte within that sector. The flash write system call will not determine if an erase operation is necessary. It will simply attempt to write the data to the specified address. A sector erase system routine is provided to perform the erase operation.

The VXI host can read and write the flash memory by instructing the PowerPC to perform the access using the system commands. Reads can be performed using the "PEEK?" command or the "FLASH:READ?" command. The "FLASH" group of commands also provides the capability to perform a flash write, erase, and block write.

6.3 PCI ACCESSES

Devices on the PCI bus are mapped into the PowerPC's address space. Therefore, accessing these devices (except for configuration accesses) is as simple as performing a standard memory read or write. The base address for the device, in the PowerPC's memory space, is determined during PCI bus enumeration and is dependant upon the resources required by all the devices on the bus. A PCI configuration read can be used to determine the base address of a specific device. Every PCI device is required to have a set of Base Address Registers (BAR) that the PCI controller configures during bus initialization. These registers determine the base address(s) of the resource(s) on the device. Each BAR can point to either PCI memory space or PCI I/O space. If bit 0 (the least significant bit) of the base address register is a '1', the resource is mapped to PCI I/O space. Otherwise the resource is mapped to PCI memory space. PCI memory space is mapped directly into the PowerPC's address map (i.e. PowerPC address 9000_0000_{16} is mapped to the same address in PCI memory space). PCI I/O space, however, is mapped relative to a base address of $FE00\_0000_{16}$ (i.e. PowerPC address $FE80\_0000_{16}$ is mapped to PCI I/O address 0080_0000_{16}). For details on the PowerPC address map refer to Figure 12. For information on a particular device's BAR registers, refer to the PCI device's documentation.

Two accesses are required to perform a single PCI configuration write or read. The PCI configuration address register at PowerPC address $FEC0\_0000_{16}$ must first be set to point to the correct device and offset. Then the data can be read from or written to the PCI configuration data register at PowerPC address $FEE0\_0000_{16}$. The PCI configuration address register value is determined by the device number, IDSEL signal routing, device function number, and the register offset. IDSEL signal routing information can be found in section 4.2.5.2 of this document. For further details on performing PCI configuration accesses refer to the MPC8245 User's Manual. System routines are provided to perform configuration reads and writes from the PowerPC.

PCI accesses can be performed by the VXI host using the system commands. Memory and I/O accesses can be performed using the "PEEK?" and "POKE" commands and configuration accesses can be performed using the "PCI:CONFIG" subset of commands. A list of all PCI devices on the bus can also be received by the VXI host using the "PCI:SCAN?" system command.

6.4 FIRMWARE DOWNLOAD

The firmware download mode allows the user to download code and data over the VXI bus via shared memory. The destination of the download can be flash, RAM or any other storage location in the PowerPC's address space. The PowerPC will go into firmware download mode when either the "DOWNLOAD" system command is received or automatically if the Boot Type configuration option is set to download at reset.

A special firmware download mode also exists that allows the user to update the on-board system utilities residing in the first sector of flash. This mode is automatically entered when the Update System Firmware switch is set to the OFF position at reset. When the update is complete, the system utilities will automatically be launched as normal. Unless the switch is set back to the ON position the VX407C will automatically launch back into the update system utilities mode at reset.

6.4.1 Firmware Download Mode Protocol

The firmware download routines use the general purpose shared memory as a buffer between the host and the PowerPC. The protocol divides the 16 kilobytes of memory into four 4 kilobyte banks as shown in Figure 29. Each bank has two associated ownership bits in the arbitration utility flag register at shared memory offset $4C0_{16}$. One bit signifies ownership of the associated bank by the host and the other signifies ownership of the bank by the PowerPC. The register functions such that the host cannot take ownership of a bank that the PowerPC has ownership of and vice-versa. Software must guarantee that it does not write to or read from a bank that it does not have ownership of. Further details of the arbitration utility flag register are discussed in section 4.4.3.1.

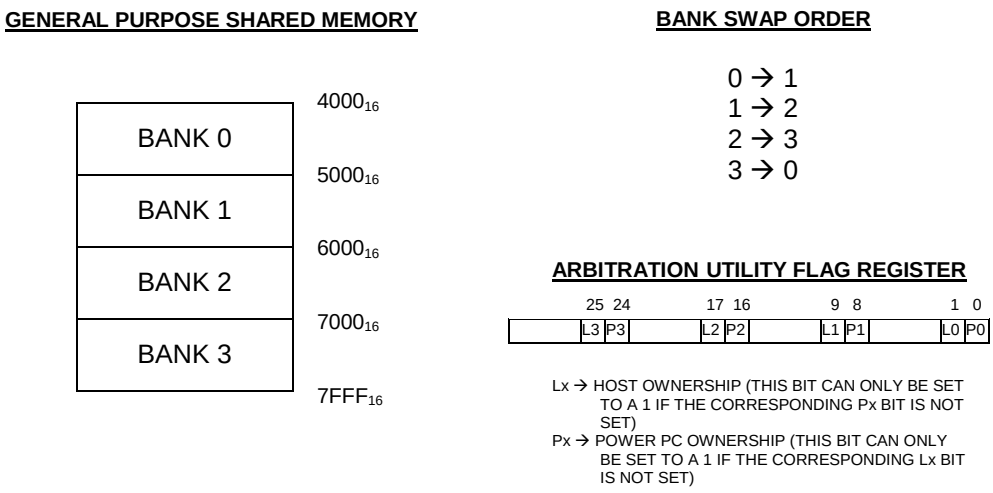


Figure 29. Shared memory banks for firmware update

The download protocol is illustrated in the flow chart of
Figure 30. The host computer takes ownership of bank 0 to signify that it is ready to download data. The PowerPC application must wait until it has verified that bank 0 is no longer free. The PowerPC application then takes ownership of bank 3 to signify that it is ready. Like the PowerPC application, the host application must wait until it has verified that bank 3 is no longer free. At this point the download can begin.

The format of the data depends on the type of download command being performed. In all cases the data starts with a command code and ends with a Cyclic Redundancy Check (CRC) value. To transfer the data, the host simply places it in consecutive address locations of shared memory. When the host fills an entire bank it must perform a bank swap by first, verifying the next bank is free, second, taking ownership of the bank, then finally, releasing ownership of the completed bank. When a bank becomes free, the PowerPC must take ownership of it prior to reading the data. Like the host, the PowerPC application must verify that it has ownership of the next bank prior to releasing the current bank. Bank swapping must be done in the order illustrated in Figure 29 in order to preserve data integrity. When the download is complete the PowerPC application verifies the CRC value and returns an acknowledge (ACK) or negative acknowledge (NACK) value to the host. Both the host and the PowerPC must release all bank ownerships prior to returning from their respective download routines.

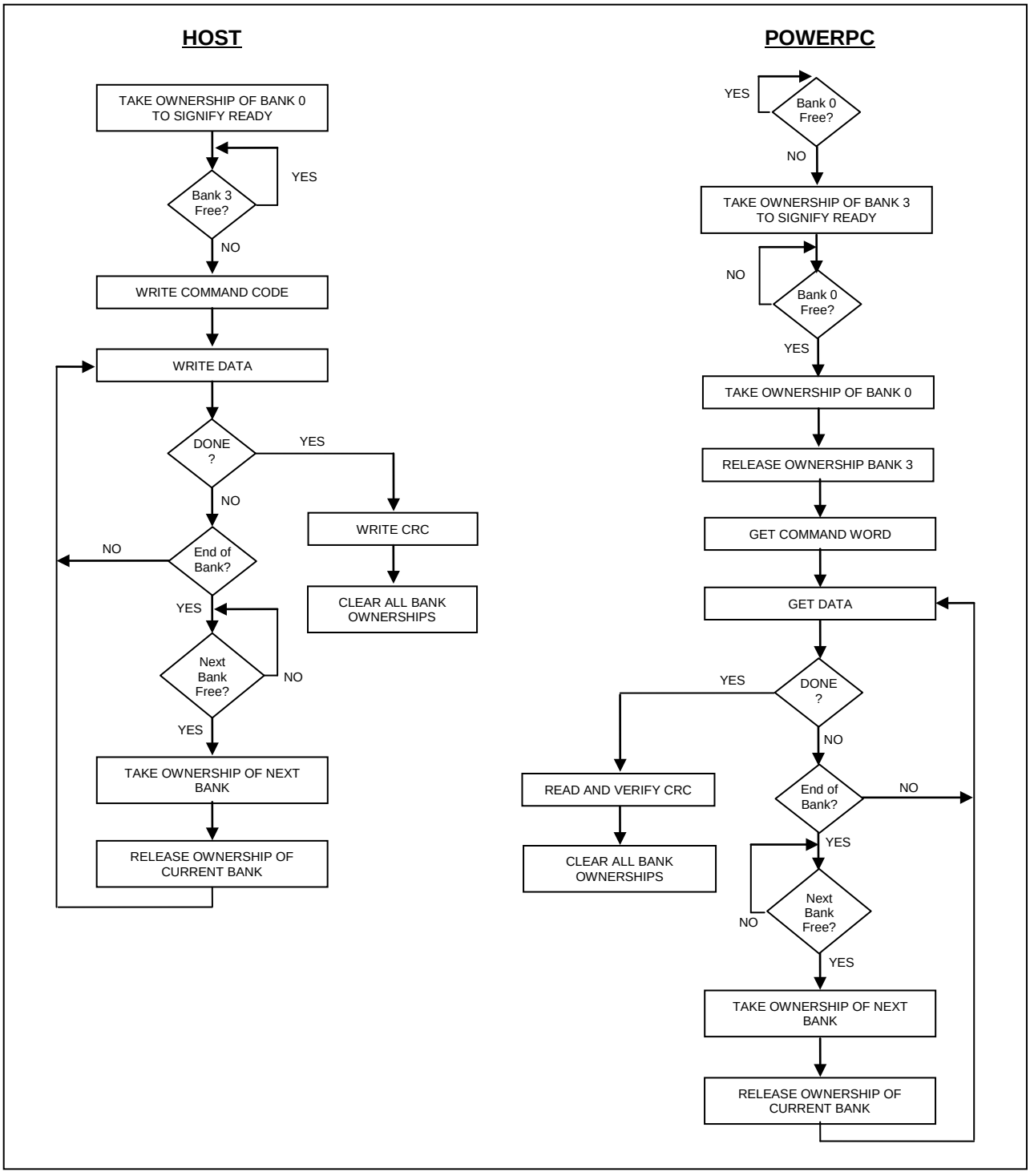


Figure 30. Firmware Update Protocol

6.4.2 Download Commands

Download commands identify the format of the data being downloaded and instruct the PowerPC what to do with the data. Each command format begins with a 32-bit command code and ends with a 32-bit CRC value. The data between the command code and the CRC can be a variable sizes and meaning depending on the command. The download protocol illustrated in Figure 30 must be followed for each download command. The behavior of the PowerPC firmware after the command is complete is dependant upon the type of command.

