GPIB-VXI

User Manual

April 1990 Edition

Part Number 320151-01

National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

(512) 794-0100
(800) IEEE-488
Fax: (512) 794-8411

Limited Warranty

The GPIB-VXI is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

FCC/DOC Radio Frequency Interference Compliance

This equipment generates and uses radio frequency energy and, if not installed and used in strict accordance with the instructions in this manual, may cause interference to radio and television reception. This equipment has been tested and found to comply with (1) the limits for a Class A computing device, in accordance with the specifications in Subpart J of Part 15 of U.S. Federal Communications Commission (FCC) Rules, and (2) the limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communication (DOC). These regulations are designed to provide reasonable protection against interference from the equipment to radio and television reception in commercial areas.

There is no guarantee that interference will not occur in a particular installation. However, the chances of interference are much less if the equipment is used according to this instruction manual.

If the equipment does cause interference to radio or television reception, which can be determined by turning the equipment on and off, one or more of the following suggestions may reduce or eliminate the problem.

- Operate the equipment and the receiver on different branches of your AC electrical system.
- Move the equipment away from the receiver with which it is interfering.
- Relocate the equipment with respect to the receiver.
- Reorient the receiver's antenna.
- Be sure that the equipment is plugged into a grounded outlet and that the grounding has not been defeated with a cheater plug.

If necessary, consult National Instruments or an experienced radio/television technician for additional suggestions. The following booklet prepared by the FCC may also be helpful: *How to Identify and Resolve Radio-TV Interference Problems.* This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock Number 004-000-00345-4.

Preface

This manual contains information you will need to use the GPIB-VXI in your VXIbus system. It describes the function and behavior of GPIB-VXI units configured with the standard firmware option.

Organization of the *GPIB-VXI User Manual*

The *GPIB-VXI User Manual* is organized as follows:

- Chapter 1, *General Description*, gives an overview of the GPIB-VXI.
- Chapter 2, *Configuration and Startup Procedures*, gives configuration information and describes the GPIB-VXI startup behavior.
- Chapter 3, *Local Command Set*, describes the GPIB-VXI local command set.
- Chapter 4, *Nonvolatile Configuration*, describes the method for editing the contents of the GPIB-VXI configuration parameter memory.
- Chapter 5, *Diagnostic Tests*, describes the operation of the GPIB-VXI offline diagnostic tests.
- Appendix A, *Specifications*, lists the specifications of the GPIB-VXI.
- Appendix B, *Error Codes*, lists the local command set error codes.
- Appendix C, *Code Instrument Overview*, describes the capabilities and implementation of Code Instruments.
- Appendix D, *Using the CDS-852 Adapter Code Instrument*, contains instructions for installing the Resident Code Instruments.
- Appendix E, *GPIB-VXI Hardware and Software Configuration Form*, contains a form that you should complete in the event that you have a technical problem. Completing the form before calling National Instruments will expedite your phone call and thus the solution to your problem.
- The *Glossary* contains an alphabetical list of terms used in this manual and a description of each.
- The *Index* contains an alphabetical list of key terms and topics used in this manual, including the page where each one can be found.

Conventions Used in This Manual

Throughout this manual, the following conventions are used to distinguish elements of text:

italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept. In this manual, italics are also used to denote Word Serial commands and queries.
`monospace`	Text in this font denotes text or characters that are to be literally input from the keyboard, sections of code, command or query syntax, console responses, and syntax examples. This font is also used for the names of all commands and queries used in the GPIB-VXI local command set.
<CR>	Angle brackets enclosing a term in Times font denote a key on the keyboard, or the equivalent ASCII character.
`<hex value>`	Angle brackets enclosing a term in monospace denote a parameter.

Numbers in this manual are base 10 unless noted as follows:

- Binary numbers are indicated by a -b suffix (for example, 11010101b)
- Octal numbers are indicated by an -o suffix (for example, 325o),
- Hexadecimal numbers are indicated by an -h suffix (for example, D5h)
- ASCII character and string values are indicated by double quotation marks (for example, "This is a string").

In this manual, the symbol <CR> is used to indicate the ASCII carriage return character. The symbol <LF> is used to indicate the ASCII linefeed character. The symbol <CRLF> is used to indicate a carriage return followed by a linefeed.

Terminology that is specific to a chapter or section is defined at its first occurrence.

Abbreviations

The following abbreviations are units of measure that are used in the text of this manual.

°	degrees
A	ampere
bytes/sec	bytes per second
C	Celsius
Hz	hertz
in.	inch
kbytes/sec	1,000 bytes per second
kHz	kilohertz

K	1,024 bytes of memory
LSB	least significant bit
m	meter
mA	milliampere
M	1,048,576 bytes of memory
MHz	megahertz
MSB	most significant bit
nsec	nanosecond
sec	second
VDC	volts direct current

Related Documents

The following documents contain information that you may find helpful as you read this manual:

- *IEEE Standard for a Versatile Backplane Bus: VMEbus*, ANSI/IEEE Standard 1014-1987
- *IEEE Standard Digital Interface for Programmable Instrumentation*, ANSI/IEEE Standard 488.1-1987
- *IEEE Standard Codes, Formats, Protocols, and Common Commands*, ANSI/IEEE Standard 488.2-1987
- *VXIbus System Specification*, Revision 1.3, VXIbus Consortium
- *16/32-Bit Highly Integrated Microprocessor SCC68070 User Manual*, Philips

Customer Communication

We appreciate communicating with the people who use our products. We are also very interested in hearing about the applications you develop using our products. To make it easy for you to communicate with us, we provide the *Hardware and Software Configuration Form* for product-related technical comments, and the *User Comment Form* for documentation comments.

If you encounter any technical problems, please complete the *Hardware and Software Configuration Form* in Appendix E and call National Instruments Corporation. Completing the form before calling National Instruments will help solve your problem faster.

You can use the following toll-free number between the hours of 8:00 a.m. and 5:30 p.m. (central time) to reach the National Instruments applications engineering department:

(512) 794-0100
(800) 433-3488 (toll-free U.S. and Canada)

For your documentation comments, we have included a *User Comment Form* at the back of the manual. Please mail it to the address printed at the bottom of the form.

Contents

Figures

Tables

Chapter 1
General Description

This chapter contains a brief overview of the GPIB-VXI and its VXIbus, VMEbus and GPIB capabilities. This chapter also contains a description of the local command set, an introduction to Code Instruments (CIs), and a description of the GPIB-VXI front panel indicators, switches and connectors.