6.4.2.1 Generic Download Command

The generic download command allows the user to download any size block of data to be written to any PowerPC addressable location. The PowerPC maintains an address counter that is initialized to the start address value. For each byte of data received the PowerPC performs a simple memory write to the location of the address counter and increments the counter. The PowerPC does not verify that the memory write was successful or that a storage device even exists at the memory location. A running CRC is calculated as each byte is read from shared memory. This calculated CRC value is compared to the CRC value received as part of the download to determine whether or not the download was successful. The format of the generic download command is illustrated in Figure 31.

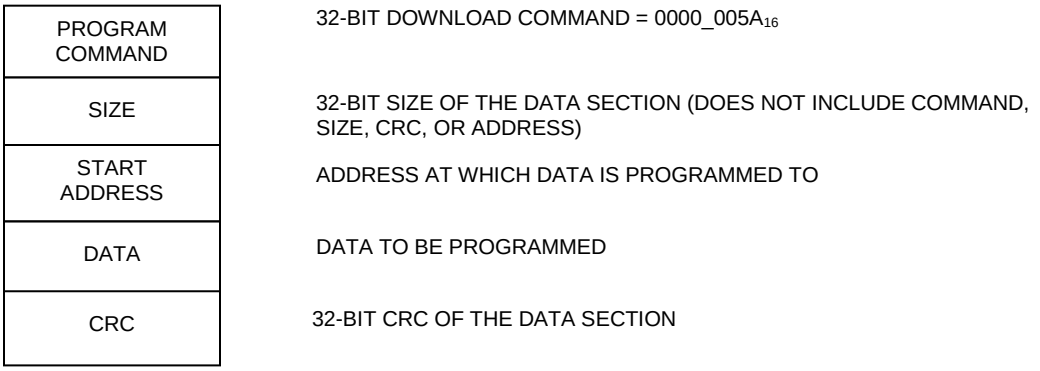


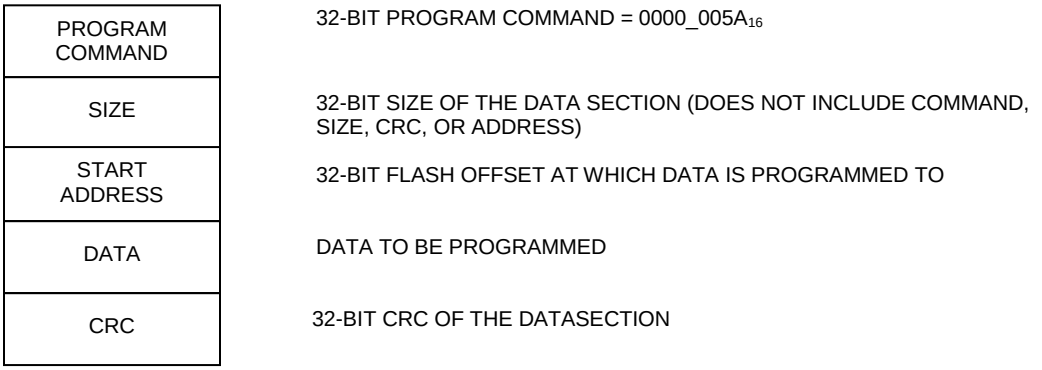
Figure 31. Generic Download Command

6.4.2.2 Flash Program Command

The flash program command allows the user to download data to be written to the flash device. The PowerPC behaves exactly as it does with the generic download command except that instead of performing a simple memory write, it performs a flash write. The start address value must point to an offset within the flash device and not an absolute PowerPC address. Also, the start address must not be within the first sector of the flash

device unless the PowerPC is in a special update system firmware mode as discussed in section 6.4.

The PowerPC does not erase the sector prior to programming the flash unless it is in the update system firmware mode. The flash sector erase command is provided for that purpose. Since the flash write function cannot toggle a bit from a '0' to a '1' the operation might fail if the sector has not been erased before this command is received. The flash program command is illustrated in figure Figure 32.

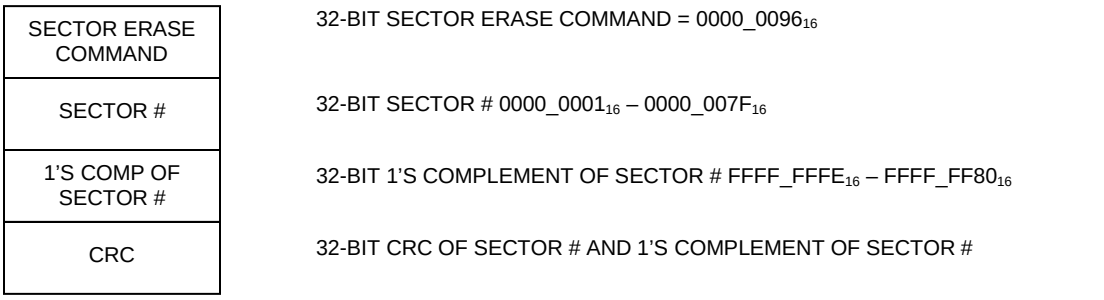


Note: The start address may be within the first sector of flash (00_0000$_{16}$ – 00_FFFF$_{16}$) only if the processor is in the update system firmware mode.

Figure 32. Flash Program Command

6.4.3 Flash Sector Erase Command

The flash sector erase command instructs the PowerPC to erase the specified sector in the flash device. A flash write operation can only toggle a bit from a '1' to a '0'. An erase operation has to be performed to toggle and bits back to a '1'. A sector is the smallest block of flash memory that can be erased by the erase operation. It is recommended that the sectors being programmed are erased prior to sending the flash program command. The format of the flash sector erase command is illustrated in Figure 33.



Note: The sector # may specify the first sector (sector 0) only if the processor is in the update system firmware mode.

Figure 33. Flash Sector Erase Command

6.4.3.1 Boot Command

The boot command instructs the PowerPC to exit the firmware download mode and launch run the application residing at the specified boot address. This command allows the user to download a user application into volatile memory, such as SDRAM, and subsequently run the application. The format of the boot command is illustrated in Figure 34.

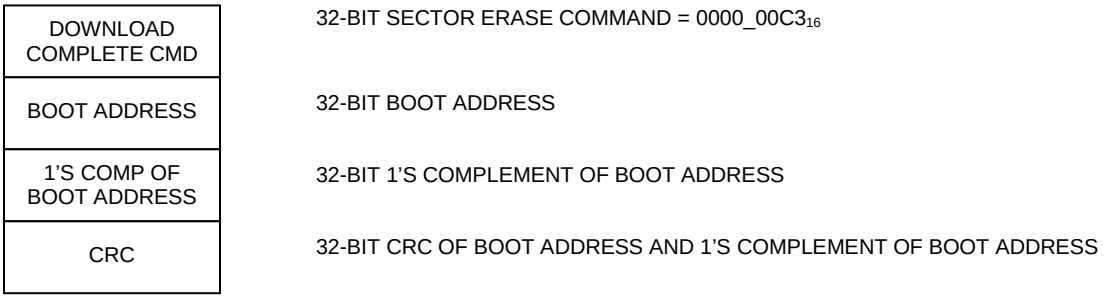


Figure 34. Boot Command

6.5 INTERRUPTS

The PowerPC application will handle all interrupts from the on-board PCI devices. In addition, the PowerPC can generate a VXI interrupt by accessing an operations register. Finally, the shared memory device can also interrupt the VXI host on various programmable conditions such as the completion of a DMA transfer.

6.5.1 PCI Interrupts

PCI interrupts are handled by the Embedded Programmable Interrupt Controller (EPIC) of the MPC8245. The interrupt lines are routed between the PCI devices and the EPIC's interrupt lines as described in Table III. Each interrupt line must be set to direct input mode so that the EPIC controller can respond to interrupts from the PCI devices. The interrupt lines can also be prioritized and can be programmed for level or edge sensitivity and either polarity.

A set of system calls are provided to assist a user application when programming and handling interrupts. The system calls allow the application to assign a vector and a priority level to each interrupt, enable the interrupt, acknowledge the interrupt, and clear the interrupt. System calls also exist that will install and uninstall an interrupt handling routine. All PCI interrupts will be handled by a single interrupt routine. This routine can perform an interrupt acknowledge to get an interrupt vector and, based on the vector, call a sub-function to handle the interrupt for the specific PCI device.

Further details on programming the EPIC controller can be found in the MPC8245 User's Manual.

6.5.2 VXI Interrupts

The source of VXI interrupts can be the on-board system firmware, the user application, or the shared memory device. In all cases the Interrupt Control register at offset in the VX407C's operations register provides the control for the interrupt. All cases share a single interrupt line and the cause of the interrupt can be determined from the status/ID value returned to the host during the interrupt acknowledge cycle.

The interrupt priority level used by the VX407C is programmable using the 'VXI Level' field of the interrupt control register. Any level between 1 and 7 can be selected. Writing a value of '0' to this field will disable the VXI interrupts. The Master Interrupt Enable (MIE) bit must also be set to a value of '1'for interrupts to occur. In addition, there is an interrupt enable bit for shared memory interrupts and one for processor based interrupts. Both the system firmware and the user application use processor based interrupts.

The VXI interrupt is always automatically released by the VX407C during the interrupt acknowledge cycle (i.e. ROAK). To achieve this behavior, the carrier automatically clears the MIE bit during the acknowledge cycle. The interrupt handler routine should re-enable this bit, after performing the interrupt handling functions, if interrupts are to continue to occur. ***If an interrupt is still pending when the MIE bit is re-enabled another interrupt will immediately be generated.*** The method of clearing a pending interrupt is dependent upon the type of interrupt.

The system firmware can generate VXI response and event interrupts as specified by the VXIbus specification for message based instruments. These are low level interrupts that allow the instrument to notify the host computer when certain conditions of the word serial protocol are met or when protocol errors occur. In most cases, the host application will not handle these interrupts. Instead a low level VXI library will enable/disable and process these types of interrupts. The firmware generates these interrupt by writing a defined vector to the vector field in the Interrupt Control register and setting the Processor Interrupt Pending (PIP) bit to a '1'.

The PowerPC user application can also interrupt the VXI host. It does so, much like the system firmware does, by setting the interrupt vector value in the Interrupt Pending register to any user defined vector values and setting the PIP bit to a '1'. The Processor Interrupt Enable (PIE) bit must also be set to a '1'. The user application can provide any interrupt vector within the range of user defined vectors. All other vector values are defined by the VXI specification or reserved system use. Figure 21 show the defined vector values. Clearing the interrupt is achieved by setting the PIP bit to a '0'. The method for instructing the PowerPC to clear the interrupt is application specific.

The shared memory device can interrupt the VXI host for several reasons including to signify the completion of a DMA transaction. If the Shared Memory Interrupt Enable (SMIE) bit is set to a '1' all interrupts from the shared memory device will be forwarded to the VXI bus. Shared memory interrupts have priority of processor based interrupts. The vector returned during the interrupt acknowledge cycle will always be 80_{16}. Configuring the shared memory device to generate interrupts is done through the device's operation registers.

6.6 CONFIGURING TRIGGERS

Triggers are configured using the VX407C Control/Status registers and the Trigger Control register at offsets 20_{16} and 24_{16} of the operations registers. The trigger architecture for the carrier consists of two programmable switching matrices as described in section 4.2.6. For both matrices, each PXI trigger can be mapped to a VXI trigger, can be enabled or disabled, and can be inverted.

To configure a specific trigger the user must first write to the VX407C Control/Status register with the TRIGSEL value set to the desired trigger then configure the selected trigger by writing to the Trigger Control register. Figure 21 describes the Trigger Control register and the VX407C Control/Status register in detail.

The current configuration for a specific trigger can be read by first setting the TRIGSEL value in the VX407C Control/Status register to the desired trigger then reading the Trigger Control register. Once the TRIGSEL value is set, any read or a write of the Trigger Control register will access the configuration for that trigger until TRIGSEL is either written another value or the carrier is reset.

6.7 VXI WORD SERIAL COMMUNICATIONS

The VXI word serial communications routines are designed to run without any intervention from the user application. The entire communications process is driven by the system management interrupt (SMI) which is asserted by the VXI interface logic every time a VXI command or data is received. ***The user application should not disable interrupts or modify the SMI exception vector.*** Doing so will cause the VX407C to not respond to VXI commands and most likely cause a VXI communications error.

The application can, however, disable the VXI communications interface using the *Enable/Disable VXI Word Serial* system call. This routine will program the VXI response register to notify the VXI host that the carrier will not respond to any VXI commands including low level protocol commands not normally visible to either the host or device-side application. This routine should only be called when the user application is running critical code and cannot be interrupted for any reason.

Alternatively, the *Enable/Disable Command Parser* system routine will disable the response to high level messages while still responding to low level protocol commands. Any command consisting of a string of characters is considered a high level message. This routine should be called whenever the instrument or the application goes into a state where the user command parser cannot respond to messages.

6.7.1 User Command Interpreter

User commands can be defined to be any string of ASCII characters. If installed, the user command interpreter will automatically be called every time a command is received. The user command interpreter should determine if the message is a supported command and either return an error or perform the command. The user command interpreter is always called prior to the system command interpreter thus allowing the user application to override any existing system commands if necessary.

In order for a user command interpreter to be called it must be installed by the user application using the system routine *Install User Command Interpreter*. This system routine takes a pointer to a function as a parameter. The user command interpreter must satisfy the following conventions:

1. A pointer to the message string is passed to the function in general purpose register r3 as the first parameter.
2. The maximum length of the message string will be 256 bytes. The user command interpreter must not attempt to access beyond the end of the string.

3. The function must return one of the valid status values using general purpose register r3. Valid status values are:
    - 0 = Success
    - -1 = Unsupported Command
    - -2 = Multiple Query
4. If the command places the processor in a state such that it cannot respond to other commands it must call the system routine *Enable/Disable Command Parser* before returning. The parser can be re-enabled at some later point.
5. The carrier will notify the VXI host that it cannot respond to and VXI commands while in the command parser. Therefore the parser should execute quickly and return. Otherwise VXI bus timeouts will occur.

The required conventions are compatible with the Embedded Applications Binary Interface (EABI) specification. Therefore the user command interpreter can be written in a high level language using a development environment that is EABI compatible. Example C code for the user command interpreter can be found in Figure 35.

```
#define     VXI_SUCCESS                      0
#define     VXI_ERROR_MULTIPLE_QUERY         -1
#define     VXI_ERROR_UNSUPPORTED_COMMAND    -2

int UserCommandInterpreter(unsigned char *message)
{  int status = VXI_SUCCESS;

   if(strncmp(command, "PXI:", 4) == 0)
   {  if(strncmp(&command[4], "READ?", 5) == 0)
      {  if(vxi_SendBufferFull())
               error = VXI_ERROR_MULTIPLE_QUERY;
         else
            pxi_Read();
      }
         else if(strncmp(&command[4], "WRITE", 5) == 0)
      {  pxi_Write();
      }
      else
             status = VXI_ERROR_UNSUPPORTED_COMMAND;
   }
   else
      status = VXI_ERROR_UNSUPPORTED_COMMAND;

   return status;
}
```

Figure 35. User Command Interpreter Example Code

6.8 HOST-SIDE PCI BUS MASTERING AND DIRECT ACCESS

The shared memory device provides a 8 Kilobyte window into PCI memory space. By accessing this window in VXI A24/A32 space, the host PC has direct access to the PXI/cPCI modules. To point the 8 Kilobyte window to the correct PCI bus address the host must control the direct access operational register defined by the shared memory device. Figure 36 shows a description of the control register.

Direct Access Register

0460_{16}

Bit	31 ... 13	12	11	10	9 ... 8	7 ... 4	3	2 ... 1	0
Write	PCI Physical Base Address	-	F	-	A1A0	Byte Enables for Reads	-	Type	-
Read	PCI Physical Base Address	-	F	-	A1A0	Byte Enables for Reads	-	Type	-

PCI Physical Base Address ⇨ PCI physical base address of the 8K byte direct access window at VXI A24/A32 offset 0x2000

F ⇨ Force contents of A1A0 to PCI during PCI address phase (0 = don't force, 1 = force)

A1A0 ⇨ Value of PCI A1 and PCI A0 to be placed on the PCI bus if F = 1

Byte Enables for Reads ⇨ Data Byte Enables for PCI Master Reads, C/BE#[3:0]

Type ⇨ PCI Command Type for PCI Master Access

- 00 Interrupt Acknowledge (read) (PCI command 0x0)
  Special Cycle (write) (PCI command 0x1)
- 01 I/O Cycle (read/write) (PCI command 0x2 or 0x3)
- 10 Memory Cycle (read/write) (PCI command 0x6 or 0x7)
- 11 Configuration Cycle (read/write) (PCI command 0xA or 0xB)

Figure 36. Direct Access Control Register

Before the shared memory device can generate a PCI access its Master Enable bit must be set in the PCI configuration registers. This bit is set by default after power-up during the configuration of the shared memory device. The master enable bit can only be cleared by the PowerPC. If for some reason this occurs the VXI host must request that the PowerPC re-enable this bit before it can perform a direct access. Even if the PowerPC's boot procedure fails, the Master Enable bit will be set by default and the VXI host will automatically gain control of the PCI bus.

The procedure for directly accessing the onboard PCI bus is as follows:

1) Make sure the Master Enable Bit in the shared memory's PCI configuration space is set.
2) Program the Direct Access Register with the desired PCI system's physical base address. This is the 8 kilobyte address block that the Direct Access Window points to.

2a) In the Direct Access Register, set the type of PCI command to be generated.

2b) In the Direct Access Register, set the byte enables for the desired accesses if they are to be read accesses.

3) Access the PCI bus by reading or writing to the 8 kilobyte Direct Accesses Window.

Note: Step 2 should be repeated if a different 8 kilobyte area of PCI space needs to be accessed or if a different access type or byte enables are needed.

6.8.1.1 PCI Configuration Accesses

PCI configuration cycles use a different addressing method than normal PCI command cycles. During the cycle, each device is selected by asserting a unique IDSEL line. The PCI specification does not stipulate the routing of the IDSEL signals however in most systems the IDSEL line for a given device is connected to one of the upper PCI address bits. Refer to Table II in Section 4.2.5 for details on IDSEL signal routing on the VX407C.

To perform a type 0 PCI configuration cycle perform the following steps:

1) Determine which address bit must be asserted in order to assert the IDSEL line of the desired device. Use Table II as a reference.
2a) Set the correct bit in the PCI Physical Base Address section of the Direct Access Register. Only set one bit to select the PCI device.
2b) Set the 'F' bit (bit 11) in the Direct Access Register to 0.
2c) Set the access type in the Direct Access Register to '11' which specifies a configuration cycle
3) Access the 8 kilobyte direct access window at an offset that incorporates the function number (bits 10-8) and the register offset (bits7-0). Determine the A24/A32 offset to read or write to using the following formula:

$$A24/A32\ Offset = 2000_{16} + (FuncNum \times 256) + Reg\ Offset$$

6.8.1.2 Byte Enables in a Direct Access Cycle

The byte enable field of the Direct Access Register is only used during a read access. Byte enable for write accesses are determined by the type of VXI bus access. The byte enable field is applied directly to the C/BE# signals on the PCI bus during direct access reads from the VXI host. Since the PCI byte enables are active low, the byte enable bits of the control register are active low. If a target supports pre-fetching, it will return all bytes regardless of byte enables.

This page was left intentionally blank.