Overview

The GPIB-VXI is a C-sized interface module that links the industry standard IEEE-488 (GPIB) bus and the VXIbus. The GPIB-VXI performs transparent conversion of the GPIB signals and protocols to VXIbus signals and protocols, so that a GPIB Controller can control VXIbus instruments in the same way that it controls GPIB instruments. Figure 1-1 shows the GPIB-VXI interface module.

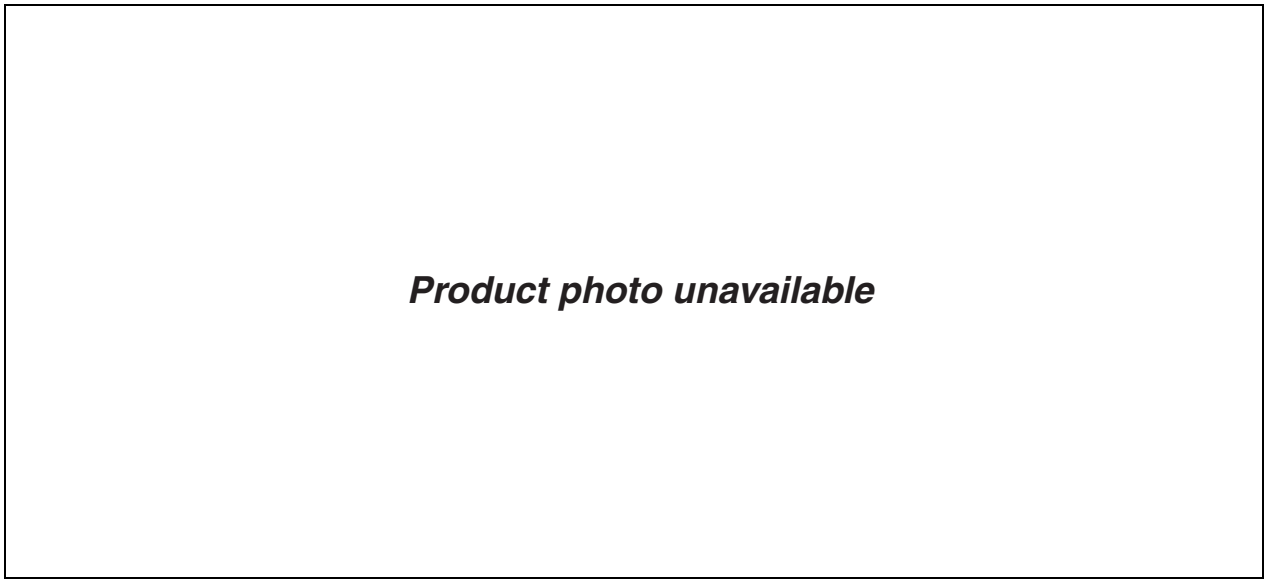


Figure 1-1. The GPIB-VXI Interface Module

The GPIB-VXI is factory configured as the system Resource Manager (RM). It performs the VXIbus startup configuration, self-test, and initialization functions, as well as VXIbus Slot 0/ VMEbus Slot 1-related services.

The RM and Slot 0 functions can be defeated individually, so that the GPIB-VXI can coexist with another RM and/or be located in any slot.

What Your Kit Should Contain

Your GPIB-VXI kit should contain the following components:

Kit Component	Part Number
One GPIB-VXI module	180715-*XYZ*
One *GPIB-VXI User Manual*	320151-01

The GPIB-VXI part number and serial number are printed on the label affixed to its shield casing.

If your kit is missing any of the listed items or if you received the wrong version, contact National Instruments.

Note: The full part number of the GPIB-VXI is determined by configuration options corresponding to the extension *-XYZ* in the part number shown in the previous table. The options are described below.

X 68881 coprocessor option
- 0 without coprocessor
- 1 with coprocessor

Y ROM option
- 1 standard firmware
- 2 development firmware

Z RAM option
- 1 512K
- 2 1M
- 3 2M
- 4 4M

Optional Equipment

Item	Part Number
Serial Port Cables:	
GPIB-VXI to IBM-PC 9-pin - 2 m	181139-02
GPIB-VXI to 25-pin DTE terminal - 2 m	181138-02
Double-Shielded GPIB Cables:	
Type X2 Cable - 1 m	763061-01
Type X2 Cable - 2 m	763061-02
Type X2 Cable - 4 m	763061-03

Unpacking

Follow these steps when unpacking your GPIB-VXI:

1. Verify that the pieces contained in the package you received match the kit parts list. Do not remove the module from its plastic bag at this point.

2. Your GPIB-VXI module is shipped packaged in an antistatic plastic bag to prevent electrostatic damage to the module. Several components on the module can be damaged by electrostatic discharge. To avoid such damage while handling the module, touch the plastic bag to a metal part of your VXIbus mainframe chassis before removing the module from the bag.

3. Remove the module from the bag and inspect the module for loose components or any other sign of damage. Notify National Instruments if the module appears damaged in any way. *Do not* install a damaged module into your VXIbus mainframe.

VXIbus and VMEbus Capabilities

The GPIB-VXI has the following VXIbus and VMEbus capabilities:

- Fully compatible with VXIbus Specification Revisions 1.2 and 1.3
- VXIbus Resource Manager (RM) (defeatable)
- VXIbus Slot 0/VMEbus Slot 1 support (defeatable)
- VXIbus Message-Based commander and Message-Based servant
- VMEbus master and slave

- Up to 4M of dual-ported (shared) memory
- Three programmable VXIbus interrupt handlers
- IEEE 488.1 and IEEE 488.2-compatible 488-VXIbus translator

GPIB Characteristics

The GPIB-VXI has the following GPIB characteristics:

- Communication with VXIbus Message-Based devices
  - VXI logical addresses are mapped to GPIB secondary addresses
  - Automatically configured at startup
  - Programmable
- Interface
  - NEC 7210 and National Instruments Turbo488 ASIC
  - Full, transparent support of individual status bytes for each secondary address
  - Buffered operation decouples GPIB and VXIbus operation
  - Controller can address one VXIbus device to talk and other VXIbus devices to listen
- IEEE 488.1 capabilities
  - SH1 (Source Handshake)
  - AH1 (Acceptor Handshake)
  - TE5 (Extended Talker)
  - LE3 (Extended Listener)
  - SR1 (Service Request)
  - DC1 (Device Clear)
  - DT1 (Device Trigger)
  - RL0 (Remote Local)
- IEEE 488.2-compatible 488-VXIbus translation

The IEEE 488.1 capabilities are supported for all VXIbus devices associated with GPIB secondary addresses. The IEEE 488.2 compatibility applies to 488.2-compatible VXIbus devices associated with GPIB secondary addresses through the GPIB-VXI.

Command Set

The GPIB-VXI local command set supports the following types of operations:

- System configuration and control
  - Help
  - General configuration
  - RM information extraction
  - VXI-defined common ASCII system commands
  - Dynamic system configuration and reconfiguration
  - GPIB address configuration
  - VXIbus interrupt handler configuration
  - IEEE 488.2 common commands
- Instrument development and test
  - VXIbus access
  - Word Serial communication
- CI use and development
  - CI configuration

You can access the command set from the GPIB port, the serial port, and through Word Serial Protocol communication. You can also use separate programmable local command response modes for interactive and control program operation.

Code Instruments

The GPIB-VXI can run software modules called *Code Instruments* or *CIs* that perform special functions in the VXIbus environment. Typical applications of CIs include:

- Command language translation and interpretation
- Virtual (hierarchical) instrument creation
- Message-Based interface creation for Register-Based devices
- Message-Based interface for non-VXI devices

CIs can be implemented in three forms:

- As part of the National Instruments-supplied firmware (*Resident CIs*, or *RCIs*)
- As user-developed downloadable object code (*Downloaded CIs*, or *DCIs*)
- As user-add-on firmware (*EPROMed CIs*, or *ECIs*)

For more information about CI capabilities and applications, see Appendix C, *Code Instrument Overview*.

Front Panel Indicators, Switches and Connectors

The GPIB-VXI has the following front panel features:

- Five front panel LEDs
  - FAILED, TEST, and ONLINE LEDs indicate the self-test status of the GPIB-VXI.
  - ACCESS LED indicates when the GPIB-VXI is accessed from GPIB or VXIbus.
  - SYSFAIL LED indicates when any VXIbus device in the system fails.
- Five front panel connectors
  - GPIB interface
  - Serial port
  - Trigger input and output (2 BNCs)
  - External CLK10 I/O (BNC)
- System reset switch

Power Consumption and Temperature Rating

Current requirements and the temperature rating of the GPIB-VXI are printed on a label affixed to its shield casing.

Chapter 2
Configuration and Startup Procedures

This chapter contains information about the system configuration, GPIB-VXI configuration, and startup operation.

System Configuration

The typical system includes the following components:

- A VXIbus system mainframe containing the GPIB-VXI and the target instrument modules
- A host computer with a GPIB interface module and associated driver software (available for many computers from National Instruments) connected to the GPIB-VXI GPIB port
- A dumb terminal or host running a terminal emulator connected to the GPIB-VXI serial port (optional)

The 9-pin serial port connector pinouts are listed in Table 2-1.

Table 2-1. Serial Port Connector RS-232 Pinouts

Pin	Signal	GPIB-VXI I/O
2	Receive Data	Input
3	Transmit Data	Output
4	Data Terminal Ready	Output
5	Signal Ground	
7	Ready to Send	Output
8	Clear to Send	Input

A three-wire connection to pins 2, 3, and 5 works well for most terminals and emulators. No connections to pins 4, 7, or 8 are necessary. The serial port settings are 9600 baud, 8-bit data, no parity, and one stop bit.

Warning: Do not make connections to pins 1 and 6. This could damage your GPIB-VXI.

Cables for connecting the GPIB-VXI serial port to an RS-232 terminal or COM1 port on an IBM PC-compatible computer are available from National Instruments (see *Optional Equipment* in Chapter 1).

GPIB-VXI Configuration

The GPIB-VXI factory configuration is shown in Table 2-2. The RAM, firmware and coprocessor are configured according to the GPIB-VXI purchase options.

Table 2-2. GPIB-VXI Factory Configuration

Function	**Factory Configuration**
Startup Mode	VXIbus system
VXIbus characteristics Resource Manager (RM) Logical Address Servant area size Dual-ported memory	 Enabled 0 0 0% of installed memory
VXIbus Slot 0 services CLK10 driver CLK10 source	 Enabled Onboard clock
VMEbus Slot 1 services SYSCLK driver Priority Arbiter	 Enabled Enabled
VMEbus requester	Level 3
VXI Interrupt handlers	Unassigned
GPIB primary address	1
Serial Port System startup messages	 Disabled

You do not have to change the GPIB-VXI factory configuration to use it as a Slot 0 Resource Manager. This section is a guide to alternate configurations.

The location of the GPIB-VXI switches and jumpers is illustrated in Figure 2-1. The figures in this section are illustrated according to the orientation of the GPIB-VXI as depicted in Figure 2-1.