APPENDIX A CONNECTORS

PIN	Z	A	B	C	D	E	F
1	GND	5V	-12V	TRST#	+12V	5V	GND
2	GND	TCK	5V	TMS	TDO	TDI	GND
3	GND	INTA#	INTB#	INTC#	5V	INTD#	GND
4	GND	*BRSVP1A4*	GND	V(I/O)	INTP	INTS	GND
5	GND	*BRSVP1A5*	*BRSVP1B5*	RST#	GND	GNT#	GND
6	GND	REQ#	GND	3.3V	CLK	AD[31]	GND
7	GND	AD[30]	AD[29]	AD[28]	GND	AD[27]	GND
8	GND	AD[26]	GND	V(I/O)	AD[25]	AD[24]	GND
9	GND	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]	GND
10	GND	AD[21]	GND	3.3V	AD[20]	AD[19]	GND
11	GND	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#	GND
12 13 14	Key Area						
15	GND	3.3V	FRAME#	IRDY#	GND	TRDY#	GND
16	GND	DEVSEL#	GND	V(I/O)	STOP#	LOCK#	GND
17	GND	3.3V	SDONE	SBO#	GND	PERR#	GND
18	GND	SERR#	GND	3.3V	PAR	C/BE[1]	GND
19	GND	3.3V	AD[15]	AD[14]	GND	AD[13]	GND
20	GND	AD[12]	GND	V(I/O)	AD[11]	AD[10]	GND
21	GND	3.3V	AD[9]	AD[8]	M66EN	C/BE[0]#	GND
22	GND	AD[7]	GND	3.3V	AD[6]	AD[5]	GND
23	GND	3.3V	AD[4]	AD[3]	5V	AD[2]	GND
24	GND	AD[1]	5V	V(I/O)	AD[0]	*ACK64#*	GND
25	GND	5V	*REQ64#*	*ENUM#*	3.3V	5V	GND

Figure A-1. PXI/CPCI Slot B P1 (P1B) Pin Configuration

PIN	Z	A	B	C	D	E	F
1	GND	***PXI_LBL9***	GND	***PXI_LBL10***	***PXI_LBL11***	***PXI_LBL12***	GND
2	GND	***PXI_LBR11***	***PXI_LBR12***	SYSEN#	***PXI_LBL7***	***PXI_LBL8***	GND
3	GND	***PXI_LBR7***	GND	***PXI_LBR8***	***PXI_LBR9***	***PXI_LBR10***	GND
4	GND	V(I/O)	***PXI_RSVB4***	*C/BE[7]#*	GND	*C/BE[6]#*	GND
5	GND	*C/BE[5]#*	GND	V(I/O)	*C/BE[4]#*	*PAR64*	GND
6	GND	*AD[63]*	*AD[62]*	*AD[61]*	GND	*AD[60]*	GND
7	GND	*AD[59]*	GND	V(I/O)	*AD[58]*	*AD[57]*	GND
8	GND	*AD[56]*	*AD[55]*	*AD[54]*	GND	*AD[53]*	GND
9	GND	*AD[52]*	GND	V(I/O)	*AD[51]*	*AD[50]*	GND
10	GND	*AD[49]*	*AD[48]*	*AD[47]*	GND	*AD[46]*	GND
11	GND	*AD[45]*	GND	V(I/O)	*AD[44]*	*AD[43]*	GND
12	GND	*AD[42]*	*AD[41]*	*AD[40]*	GND	*AD[39]*	GND
13	GND	*AD[38]*	GND	V(I/O)	*AD[37]*	*AD[36]*	GND
14	GND	*AD[35]*	*AD[34]*	*AD[33]*	GND	*AD[32]*	GND
15	GND	***PXI_BRSVA15***	GND	FAL#	***PXI_LBL6***	***PXI_LBR6***	GND
16	GND	**PXI_TRIG1**	**PXI_TRIG0**	DEG#	GND	**PXI_TRIG7**	GND
17	GND	**PXI_TRIG2**	GND	PRST#	**PXI_STAR**	**PXI_CLK10**	GND
18	GND	**PXI_TRIG3**	**PXI_TRIG4**	**PXI_TRIG5**	GND	**PXI_TRIG6**	GND
19	GND	***PXI_LBL2***	GND	***PXI_LBL3***	***PXI_LBL4***	***PXI_LBL5***	GND
20	GND	***PXI_LBR4***	***PXI_LBR5***	***PXI_LBL0***	GND	***PXI_LBL1***	GND
21	GND	***PXI_LBR0***	GND	***PXI_LBR1***	***PXI_LBR2***	***PXI_LBR3***	GND
22	GND	***PXI_RSVA22***	***PXI_RSVB22***	***PXI_RSVC22***	***PXI_RSVD22***	***PXI_RSVE22***	GND

Figure A-2. PXI/CPCI Slot B P2 (P2B) Pin Configuration

NOTES: Bold-faced words are PXI defined signals.
Italicized words are unused signals.
Underlined words are used by the system slot.

PIN	Z	A	B	C	D	E	F
1	GND	5V	-12V	TRST#	+12V	5V	GND
2	GND	TCK	5V	TMS	TDO	TDI	GND
3	GND	INTA#	INTB#	INTC#	5V	INTD#	GND
4	GND	*BRSVP1A4*	GND	V(I/O)	INTP	INTS	GND
5	GND	*BRSVP1A5*	*BRSVP1B5*	RST#	GND	GNT#	GND
6	GND	REQ#	GND	3.3V	CLK	AD[31]	GND
7	GND	AD[30]	AD[29]	AD[28]	GND	AD[27]	GND
8	GND	AD[26]	GND	V(I/O)	AD[25]	AD[24]	GND
9	GND	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]	GND
10	GND	AD[21]	GND	3.3V	AD[20]	AD[19]	GND
11	GND	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#	GND
12 13 14	Key Area						
15	GND	3.3V	FRAME#	IRDY#	GND	TRDY#	GND
16	GND	DEVSEL#	GND	V(I/O)	STOP#	LOCK#	GND
17	GND	3.3V	SDONE	SBO#	GND	PERR#	GND
18	GND	SERR#	GND	3.3V	PAR	C/BE[1]	GND
19	GND	3.3V	AD[15]	AD[14]	GND	AD[13]	GND
20	GND	AD[12]	GND	V(I/O)	AD[11]	AD[10]	GND
21	GND	3.3V	AD[9]	AD[8]	M66EN	C/BE[0]#	GND
22	GND	AD[7]	GND	3.3V	AD[6]	AD[5]	GND
23	GND	3.3V	AD[4]	AD[3]	5V	AD[2]	GND
24	GND	AD[1]	5V	V(I/O)	AD[0]	*ACK64#*	GND
25	GND	5V	*REQ64#*	*ENUM#*	3.3V	5V	GND

Figure A-3. PXI/CPCI Slot A P1 (P1A) Pin Configuration

PIN	Z	A	B	C	D	E	F
1	GND	***PXI_STAR9***	GND	***PXI_ STAR10***	***PXI_ STAR11***	***PXI_ STAR12***	GND
2	GND	***PXI_LBR11***	***PXI_LBR12***	SYSEN#	***PXI_ STAR7***	***PXI_ STAR8***	GND
3	GND	***PXI_LBR7***	GND	***PXI_LBR8***	***PXI_LBR9***	***PXI_LBR10***	GND
4	GND	V(I/O)	***PXI_RSVB4***	*C/BE[7]#*	GND	*C/BE[6]#*	GND
5	GND	*C/BE[5]#*	GND	V(I/O)	*C/BE[4]#*	*PAR64*	GND
6	GND	*AD[63]*	*AD[62]*	*AD[61]*	GND	*AD[60]*	GND
7	GND	*AD[59]*	GND	V(I/O)	*AD[58]*	*AD[57]*	GND
8	GND	*AD[56]*	*AD[55]*	*AD[54]*	GND	*AD[53]*	GND
9	GND	*AD[52]*	GND	V(I/O)	*AD[51]*	*AD[50]*	GND
10	GND	*AD[49]*	*AD[48]*	*AD[47]*	GND	*AD[46]*	GND
11	GND	*AD[45]*	GND	V(I/O)	*AD[44]*	*AD[43]*	GND
12	GND	*AD[42]*	*AD[41]*	*AD[40]*	GND	*AD[39]*	GND
13	GND	*AD[38]*	GND	V(I/O)	*AD[37]*	*AD[36]*	GND
14	GND	*AD[35]*	*AD[34]*	*AD[33]*	GND	*AD[32]*	GND
15	GND	***PXI_BRSVA15***	GND	FAL#	***PXI_ STAR6***	***PXI_LBR6***	GND
16	GND	**PXI_TRIG1**	**PXI_TRIG0**	DEG#	GND	**PXI_TRIG7**	GND
17	GND	**PXI_TRIG2**	GND	PRST#	***PXI_CLK10_IN***	**PXI_CLK10**	GND
18	GND	**PXI_TRIG3**	**PXI_TRIG4**	**PXI_TRIG5**	GND	**PXI_TRIG6**	GND
19	GND	***PXI_ STAR2***	GND	***PXI_ STAR3***	***PXI_ STAR4***	***PXI_ STAR5***	GND
20	GND	***PXI_LBR4***	***PXI_LBR5***	**PXI_ STAR0**	GND	***PXI_ STAR1***	GND
21	GND	***PXI_LBR0***	GND	***PXI_LBR1***	***PXI_LBR2***	***PXI_LBR3***	GND
22	GND	***PXI_RSVA22***	***PXI_RSVB22***	***PXI_RSVC22***	***PXI_RSVD22***	***PXI_RSVE22***	GND

Figure A-4. PXI/CPCI Slot A P2 (P2A) Pin Configuration

NOTES: Bold-faced words are PXI defined signals.
Italicized words are unused signals.
Underlined words are used by the system slot.