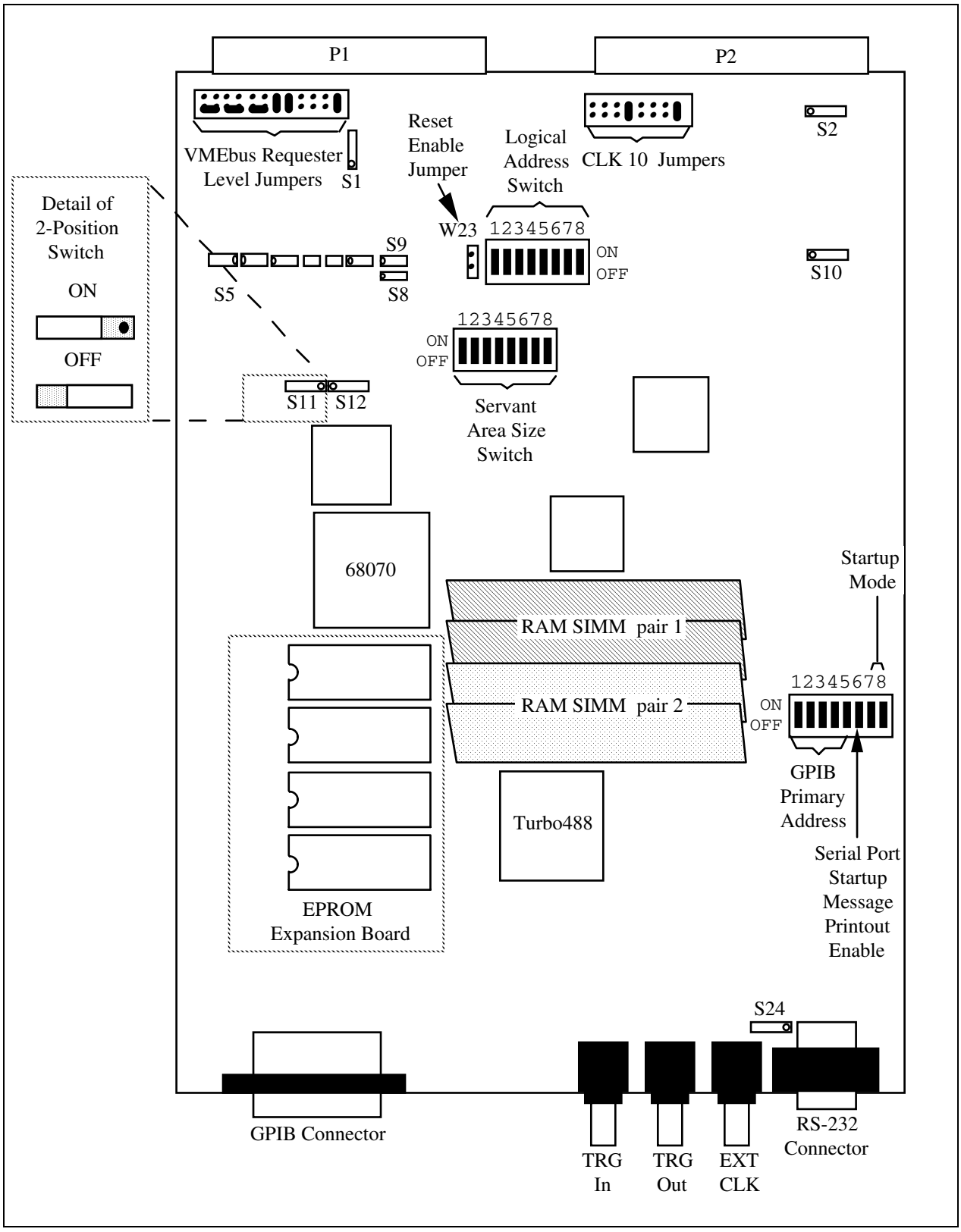


Figure 2-1. GPIB-VXI Parts Locator Diagram

Setting the Logical Address

The logical address switches 8 through 1 correspond to the GPIB-VXI's logical address bits 7 through 0, respectively. The *ON* position corresponds to a bit value of 0, and *OFF* corresponds to a value of 1. For example, to set the logical address of the GPIB-VXI to 25 (19h), set the switches as shown in Figure 2-2. Notice that setting the logical address to any setting but 0 will disable the GPIB-VXI RM.

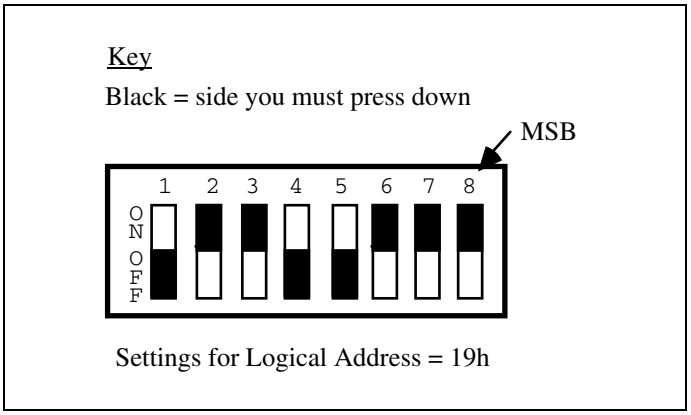


Figure 2-2. Example Logical Address Switch Setting

Setting the GPIB Primary Address

The GPIB switches 5 through 1 correspond to GPIB primary address bits 4 through 0, respectively. The *ON* position corresponds to a bit value of 0, and *OFF* corresponds to a value of 1. To set the primary address of the GPIB-VXI to 3 (03h), for example, set the switches as shown in Figure 2-3.

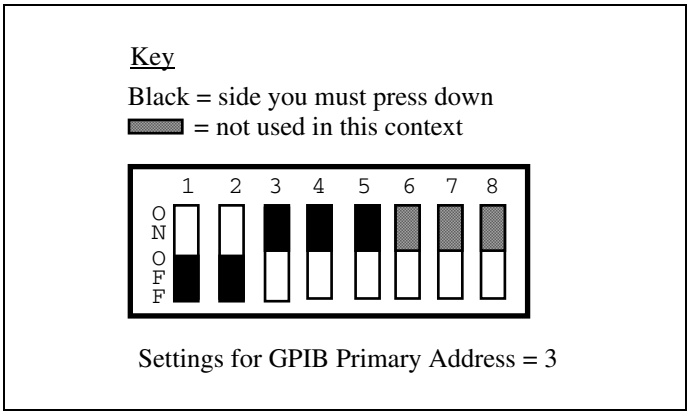


Figure 2-3. Example GPIB Primary Address Switch Setting

The primary address switches can be overridden by the nonvolatile memory configuration as described in the *Change Configuration Information* section of Chapter 4, *Nonvolatile Configuration*.

Setting the Servant Area Size

The servant area size is an 8-bit value (0 through 255) that indicates the GPIB-VXI servant area. The servant area begins at the logical address following the GPIB-VXI's logical address, and includes *N* contiguous logical addresses, where *N* is the value of the servant area size. The RM uses the servant area of all commanders in the VXIbus system to configure the commander/ servant hierarchy, as described in the VXIbus specification. Notice that if the GPIB-VXI is RM, the servant area size does not apply.

The servant area size switches 8 through 1 correspond to setting the GPIB-VXI servant area size bits 7 through 0, respectively. The *ON* position corresponds to a bit value of 0, and *OFF* corresponds to a value of 1. To set the servant area size to 7 (07h), for example, set the switches as shown in Figure 2-4.

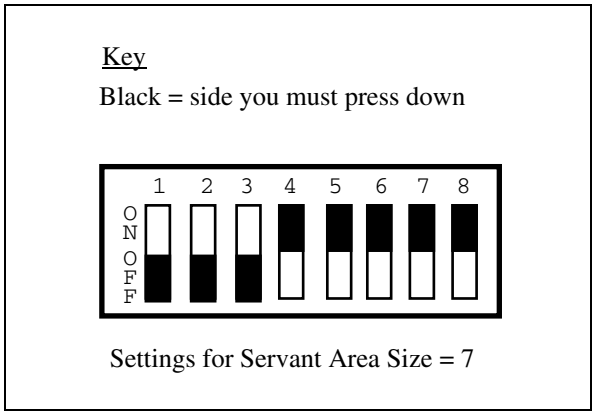


Figure 2-4. Example Servant Area Size Switch Setting

The servant area size switch can be overridden by the nonvolatile memory configuration as described in the *Change Configuration Information* section of Chapter 4, *Nonvolatile Configuration.*

Setting the Installed RAM Size

You can install up to 4M of local RAM on the GPIB-VXI. The minimum amount of memory is 512K. You can install additional memory by inserting 256K by 8-bit (Texas Instruments part number TMS41256GU8 or equivalent) or 1M by 8-bit (Texas Instruments part number TMS024EAD9 or equivalent) DRAM SIMM modules into the SIMM sockets as illustrated in Figure 2-1. You must install the RAM modules in *pairs* because the SIMM sockets accommodate *two SIMMs each*. The allowed RAM configurations and their associated switch settings are given in Table 2-3.

Note: Use only memory modules with an access time of 120 nsec or less. When installing the memory modules, follow proper procedures for handling MOS ICs.

Remove any memory modules you have installed prior to returning the GPIB-VXI to National Instruments for upgrades or repair.

Table 2-3. Installed RAM Switch Settings

Installed Memory Size	Switch S8 Setting	Switch S9 Setting	RAM SIMM Pair 1	RAM SIMM Pair 2
512K	OFF	OFF	256K by 8-bit	none
1M	OFF	ON	256K by 8-bit	256K by 8-bit
2M	ON	OFF	1M by 8-bit	none
4M	ON	ON	1M by 8-bit	1M by 8-bit

The relationship between the amount of installed memory, the local address range occupied by the memory, and range of VME A24 addresses accessible by the GPIB-VXI CPU is listed in Table 2-4.

Table 2-4. GPIB-VXI CPU Local and A24 Memory Ranges

Installed Memory Size	Installed Memory Local Address Range		Accessible VME A24 Address Range	
	Start	End	Start	End
512K	000000h	07FFFFh	080000h	E7FFFFh
1M	000000h	0FFFFFh	100000h	E7FFFFh
2M	000000h	1FFFFFh	200000h	E7FFFFh
4M	000000h	3FFFFFh	400000h	E7FFFFh

Setting the Dual-Ported Memory Size

The amount of installed memory that is dual-ported to the VMEbus can be set with switches S11 and S12. Table 2-5 gives the S11 and S12 switch settings for dual-porting various portions of RAM with the VMEbus, for each possible installed memory configuration.

Table 2-5. Dual-Ported Memory Size Configuration Switch Settings

Installed Memory Size	**Dual-Ported Portion of Installed Memory**			
	All	**One-Half**	**One-Fourth**	**None**
Switch Positions	**S12 ON S11 ON**	**S12 ON S11 OFF**	**S12 OFF S11 ON**	**S12 OFF S11 OFF**
512K	512K	256K	128K	0K
1M	1M	512K	256K	0K
2M	2M	1M	512K	0K
4M	4M	2M	1M	0K

The dual-ported memory VME A24 base address is held in the GPIB-VXI Offset Register, as described in the VXIbus specification.

Setting the Front Panel Reset Operation

The Reset button on the front panel can be configured to reset the GPIB-VXI and drive SYSRESET on the VXIbus backplane or just reset the GPIB-VXI. If jumper W23 is installed, the GPIB-VXI is reset *and* SYSRESET is driven. If jumper W23 is *not* installed, only the GPIB-VXI is reset.

Setting the VMEbus Requester Level

The VMEbus requester level of the GPIB-VXI is jumper-configurable as shown in Figure 2-5.

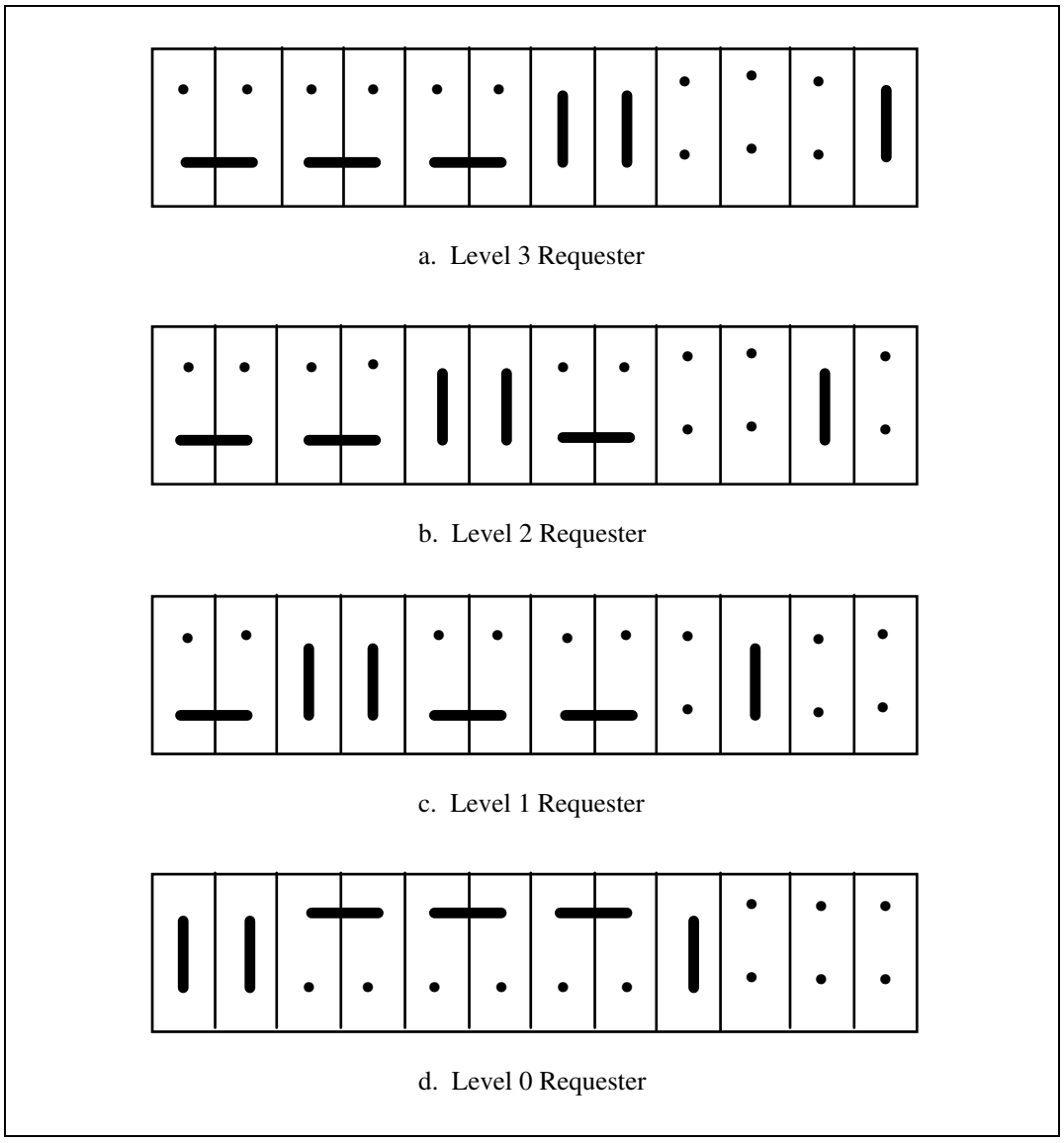


Figure 2-5. VMEbus Requester Jumper Settings

Setting the VXI Interrupt Handler Levels

The three default VXI interrupt handler levels are configured by the contents of the onboard nonvolatile memory, as described in Chapter 4, *Nonvolatile Configuration*.