PIN	C	B	A
1	D08	-	D00
2	D09	-	D01
3	D10	-	D02
4	D11	BG0IN*	D03
5	D12	BG0OUT*	D04
6	D13	BG1IN*	D05
7	D14	BG10UT*	D06
8	D15	BG2IN*	D07
9	GND	BG20UT*	GND
10	SYSFAIL*	BG3IN*	-
11	-	BG3OUT*	-
12	SYSRESET*	-	DS1*
13	LWORD*	-	DS0*
14	AM5	-	WRITE*
15	A23	-	-
16	A22	AM0	DTACK*
17	A21	AM1	_
18	A20	AM2	_
19	A19	AM3	_
20	A18	GND	IACK*
21	A17	-	IACKIN*
22	A16	-	IACKOUT*
23	A15	GND	AM4
24	A14	IRQ7*	A07
25	A13	IRQ6*	A06
26	A12	IRQ5*	A05
27	A11	IRQ4*	A04
28	A10	IRQ3*	A03
29	A09	IRQ2*	A02
30	A08	IRQ1*	A01
31	+12 V	-	-12 V
32	+5 V	+5 V	+5 V

Figure A-5. VXI P1 Pin Configuration

PIN	C	B	A
1	-	+5V	-
2	-	GND	-
3	GND	-	-
4	-	A24	GND
5	-	A25	-
6	-	A26	-
7	GND	A27	-
8	-	A28	-
9	-	A29	-
10	GND	A30	GND
11	-	A31	-
12	-	GND	-
13	-	+5V	-
14	-	D16	-
15	-	D17	-
16	GND	D18	GND
17	-	D19	-
18	-	D20	-
19	-	D21	-
20	-	D22	-
21	-	D23	-
22	GND	GND	GND
23	TTLTRG1*	D24	TTLTRG0*
24	TTLTRG3*	D25	TTLTRG2*
25	GND	D26	+5V
26	TTLTRG5*	D27	TTLTRG4*
27	TTLTRG7*	D28	TTLTRG6*
28	GND	D29	GND
29	-	D30	-
30	GND	D31	MODID
31	-	GND	GND
32	-	+5V	-

Figure A-6. VXI P2 Pin Configuration

P3				P4			
PIN	Signal	PIN	Signal	PIN	Signal	PIN	Signal
1	TCK	2	-12V	1	+12V	2	TRST#
3	GND	4	INTA#	3	TMS	4	TDO
5	INTB#	6	INTC#	5	TDI	6	GND
7	BUSMODE1#	8	+5V	7	GND	8	*PCI-RSVD*
9	INTD#	10	*PCI-RSVD*	9	*PCI-RSVD*	10	*PCI-RSVD*
11	GND	12	+3.3Vaux	11	BUSMODE2#	12	+3.3V
13	CLK	14	GND	13	RST#	14	BUSMODE3#
15	GND	16	GNT#	15	+3.3V	16	BUSMODE4#
17	REQ#	18	+5V	17	*PME#*	18	GND
19	V(I/O)	20	AD[31]	19	AD[30]	20	AD[29]
21	AD[28]	22	AD[27]	21	GND	22	AD[26]
23	AD[25]	24	GND	23	AD[24]	24	+3.3V
25	GND	26	C/BE[3]#	25	IDSEL	26	AD[23]
27	AD[22]	28	AD[21]	27	+3.3V	28	AD[20]
29	AD[19]	30	+5V	29	AD[18]	30	GND
31	V(I/O)	32	AD[17]	31	AD[16]	32	C/BE[2]#
33	FRAME#	34	GND	33	GND	34	*PMC-RSVD*
35	GND	36	IRDY#	35	TRDY#	36	+3.3V
37	DEVSEL#	38	+5V	37	GND	38	STOP#
39	GND	40	LOCK#	39	PERR#	40	GND
41	*PCI-RSVD*	42	*PCI-RSVD*	41	+3.3V	42	SERR#
43	PAR	44	GND	43	C/BE[1]#	44	GND
45	V(I/O)	46	AD[15]	45	AD[14]	46	AD[13]
47	AD[12]	48	AD[11]	47	*M66EN*	48	AD[10]
49	AD[09]	50	+5V	49	AD[08]	50	+3.3V
51	GND	52	C/BE[0]#	51	AD[07]	52	*PMC-RSVD*
53	AD[06]	54	AD[05]	53	+3.3V	54	*PMC-RSVD*
55	AD[04]	56	GND	55	*PMC-RSVD*	56	GND
57	V(I/O)	58	AD[03]	57	*PMC-RSVD*	58	*PMC-RSVD*
59	AD[02]	60	AD[01]	59	GND	60	*PMC-RSVD*
61	AD[00]	62	+5V	61	*ACK64#*	62	+3.3V
63	GND	64	*REQ64#*	63	GND	64	*PMC-RSVD*

Note: Italicized words are unused signals.

Figure A-7. PMC Pin Configuration

P6				P5			
PIN	Signal	PIN	Signal	PIN	Signal	PIN	Signal
1	*PCI-RSVD*	2	*GND*	1	I/O	2	I/O
3	*GND*	4	*C/BE[7]#*	3	I/O	4	I/O
5	*C/BE[6]#*	6	*C/BE[5]#*	5	I/O	6	I/O
7	*C/BE[4]#*	8	*GND*	7	I/O	8	I/O
9	*V(I/O)*	10	*PAR64*	9	I/O	10	I/O
11	*AD[63]*	12	*AD[62]*	11	I/O	12	I/O
13	*AD[61]*	14	*GND*	13	I/O	14	I/O
15	*GND*	16	*AD[60]*	15	I/O	16	I/O
17	*AD[59]*	18	*AD[58]*	17	I/O	18	I/O
19	*AD[57]*	20	*GND*	19	I/O	20	I/O
21	*V(I/O)*	22	*AD[56]*	21	I/O	22	I/O
23	*AD[55]*	24	*AD[54]*	23	I/O	24	I/O
25	*AD[53]*	26	*GND*	25	I/O	26	I/O
27	*GND*	28	*AD[52]*	27	I/O	28	I/O
29	*AD[51]*	30	*AD[50]*	29	I/O	30	I/O
31	*AD[49]*	32	*GND*	31	I/O	32	I/O
33	*GND*	34	*AD[48]*	33	I/O	34	I/O
35	*AD[47]*	36	*AD[46]*	35	I/O	36	I/O
37	*AD[45]*	38	*GND*	37	I/O	38	I/O
39	*V(I/O)*	40	*AD[44]*	39	I/O	40	I/O
41	*AD[43]*	42	*AD[42]*	41	I/O	42	I/O
43	*AD[41]*	44	*GND*	43	I/O	44	I/O
45	*GND*	46	*AD[40]*	45	I/O	46	I/O
47	*AD[39]*	48	*AD[38]*	47	I/O	48	I/O
49	*AD[37]*	50	*GND*	49	I/O	50	I/O
51	*GND*	52	*AD[36]*	51	I/O	52	I/O
53	*AD[35]*	54	*AD[34]*	53	I/O	54	I/O
55	*AD[33]*	56	*GND*	55	I/O	56	I/O
57	*V(I/O)*	58	*AD[32]*	57	I/O	58	I/O
59	*PCI-RSVD*	60	*PCI-RSVD*	59	I/O	60	I/O
61	*PCI_RSVD*	62	*GND*	61	I/O	62	I/O
63	*GND*	64	*PCI-RSVD*	63	I/O	64	I/O

Note: Italicized words are unused signals.

Figure A-8. PMC Pin Configuration (continued)

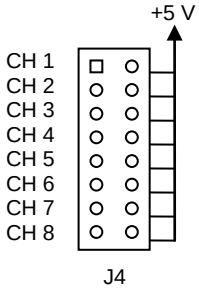


Figure A-9. External Driver Outputs (J4)

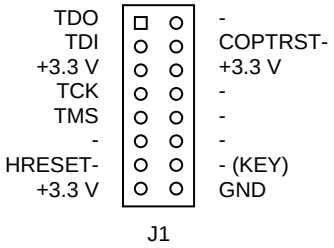


Figure A-10. JTAG/COP Header (J1)

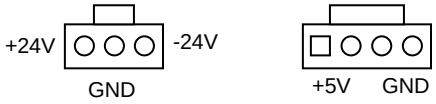


Figure A-11. External Power Connectors

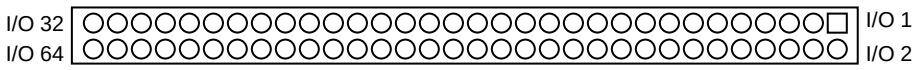


Figure A-12. PMC I/O Connector

This page was left intentionally blank.

APPENDIX B CONFIGURATION REGISTERS

Table B-1. PowerPC Configuration Registers

Device ID		Vendor ID		00
PCI Status		PCI Command		04
Class Code	Subclass Code	Standard Programming	Revision ID	08
BIST Control	Header Type	Latency Timer	Cache Line Size	0C
Local Memory Base Address Register				10
Peripheral Control And Status Register Base Address Register				14
Local Memory Base Address Register 1				18
Subsystem ID		Subsystem Vendor ID		2C
Expansion Rom Base Address				30
MAX LAT	MIN GNT	Interrupt Pin	Interrupt Line	3C
Reserved		Subordinate Bus #	Bus Number	40
PCI Arbiter Control		PCI General Control		44
Output Driver Control	PMCR2	PMCR1		70
Misc Driver Control Reg 2	Misc Driver Control Reg 1	Clock Driver Control Register		74
Embedded Utilities Memory Block Base Address Register				78
Memory Starting Address				80
Memory Starting Address				84
Extended Memory Starting Address				88
Extended Memory Starting Address				8C
Memory Ending Address				90
Memory Ending Address				94
Extended Memory Ending Address				98
Extended Memory Ending Address				9C
Memory Page Mode	Reserved		Memory Bank Enable	A0
Processor Interface Configuration Register 1				A8
Processor Interface Configuration Register 2				AC
Reserved		ECC Single-Bit Trigger	ECC Single-Bit Counter	B8
Proc. Bus Error Status	Reserved	Error Detection 1	Error Enabling 1	C0
PCI Bus Error Status	Reserved	Error Detection 2	Error Enabling 2	C4
Processor/PCI Error Address				C8
Extended ROM Configuration Register 1				D0
Extended ROM Configuration Register 2				D4
Extended ROM Configuration Register 3				D8
Extended ROM Configuration Register 4				DC
Reserved	PLL Configuration	Reserved	Addr. Map B Options	E0
Memory Control Configuration Register 1				F0
Memory Control Configuration Register 2				F4
Memory Control Configuration Register 3				F8
Memory Control Configuration Register 4				FC

B.1 PCI INTERFACE CONFIGURATION REGISTERS

00_{16}	**Device ID (Read)**	**Vendor ID (Read)**
Default	0006_{16}	1057_{16}
Initialized	(read only)	(read only)

04_{16}	**PCI Status**	**PCI Command**
Default	$00A0_{16}$	0004_{16}
Initialized	(bit reset only)	0004_{16}

08_{16}	**Class Code**	**Subclass Code**	**Standard Programming**	**Revision ID**
Default	06_{16}	00_{16}	00_{16}	MPC8245 revision
Initialized	(read only)	(read only)	(read only)	(read only)