GPIB-VXI Startup Mode Configuration

Startup mode switches 8 and 7 control the GPIB-VXI operation mode at system startup and system reset, respectively. They select one of four modes, as shown in Figure 2-6.

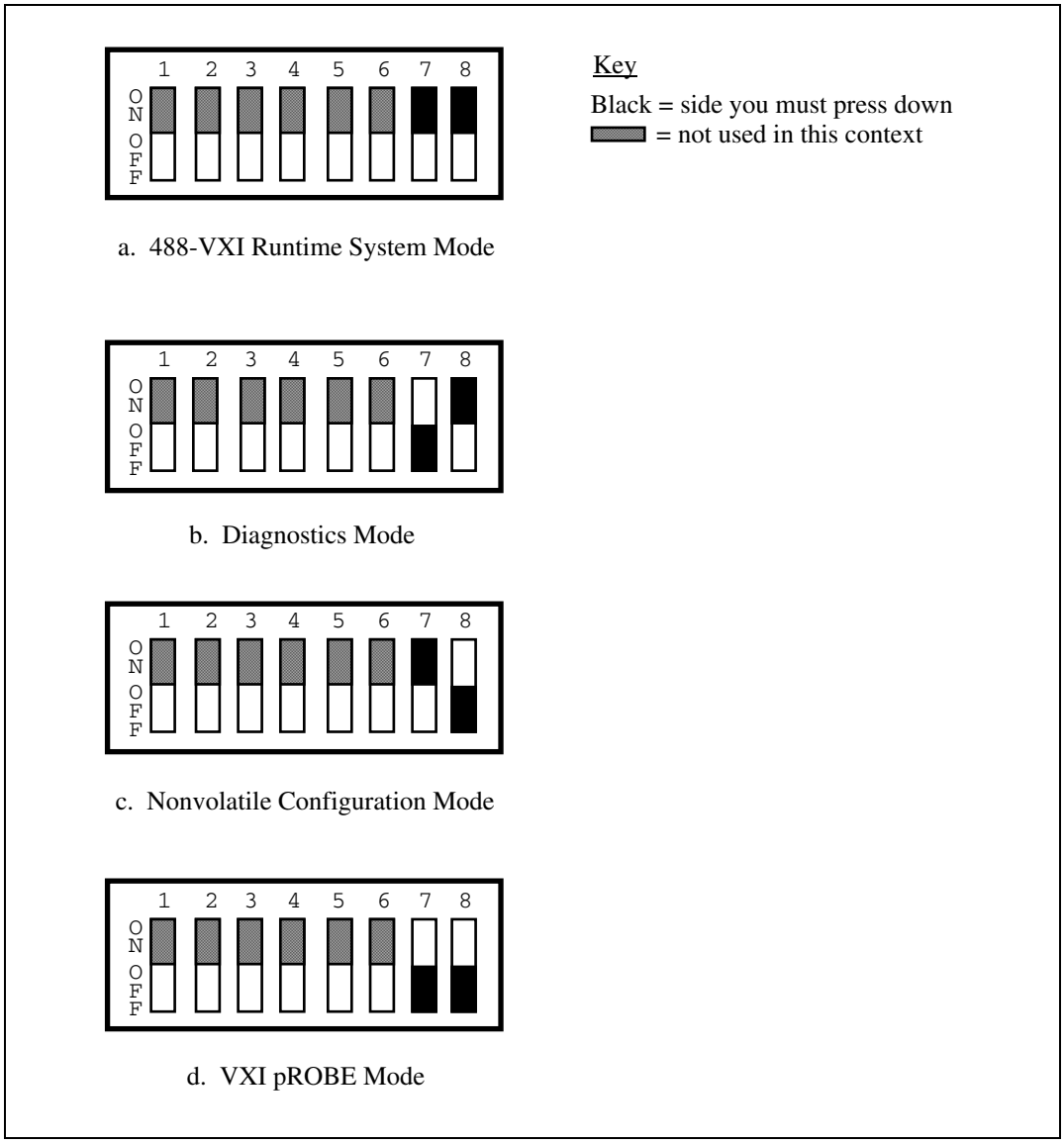


Figure 2-6. Startup Mode Switch Settings

488-VXI System Mode

VXI system mode is the startup mode for normal operation in a VXI system. The GPIB-VXI comes up as described in the *488-VXI System Operation* section later in this chapter.

Diagnostics Mode

In *diagnostics mode*, you can perform extensive offline diagnostic tests on the GPIB-VXI. See Chapter 5, *Diagnostic Tests*, for a description of the GPIB-VXI self-tests.

Nonvolatile Configuration Mode

In *nonvolatile configuration mode*, you can edit the contents of the nonvolatile configuration parameter memory. See Chapter 4, *Nonvolatile Configuration*, for a description of how to edit the GPIB-VXI nonvolatile memory.

VXI pROBE Mode

In *VXI pROBE mode*, you can use the enhanced pROBE debugger. This mode is available only with the GPIB-VXI development firmware option. The VXI pROBE debugger is described in the *GPIB-VXI Software Reference Manual*, part number 320152-01.

488-VXI System Operation

The GPIB-VXI is factory configured as a Slot 0 Resource Manager. The Slot 0 and Resource Manager (RM) functions can be independently defeated, resulting in four modes of operation:

- Slot 0 Resource Manager
- Non-Slot 0 Resource Manager
- Non-Slot 0 Message-Based device
- Slot 0 Message-Based device

This section describes the GPIB-VXI configuration procedures and startup behavior for each mode of operation.

Warning: Installation of a Non-Slot 0-configured GPIB-VXI in Slot 0 or a Slot 0-configured GPIB-VXI in any slot other than Slot 0 is not allowed, and may result in damage to the GPIB-VXI, the mainframe, or other modules.

System Startup Message Printing

The serial port startup printout enable switch controls whether or not VXI system startup messages are printed to the serial port, as shown in Figure 2-7.

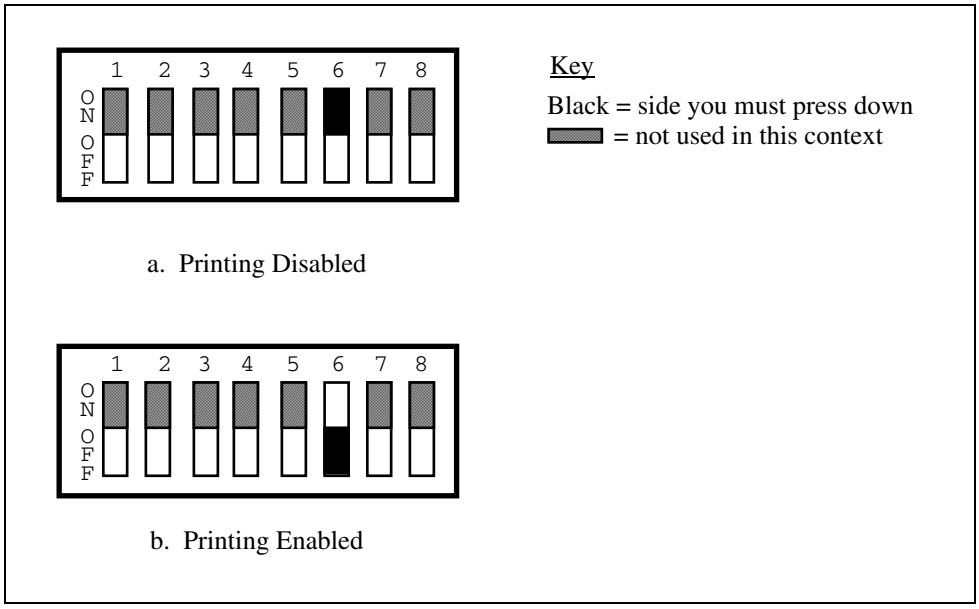


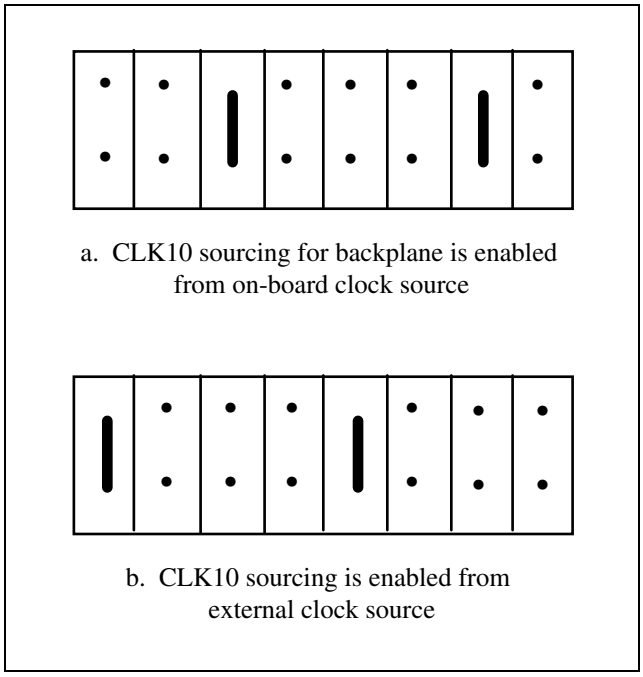
Figure 2-7. VXI System Startup Message Switch Settings

Slot 0 Resource Manager Configuration

You can configure the GPIB-VXI for Slot 0 Resource Manager operation by enabling the VXIbus Slot 0 functions, setting the model code to 0FFh, and setting the logical address to 0, as shown in Table 2-6.

Table 2-6. Slot 0 Resource Manager Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
CLK10 jumpers	See Figure 2-8 a.	CLK10 sourcing for backplane is enabled from onboard clock source. If S24 is OFF, the GPIB-VXI will also source CLK10 at the front panel BNC.
	or	
	See Figure 2-8 b.	CLK10 sourcing is enabled from external source via front panel BNC (S24 must be ON).
Switch S1	ON	SYSCLK sourcing is enabled.
Switch S2	ON	MODID pull up resistor is enabled.
Switch S5	ON	Bus arbiter is enabled.
Switch S10	ON	Model code is 0FFh.
Logical Address Switches 8 through 1	All ON	Logical address is 0.



a. CLK10 sourcing for backplane is enabled from on-board clock source

b. CLK10 sourcing is enabled from external clock source

Figure 2-8. CLK10 Jumper Settings for Slot 0 Resource Manager Operation

Slot 0 Resource Manager Operation

At startup, a GPIB-VXI configured as a Slot 0 Resource Manager performs its self-tests, executes the RM functions, and then enters its normal mode of operation.

Front Panel LED Indications for RM Operation

The five front panel LEDs are SYSFAIL, FAILED, TEST, ONLINE, and ACCESS. The GPIB-VXI uses the FAILED, TEST, and ONLINE LEDs to indicate the progress of its self-initialization, self-test, and RM functions. The LED indications are shown in Table 2-7. A successful system startup will sequence through the first five states. The probable cause of failure shown for each combination of LEDs is valid if the FAILED LED is lit five seconds following system startup. The LED indications are identical for Slot 0 Resource Manager and Non-Slot 0 Resource Manager operation.

Table 2-7. Front Panel LED Indications for RM Operation

Sequence	FAILED	TEST	ONLINE	State	Point of Failure
1	OFF	OFF	OFF	No power	
2	ON	OFF	OFF	In self-initialization	Failed before self-test
3	ON	ON	OFF	In self-test	Failed in self-test
4	OFF	ON	ON	Performing RM	
5	OFF	OFF	ON	Online	
	ON	ON	ON	Failed	Failed while in RM
	ON	OFF	ON	Failed	Failed while online

The SYSFAIL LED indicates the status of the VMEbus SYSFAIL signal. If any device in the system is asserting SYSFAIL, the SYSFAIL LED is lit.

The ACCESS LED flashes whenever the GPIB-VXI is accessed from the GPIB or from the VXIbus.

Self-Test Operation

The self-test sequence tests the basic functionality of many GPIB-VXI components, including EPROM, RAM, I^2C bus, RS-232 port, DMA channels, GPIB port, interrupt logic, timer, and VXIbus registers. The five-second limitation imposed by the VXIbus specification does not allow exhaustive tests to be executed at system startup. Full tests of the GPIB-VXI can be executed in diagnostics mode, as described in Chapter 5, *Diagnostic Tests*.

RM Operation

The RM waits until all devices have stopped driving the VME SYSFAIL signal, or until five seconds have elapsed after the VMEbus SYSRESET signal is negated. During this period, all of the VXIbus devices in the system should have completed their self-tests.