$0C_{16}$	**BIST control**	**Header type**	**Latency timer**	**Cache line size**
Default	00_{16}	00_{16}	00_{16}	00_{16}
Initialized	(read only)	(read only)	01_{16}	08_{16}

10_{16}	Local memory Base Address Register 0
Default	0000_0008_{16}
Initialized	0000_0008_{16}

14_{16}	Peripheral control and status register base address register
Default	0000_0000_{16}
Initialized	0000_0000_{16}

18_{16}	Local memory Base Address Register 1 (Read/Write)
Default	0000_0008_{16}
Initialized	0000_0008_{16}

$2C_{16}$	Subsystem ID	Subsystem Vendor ID
Default	0000_{16}	0000_{16}
Initialized	0000_{16}	0000_{16}

30_{16}	Expansion ROM base address
Default	0000_0000_{16}
Initialized	(read only)

$3C_{16}$	MAX LAT	MIN GNT	Interrupt Pin	Interrupt line
Default	00_{16}	00_{16}	01_{16}	00_{16}
Initialized	(read only)	(read only)	(read only)	01_{16}

40_{16}	Reserved	Reserved	Subordinate Bus Number	Bus Number
Default	-	-	00_{16}	00_{16}
Initialized	-	-	00_{16}	00_{16}

44_{16}	PCI arbiter control register	PCI general control register
Default	0000_{16}	0000_{16}
Initialized	$C080_{16}$	0020_{16}

B.2 PERIPHERAL POWER MANAGEMENT CONFIGURATION REGISTERS

70_{16}	Output Dr Cntl	PMCR2	PMCR1
Default	switch configured	switch configured	0000_{16}
Initialized	$B7_{16}$	44_{16}	$C007_{16}$

B.3 OUTPUT/CLOCK DRIVER AND MISC I/O CONTROL REGISTERS

74_{16}	Misc. Dr. Cntl 2	Misc. Dr. Cntl 1	CLK Driver Control register
Default	00_{16}	00_{16}	0000_{16}
Initialized	00_{16}	FC_{16}	0000_{16}

B.4 EMBEDDED UTILITIES MEMORY BLOCK ADDRESS REGISTER

78_{16}	Embedded utilities memory block base address register
Default	0000_0000_{16}
Initialized	8000_0000_{16}

B.5 MEMORY INTERFACE CONFIGURATION REGISTERS

80_{16} Memory Starting Address

	Bank 3	Bank 2	Bank 1	Bank 0
Default	0000_0000_{16}			
Initialized	8080_8000_{16}			

84_{16} Memory Starting Address

	Bank 7	Bank 6	Bank 5	Bank 4
Default	0000_0000_{16}			
Initialized	8080_8080_{16}			

88_{16} Extended Memory Starting Address

	Reserved	Bank 3	Reserved	Bank 2	Reserved	Bank 1	Reserved	Bank 0
Default	0000_0000_{16}							
Initialized	8080_8000_{16}							

$8C_{16}$ Extended Memory Starting Address

	Reserved	Bank 7	Reserved	Bank 6	Reserved	Bank 5	Reserved	Bank 4
Default	0000_0000_{16}							
Initialized	8080_8080_{16}							

90_{16} Memory Ending Address

	Bank 3	Bank 2	Bank 1	Bank 0
Default	0000_0000_{16}			
Initialized	8080_807C_{16}			

94_{16} Memory Ending Address

	Bank 7	Bank 6	Bank 5	Bank 4
Default	0000_0000_{16}			
Initialized	8080_8080_{16}			

98_{16} Extended Memory Ending Address

	Reserved	Bank 3	Reserved	Bank 2	Reserved	Bank 1	Reserved	Bank 0
Default	0000_0000_{16}							
Initialized	8080_8000_{16}							

$9C_{16}$ Extended Memory Ending Address

	Reserved	Bank 7	Reserved	Bank 6	Reserved	Bank 5	Reserved	Bank 4
Default	0000_0000_{16}							
Initialized	8080_8080_{16}							

$A0_{16}$	**Memory Page Mode**	**Reserved**	**Reserved**	**Memory Bank Enable**
Default	00_{16}	-	-	00_{16}
Initialized	00_{16}	-	-	01_{16}

B.6 PROCESSOR INTERFACE CONFIGURATION REGISTERS

$A8_{16}$ Processor interface configuration register 1

Default	$00n4\_0010_{16}$
Initialized	0014_1310_{16}

AC_{16} Processor interface configuration register 2

Default	$000C\_000C_{16}$
Initialized	0000_0000_{16}

B.7 ERROR HANDLING REGISTERS

B8$_{16}$	**Reserved**	**ECC Single-Bit Trigger**	**ECC Single-Bit Counter**
Default	-	00_{16}	00_{16}
Initialized	-	00_{16}	00_{16}

C0$_{16}$	**Proc. Bus Error Status**	**Reserved**	**Error Detection 1**	**Error Enabling 1**
Default	00_{16}	-	00_{16}	01_{16}
Initialized	(bit reset)	-	(bit reset)	00_{16}

C4$_{16}$	**PCI Bus Error Status**	**Reserved**	**Error Detection 2**	**Error Enabling 2**
Default	00_{16}	00_{16}	00_{16}	00_{16}
Initialized	(bit reset)	-	(bit reset)	00_{16}

C8$_{16}$	**Processor/PCI Error Address Register**
Default	0000_0000_{16}
Initialized	(read only)

B.8 EXTENDED ROM CONFIGURATION REGISTERS

D0$_{16}$	**Extended ROM Configuration Register 1**
Default	$B5FF\_8000_{16}$
Initialized	$8C1F\_8180_{16}$

D4$_{16}$	**Extended ROM Configuration Register 2**
Default	$B5FF\_8000_{16}$
Initialized	$84FF\_8000_{16}$

D8$_{16}$	**Extended ROM Configuration Register 3**
Default	$0C00\_000E_{16}$
Initialized	0000_0000_{16}

DC$_{16}$	**Extended ROM Configuration Register 4**
Default	$0C00\_000E_{16}$
Initialized	0000_1000_{16}

B.9 ADDRESS MAP B OPTIONS AND PLL CONFIGURATION REGISTER

E0$_{16}$	**PLL Config Register**	**Reserved**	**Reserved**	**Addr. Map B Options**
Default	config setting	-	-	$C0_{16}$
Initialized	(read only)	-	-	40_{16}

B.10 MEMORY CONTROLCONFIGURATION REGISTER

F0$_{16}$	**Memory Control Configuration Register 1**
Default	$FFn2\_0000_{16}$
Initialized	$0B88\_0002_{16}$

F4$_{16}$	**Memory Control Configuration Register 2**
Default	0000_0000_{16}
Initialized	$A660\_0B3C_{16}$

$F8_{16}$ Memory Control Configuration Register 3

Default	0000_0000_{16}
Initialized	0700_0000_{16}

FC_{16} Memory Control Configuration Register 4

Default	0010_0000_{16}
Initialized	$25B2\_3220_{16}$

This page was left intentionally blank.

APPENDIX C SYSTEM CALLS

Table C-1. System Calls

System Call	Function Code (hex)	System Call	Function Code (hex)
Get Shared Memory Base Address	0001	End of Interrupt	0044
Peek	0002	Set Current Task Priority	0045
Poke	0003	Interrupt Pending	0046
Self Test	0004	Install Interrupt Handler	0047
		Un-Install Interrupt Handler	0048
VXI Send Message	0010		
VXI Send Buffer Status	0011	Firmware Version	0050
Enable/Disable VXI Word Serial	0012		
Install User Command Interpreter	0013	488.2 Set Event Status Register	0100
Enable/Disable Command Interpreter	0014	488.2 Get Event Status Register	0101
Generate VXI Event Interrupt	0015	488.2 Set Event Status Bit	0102
		488.2 Clear Event Status Bit	0103
Read PCI Config. Offset	0020	488.2 Set Event Status Enable Register	0104
Write PCI Config. Offset	0021	488.2 Get Event Status Enable Register	0105
		488.2 Set Event Status Enable Bit	0106
Flash Write	0030	488.2 Clear Event Status Enable Bit	0107
Flash Read	0031	488.2 Set Status Register	0108
Flash Erase Sector	0032	488.2 Get Status Register	0109
Flash Block Write	0033	488.2 Set Status Bit	0110
		488.2 Clear Status Bit	0111
Configure Interrupt	0040	488.2 Set Status Enable Register	0112
Enable Interrupt	0041	488.2 Get Status Enable Register	0113
Disable Interrupt	0042	488.2 Set Status Enable Bit	0114
Acknowledge Interrupt	0043	488.2 Clear Status Enable Bit	0115

Get Shared Memory Base Address

Description: This routine will return the PCI base address of the shared memory device. Refer to Figure 13 for details on the shared memories address space. Note that the base address returned from this function is the base address of the device not just the general purpose shared memory.

Code: 0001_{16}

Parameters: None

Return: r3: Base Address

Peek	
Description:	Performs a read of any location in the PowerPC's address space. This routine uses the PowerPC's reverse byte load instructions so the byte alignment is little endian even though the PowerPC is in big endian mode.
Code:	0002_{16}
Parameters:	r3: Address to perform read r4: Width in bytes 1 = 8 bits 2 = 16 bits 4 = 32 bits
Return:	r3: Data

Poke	
Description:	Performs a write to any location in the PowerPC's address space. This routine uses the PowerPC's reverse byte store instructions so the byte alignment is little endian even though the PowerPC is in big endian mode.
Code:	0003_{16}
Parameters:	r3: Address to perform write r4: Width in bytes 1 = 8 bits 2 = 16 bits 4 = 32 bits r5: Data
Return:	None

Self Test	
Description:	Runs the carriers self test. This test is exactly like the Power-On Self Test (POST) except it does not test SDRAM. SDRAM can only be tested from a hard reset. **Note:** This test will invalidate any values in the operation registers and the shared memory.
Code:	0004h
Parameters:	None
Return:	r3: 0 = Passed 1 = Failed (see the POST Result value in the VXI Control Status register for details)

VXI Send Message

Description: Send a message to the VXI host. This function will put the message in the send buffer and notify the VXI host that a data is ready. It is up to the host to then read the message from the instrument.
Code: 0012_{16}
Parameters: r3: Pointer to VXI message string
Return: r3: Result
0 = Success
-1 = Buffer Full

Get VXI Send Buffer Status

Description: This routine will return the status of the VXI Send Buffer.
Code: 0014_{16}
Parameters: None
Return: r3: Buffer Status
0 = Buffer Empty
1 = Buffer Full
-1 = Error

Enable Disable VXI Word Serial

Description: This routine will enable or disable the VXI Word Serial Protocol Handler. If disabled, the instrument will indicate to the VXI bus that it will not respond to any VXI commands including low-level protocol commands normally invisible to the application.
Code: 0015_{16}
Parameters: r3: Enable (0=Disable, 1=Enable)
Return: None

Install User Command Interpreter

Description: This routine will install a user command parser that is automatically called by the VXI Word Serial Protocol handler whenever a message is received.
Code: 0016_{16}
Parameters: r3: pointer to function of type int (*func)(unsigned char *)
Return: None

Enable/Disable Command Interpreter	
Description:	This routine will enable or disable the command interpreter. If disabled, the carrier will indicate to the VXI host that is will not respond to VXI messages. This includes any messages that are handled by the system command interpreter and the user command interpreter. Low-level VXI commands are not disabled by this function.
Code:	0016_{16}
Parameters:	r3: Enable (0=Disable, 1=Enable)
Return:	None

Generate VXI Event Interrupt	
Description:	This routine will cause the carrier to generate a VXI event interrupt using the specified event value as part of the status/id returned to the controller during the interrupt acknowledge cycle. Any user defined event value between 81_{16} and BF_{16} can be used.
Code:	0017_{16}
Parameters:	r3: Event (81_{16} – BF_{16})
Return:	None

Read PCI Configuration Offset	
Description:	This routine will read the configuration offset of the specified device and return the value.
Code:	0020_{16}
Parameters:	r3: Bus Number (0-255, 0=Primary PCI Bus) r4: Device Number (See Table II) r5: Function Number r6: Offset (0-255) r7: Width in bytes 1 = 8 bits 2 = 16 bits 4 = 32 bits
Return:	r3: Value read from specified offset

Write PCI Configuration Offset	
Description:	This routine will write the configuration offset of the specified device with the specified value.
Code:	0021_{16}
Parameters:	r3: Bus Number (0-255, 0=Primary PCI Bus) r4: Device Number (See Table II) r5: Function Number r6: Offset (0-255) r7: Width in bytes 1 = 8 bits 2 = 16 bits 4 = 32 bits r7: Value to write
Return:	None

Flash Write	
Description:	This routine will write a value to the flash memory. Only single byte writes are supported. This function will return an error if an offset in the system sector is specified. The flash write capability must be enabled as described in section 3.4.2.
Code:	0030_{16}
Parameters:	r3: Offset (10000_{16} – $7FFFF_{16}$) r4: Value (8-bit)
Return:	r3: Result 0 = Success -1 = Offset in system sector specified

Flash Read	
Description:	This routine will read the flash device at the specified offset.
Code:	0031_{16}
Parameters:	r3: Offset (00000_{16} – $7FFFF_{16}$)
Return:	r3: Data (8-bit)

Flash Erase Sector	
Description:	This routine will erase the specified flash sector. All data in the sector will be lost. Sector 0, the system sector, cannot be erased by this function. This function will return an error if sector 0 is specified. The flash write capability must be enabled as described in section 3.4.2.
Code:	0032_{16}
Parameters:	r3: Sector (1-127)
Return:	r3: Result 0 = Success -1 = System sector specified

Flash Block Write	
Description:	This routine will write a block of data to the flash device. This function will return an error if an offset in the system sector is specified. The flash write capability must be enabled as described in section 3.4.2.
Code:	0033_{16}
Parameters:	r3: Offset (10000_{16} – $7FFFF_{16}$) r4: Data pointer (any memory location in PowerPC address space) r5: Number of bytes
Return:	r3: Result 0 = Success -1 = System sector specified

Configure Interrupt	
Description:	Configure the EPIC controller to handle a PCI interrupt. This function will configure but not enable the interrupt.
Code:	0040_{16}
Parameters:	r3: IRQ # (0-4) (See Table III) r4: Interrupt vector to return during an interrupt acknowledge command r5: Interrupt priority (0-F_{16})
Return:	None

Enable Interrupt	
Description:	Enable the specified interrupt.
Code:	0041_{16}
Parameters:	r3: IRQ # (0-4) (See Table III)
Return:	None

Disable Interrupt	
Description:	Disable the specified interrupt.
Code:	0042_{16}
Parameters:	r3: IRQ # (0-4) (See Table III)
Return:	None

Acknowledge Interrupt	
Description:	Perform an interrupt acknowledge cycle on the EPIC controller.
Code:	0043_{16}
Parameters:	None
Return:	r3: Interrupt vector of the pending interrupt

End of Interrupt	
Description:	Perform an end of interrupt cycle on the EPIC controller. This is required to clear the interrupt pending to the processor. If the interrupt condition is no longer valid the interrupt should be cleared. Otherwise another interrupt will immediately occur
Code:	0044_{16}
Parameters:	None
Return:	None

Set current task priority	
Description:	This will set the EPIC priority register to the specified value. If an interrupt occurs of lesser priority, the processor will not be notified until the current task priority is lowered. This function should be called at the beginning and end of an interrupt service routine to block lower priority interrupts from occurring.
Code:	0045_{16}
Parameters:	r3: Priority Level ($0\text{-}F_{16}$)
Return:	None

Interrupt Pending	
Description:	Determines if the specified interrupt is pending or not.
Code:	0046_{16}
Parameters:	r3: IRQ # (0-4) (See Table III)
Return:	r3: 1 = pending 0 = not pending

Install Interrupt Handler	
Description:	Installs an interrupt handler to be called whenever the processor is interrupted by the EPIC controller. All PCI interrupts will result in the same interrupt handler being called.
Code:	0047_{16}
Parameters:	r3: pointer to a function of type void (*func)(void)
Return:	None 0 = not pending

Un-Install Interrupt Handler	
Description:	Un-installs the interrupt handler for EPIC interrupts. All interrupts should be cleared and disabled prior to calling this routine
Code:	0048_{16}
Parameters:	r3: pointer to a function of type void (*func)(void)
Return:	None

Firmware Version	
Description:	Returns the current version of the system firmware.
Code:	0050_{16}
Parameters:	r3: address of buffer where the string will be copied to
Return:	None

488.2 Set Event Status Register	
Description:	Sets the 488.2 Event Status Register to the specified value.
Code:	0100_{16}
Parameters:	r3: value
Return:	None

488.2 Get Event Status Register	
Description:	Returns the current value of the 448.2 Event Status Register.
Code:	0101_{16}
Parameters:	None
Return:	r3: value

488.2 Set Event Status Bit

Description: Sets a single bit in the 488.2 Event Status Register.
Code: 0102_{16}
Parameters: r3: bit number (0-7)
Return: None

488.2 Clear Event Status Bit

Description: Clears a single bit in the 488.2 Event Status Register.
Code: 0103_{16}
Parameters: r3: bit number (0-7)
Return: None

488.2 Set Event Status Enable Register

Description: Sets the 488.2 Event Status Enable Register to the specified value.
Code: 0104_{16}
Parameters: r3: value
Return: None

488.2 Get Event Status Enable Register

Description: Returns the current value of the 448.2 Event Status Enable Register.
Code: 0105_{16}
Parameters: None
Return: r3: value

488.2 Set Event Status Enable Bit

Description: Sets a single bit in the 488.2 Event Status Enable Register.
Code: 0106_{16}
Parameters: r3: bit number (0-7)
Return: None

488.2 Clear Event Status Enable Bit

Description: Clears a single bit in the 488.2 Event Status Enable Register.
Code: 0107_{16}
Parameters: r3: bit number (0-7)
Return: None

488.2 Set Status Byte Register

Description: Sets the 488.2 Status Byte Register to the specified value.
Code: 0108_{16}
Parameters: r3: value
Return: None

488.2 Get Status Byte Register

Description: Returns the current value of the 448.2 Status Byte Register.
Code: 0109_{16}
Parameters: None
Return: r3: value

488.2 Set Status Bit

Description: Sets a single bit in the 488.2 Status Byte Register.
Code: 0110_{16}
Parameters: r3: bit number (0-7)
Return: None

488.2 Clear Status Bit

Description: Clears a single bit in the 488.2 Status Byte Register.
Code: 0111_{16}
Parameters: r3: bit number (0-7)
Return: None

488.2 Set Status Byte Enable Register

Description: Sets the 488.2 Status Byte Enable Register to the specified value.
Code: 0112_{16}
Parameters: r3: value
Return: None

488.2 Get Status Byte Enable Register

Description: Returns the current value of the 448.2 Status Byte Enable Register.
Code: 0113_{16}
Parameters: None
Return: r3: value

488.2 Set Status Byte Enable Bit	
Description:	Sets a single bit in the 488.2 Status Byte Enable Register.
Code:	0114_{16}
Parameters:	r3: bit number (0-7)
Return:	None

488.2 Clear Status Byte Enable Bit	
Description:	Clears a single bit in the 488.2 Status Byte Enable Register.
Code:	0115_{16}
Parameters:	r3: bit number (0-7)
Return:	None

This page was left intentionally blank.

APPENDIX D SYSTEM COMMANDS

Table D-1. System Commands

Command	Parameters
SYSTem	
:PEEK?	<address>, <width>
:POKE	<address>, <width>, <data>
:FLASH	
:WRITE	<offset>, <data>
:READ?	<offset>
:ERASE	<sector>
:BLOCK	<offset>,<data ptr>,<num bytes>
:PCI	
:CONFIGure	
:READ?	<bus>, <device>, <func>, <width>, <offset>
:WRITE	<bus>, <device>, <func>, <width>, <offset>, <data>
:SCAN?	
:CONFIGure	
:BOOT	
:TYPE[?]	<type>
:ADDR[?]	<address>
:VXI	
:TYPE[?]	<type>
:MANF[?]	<manufacturer id>
:MODEL[?]	<model code>
:WIDTH[?]	<A24/A32 width>
:DEFAULT	
:VXI	
:WIDTH[?]	<width>
:VER?	
:DOWNLOAD	
:ERR?	

Table D-2. IEEE 488.2 Common Commands

Command	Description
*IDN?	Identification query
*RST	Reset
*TST?	Self test query
*OPC	Operation complete
*OPC?	Operation complete query
*WAI	Wait to continue
*CLS	Clear status
*ESE	Event status enable register
*ESE?	Event status enable register query
*ESR?	Event status register query
*SRE	Service request enable register
*SRE?	Service request enable register query
*STB?	Status byte register query

:PEEK?		
Description:	Performs a read of the specified width to any address in the PowerPC's address map. This routine uses the PowerPC's reverse byte load instructions so the byte alignment is little endian even though the PowerPC is in big endian mode.	
Parameters:	<address>	any location in the PowerPCs address space; must be a 32-bit hex value; must be aligned for the specified width (i.e. if width is 8-bit the address can be any value, if 16-bit the address must be a multiple of 2, and if 32-bit the address must be a multiple of 4)
	<width>	Access width in bytes: 1 = 8-bit 2 = 16-bit 4 = 32-bit

:POKE		
Description:	Performs a write of the specified width to any address in the PowerPC's address map. This routine uses the PowerPC's reverse byte load instructions so the byte alignment is little endian even though the PowerPC is in big endian mode.	
Parameters:	<address>	any location in the PowerPCs address space; must be a 32-bit hex value; must be aligned for the specified width (i.e. if width is 8-bit the address can be any value, if 16-bit the address must be a multiple of 2, and if 32-bit the address must be a multiple of 4)
	<width>	Access width in bytes: 1 = 8-bit 2 = 16-bit 4 = 32-bit
	<data>	a hex value of the same width as specified

:FLASH:WRITE		
Description:	Programs the flash memory device at the specified address, with the specified data. The flash write capability must be enabled as described in section 3.4.2. Offsets 00_0000_{16} – 00_FFFF_{16} are reserved for system use. If an offset within this range is specified the command will return an error. The write command can only toggle a bit from a '1' to a '0'. To set a bit back to a '1' an erase operation must be performed.	
Parameters:	<offset>	offset in the flash memory device to which the data is written; must be a 24-bit hex value between 01_0000_{16} and $7F\_FFFF_{16}$.
	<data>	8-bit hex value

:FLASH:READ?		
Description:	Reads the flash memory device at the specified offset.	
Parameters:	<offset>	offset in the flash memory device from which the data is read; must be a 24-bit hex value between 00_0000_{16} and $7F\_FFFF_{16}$.

:FLASH:ERASE

Description: Erases the specified sector in the flash memory device. The erase operation sets all bits in the sector to a '1'. Sectors are 64 kilobytes and are organized sequentially (i.e. offsets 00_0000_{16} – 00_FFFF_{16} are in sector 0, offsets 01_0000_{16} – 01_FFFF_{16} are in sector 1 and so forth). Sector 0 is reserved for system use. If sector 0 is specified the command will return an error. The flash write capability must be enabled as described in section 3.4.2.

Parameters: <sector> 1 - 127

:FLASH:BLOCK

Description: Programs the flash device with a block of data read from the shared memory device. The flash write capability must be enabled as described in section 3.4.2. Offsets 00_0000_{16} – 00_FFFF_{16} are reserved for system use. If an offset within this range is specified the command will return an error. The write command can only toggle a bit from a '1' to a '0'. To set a bit back to a '1' an erase operation must be performed.

Parameters:

<offset>	start offset in the flash memory device to which the data is written; must be a 24-bit hex value between 01_0000_{16} and $7F\_FFFF_{16}$.
<data ptr>	shared memory address of the beginning of the data block
<num bytes>	num of bytes to write to flash

:PCI:CONFIG:READ?

Description: Performs a PCI configuration read of the specified offset of the specified device.

Parameters:

<bus>	bus number (0-255, 0 = Primary PCI bus)
<device>	device number (See Table II)
<func>	function number (0 for non multi-function devices)
<width>	Access width in bytes: 1 = 8-bit 2 = 16-bit 4 = 32-bit
<offset>	configuration register offset; must be a 8-bit hex value

:PCI:CONFIG:WRITE		
Description:	Performs a PCI configuration write to the specified offset of the specified device.	
Parameters:	<bus>	bus number (0-255, 0 = Primary PCI bus)
	<device>	device number (See Table II)
	<func>	function number (0 for non multi-function devices)
	<width>	Access width in bytes: 1 = 8-bit 2 = 16-bit 4 = 32-bit
	<offset>	configuration register offset; must be a 8-bit hex value;
	<data>	32-bit hex value

:PCI:SCAN?	
Description:	Returns a string description of the entire PCI bus architecture including any existing secondary buses.
Parameters:	None
Return:	String format is as follows: "bus_num:device_num(func_num); bus_num:device_num(func)num;…"

:CONFIG:BOOT:TYPE[?]		
Description:	Sets or queries the boot type configuration option. This setting will be stored in flash memory and will remain valid even after power is removed from the carrier. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability	
Parameters:	<type>	0 = normal (default)
or Returns		1 = download

:CONFIG:BOOT:ADDR[?]		
Description:	Sets or queries the boot address configuration option. This setting will be stored in flash memory and will remain valid even after power is removed from the carrier. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability	
Parameters:	<address>	any PowerPC addressable location; must be a 32-bit hex value
or Returns		

:CONFIG:VXI:TYPE[?]

Description: Sets or queries the VXI type configuration option. This setting will be stored in flash memory and will remain valid even after power is removed from the carrier. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability

Parameters: <type> 0 = message based (default)
or Returns 1 = register based

:CONFIG:VXI:MANF[?]

Description: Sets or queries the VXI Manufacturer ID configuration option. This setting will be stored in flash memory and will remain valid even after power is removed from the carrier. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability

Parameters: <manf id> any 12-bit hex value (default = $FC1_{16}$)
or Returns

:CONFIG:VXI:MODEL[?]

Description: Sets or queries the VXI Model Code configuration option. This setting will be stored in flash memory and will remain valid even after power is removed from the carrier. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability

Parameters: <model code> any 12-bit hex value (default = $FE4_{16}$)
or Returns

:CONFIG:VXI:WIDTH[?]

Description: Sets or queries the VXI A24/A32 Width configuration option. This setting will be stored in flash memory and will remain valid even after power is removed from the carrier. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability

Parameters: <width> 2 = 16-bit (default)
or Returns 4 = 32-bit

:CONFIG:DEFAULT	
Description:	Returns all the configuration options to their factory defaults. Refer to section 5.2.2 for details on the configuration options. Flash write capability must be enabled for the configuration to be saved. Refer to section 3.4.2 for details on enabling or disabling the flash write capability
Parameters:	None
Return:	None

:VXI:WIDTH[?]	
Description:	Sets or queries the width of the VXI A24/A32 address space. Refer to section 4.3.2 for details on the VXI data bus width. This setting remains valid until this command or the "SYS:CONFIG:VXI:WIDTH" command is performed again. This command is different from the "SYS:CONFIG:VXI:WIDTH" command in that it does not store the setting in non-volatile memory.
Parameters:	<width> 2 = 16-bit (default)
or Returns	4 = 32-bit

:VER?	
Description:	Returns the firmware version of the on-board system utilities
Parameters:	None
Returns:	String description of the firmware version

:DOWNLOAD	
Description:	Starts the firmware download mode which allows the user to download code or data to the VX407C. This command cannot be used to update the system sector of the flash. System firmware update mode can only be entered at power-up by setting the Update System Firmware configuration switch.
Parameters:	None
Return:	None

:ERR?	
Description:	Returns the next message from the error message queue. Messages are returned in first in first out order.
Parameters:	None
Return:	Error code and error description

***IDN**

Description: Returns identification information for the carrier.

Parameters: None

Return: ID string in the following format:
- "C&H Technologies, VX407C, 0, firmware rev"

***RST**

Description: Resets the carrier to its default state. This is different from a hard reset in that the PowerPC is not reset. The effect of the *RST command are outlined below.

Effects

- Un-map all triggers
- Clear all pending interrupts
- Clear operations registers
- Clear all *OPC, *OPC? and *WAI requests
- Reset PXI/cPCI and PMC modules
- Clear all external relay drivers
- Reset the shared memory device
- Enumerate PCI bus

Does not effect

- Self test is not run
- Status and status enable Registers
- VXI interface
- EPIC controller
- PowerPC configuration registers
- User Application

Parameters: None

Return: None

***TST?**

Description: Runs the carriers self test and returns either Pass or Fail.

Note: This action will invalidate any values in the operations registers and the shared memory.

Parameters: None

Returns: 0 = Passed
1 = Failed (see the POST Result value in the VXI Control Status register for details)

***OPC**	
Description:	Instructs the carrier to set the operation complete bit in the event status register when all pending operations are complete. **Note:** All defined system commands execute in a sequential order. Therefore this command is only effective if a user application exists that defines non-sequential commands. Otherwise the operation complete bit will be set immediately.
Parameters:	None
Return:	Error code and error description

***OPC?**	
Description:	Instructs the carrier to return an ASCII '1' via the VXI word serial interface when all pending operations are complete. **Note:** All defined system commands execute in a sequential order. Therefore this command is only effective if a user application exists that defines non-sequential commands. Otherwise a '1' will be returned immediately.
Parameters:	None
Return:	Error code and error description

***WAI**	
Description:	This command will stop the carrier from receiving VXI commands until all pending operations are complete. **Note:** All defined system commands execute in a sequential order. Therefore this command is only effective if a user application exists that defines non-sequential commands.
Parameters:	None
Returns:	Error code and error description

***CLS**

Description: Clears the 488.2 status and reporting registers and flags.

Clears

- Status Byte Register
- Event Status Register
- Error Queue
- WAI flag
- OPC flag

Does not clear

- Status Byte Enable Register
- Event Status Enable Register

Parameters: None
Return: Error code and error description

***ESE[?]**

Description: Sets or Queries the 488.2 Event Status Enable Register
Parameters or Returns: <value> (8-bit decimal value)

***ESR?**

Description: Queries the 488.2 Event Status Register
Parameters: None
Returns: <value> (8-bit decimal value)

***SRE[?]**

Description: Sets or Queries the 488.2 Status Byte Enable Register
Parameters or Returns: <value> (8-bit decimal value)

***STB?**

Description: Queries the 488.2 Status Byte Register
Parameters: None
Returns: <value> (8-bit decimal value)