The RM then scans logical addresses 1 through 254 for *static configuration devices (SC devices)*. For each SC device found, it reads the device class and manufacturer's ID code from the ID Register, and the model code from the Device Type Register. If the device is an extended device, the RM reads its Subclass Register. The RM then performs slot associations for each static configuration device by reading its Status Register while asserting each MODID line.

The RM then looks for *dynamic configuration devices (DC devices)* at logical address 255 by asserting each MODID line and reading the device's ID Register. DC devices initially have a logical address of 255. The RM subsequently assigns each DC device a different logical address. For each DC device found, it reads the device's configuration registers, as with SC devices, but also assigns each device the next unused logical address by writing the appropriate value to the device's Logical Address Register. The starting logical address to begin assigning DC devices may be configured in the nonvolatile configuration, as described in Chapter 4, *Nonvolatile Configuration.*

If any device has not passed its self-test, the RM forces that device offline by setting the Sysfail Inhibit and Reset bits in that device's Control Register.

The RM then determines the address space of each device by reading its ID Register. If the device's address space is A16/A24 or A16/A32, the RM allocates a section of A24 or A32 memory space to the device according to the memory requirements indicated by the content of its Device Type Register, and writes the appropriate value to the device's Offset Register.

The RM configures the initial commander/servant hierarchy according to each commander's servant area size, using the algorithm described in the VXIbus specification. The RM issues the appropriate *Read Servant Area* and *Device Grant* commands to each SC commander. The RM retains all devices not assigned to other commanders as its immediate servants. Regardless of where DC device logical addresses are assigned, they are never granted to an SC commander. DC commander/servant hierarchy creation is done through the use of the GPIB-VXI Local Command Set, as described in the *DC Commands and Queries* section of Chapter 3, *Local Command Set*.

The RM then sends the Word Serial Query Read Protocols to all Message-Based devices. The response to the query is saved internally for later use in interrupt handler and GPIB configuration.

The RM configures the VXI interrupter and interrupt handlers using a seven entry table contained in nonvolatile configurations. During the VXI interrupt configuration, all Programmable Handlers (PH) and Programmable Interrupters (PI) are assigned interrupt levels. Each entry in the table represents the logical address of the handler that handles the corresponding level (1 through 7). If the handler is static, PI servants are assigned to the level. If the device is a PH device, both it and any PI servants are assigned to the corresponding level. Notice that if the table entry is FFh, the level is free to be assigned to any PH device. If only PH and PI devices are in a system, all entries may contain FFh. See Chapter 4, *Nonvolatile Configurations*, for more complete details.

The remainder of the RM procedure depends upon whether the RM found any DC devices in the system.

Static Configuration Operation

When all of the previous operations are complete and successful, the RM sends the Word Serial command *Identify Commander* to all immediate Message-Based servants with the master capability. At this point, the RM is ready to bring the system into the Normal Operation sub-state. This is accomplished by sending the Word Serial query *Begin Normal Operation* to all top-level commanders and immediate Message-Based servants.

Dynamic Configuration Operation

If the system is a DC system (at least one DC device was found), the GPIB-VXI RM does not send *Identify Commander* or *Begin Normal Operation* to any devices. The outside controller (or embedded CI) can then create the DC commander/servant hierarchy without having to dynamically reconfigure the system. Use the GPIB-VXI local command `DCGrantDev` to create the DC hierarchy. When the system is configured and ready to make a transition to the Normal Operation sub-state, send the GPIB-VXI local command `DCBNOSend`. `DCBNOSend` sends the *Identify Commander* and *Begin Normal Operation* commands to Message-Based devices as described in *Static Configuration Operation*. See the *DC Commands and Queries* section of Chapter 3, *Local Command Set* for further information about dynamic configuration operation.

The GPIB-VXI then performs general configuration operations. The GPIB-VXI creates GPIB secondary address links for its immediate Message-Based SC servants. After this, the GPIB-VXI RM and general configuration operations are complete.

GPIB Secondary Address Assignment

The GPIB-VXI automatically assigns a GPIB secondary address to itself and to each of its immediate Message-Based SC servants. If the Message-Based device does not support minimal Word Serial[I] or VXIbus 488.2[I4] capabilities, no GPIB secondary address link is created. The GPIB-VXI assigns a secondary address to each device according to the top five bits of its logical address. For example, the secondary address of a device with logical address 96 (01100000b) would be 12 (01100b).

If two or more devices have logical addresses with the same top five bits, the GPIB-VXI assigns secondary addresses to devices in order of the least significant three bits. Conflicting devices are given the next available secondary address. For example, if the GPIB-VXI and its Message-Based servants have logical addresses 0, 24, 27, and 33, the GPIB-VXI assigns secondary addresses as shown in Table 2-8.

Table 2-8. Example Secondary Address Assignment

Logical Address	**Logical Address**		**Upper 5 Bits**	**Secondary Address**
3 LSB Group (Order of Assignment)	**Decimal**	**Binary**		**Decimal**
000	0	00000000h	00000h	0
	24	00011000h	00011h	3
001	33	00100001h	00100h	4
010	none			
011	27	00011011h	00011h	5
100	none			
101	none			
110	none			
111	none			

In the example shown in Table 2-8, the device at logical address 27 was assigned secondary address 5 because addresses 3 and 4 were previously assigned. Spacing the GPIB-VXI's Message-Based servants at intervals of eight logical address locations is recommended in order to avoid situations in which removing or adding one device changes the secondary address of another device.

The self-assigned default secondary address of the GPIB-VXI can be overridden by the nonvolatile memory configuration as described in the *Change Configuration Information* section of Chapter 4, *Nonvolatile Configuration*.

System Configuration Table

During the execution of the RM and general configuration operations, the GPIB-VXI builds up a table of system configuration information. Each device has an entry in the table containing the [illegible]he secondary address entry is only meaningful for immediate Message-Based servants of the GPIB-VXI.

Non-Slot 0 Resource Manager Configuration

You can configure the GPIB-VXI for Non-Slot 0 Resource Manager operation by disabling the VXIbus Slot 0 functions, setting the model code 8FFh, and setting the logical address to 0, as shown in Table 2-9.

Table 2-9. Non-Slot 0 Resource Manager Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
CLK10 jumpers enabled.	See Figure 2-9.	CLK10 receiving from backplane is If S24 is OFF, the GPIB-VXI will source CLK10 at the front panel BNC.
Switch S1	OFF	SYSCLK sourcing is disabled.
Switch S2	OFF	MODID pull-down resistor is disabled.
Switch S5	OFF	Bus arbiter is disabled.
Switch S10	OFF	Model code is 8FFh.
Logical Address Switches 8 through 1	All ON	Logical address is 0.

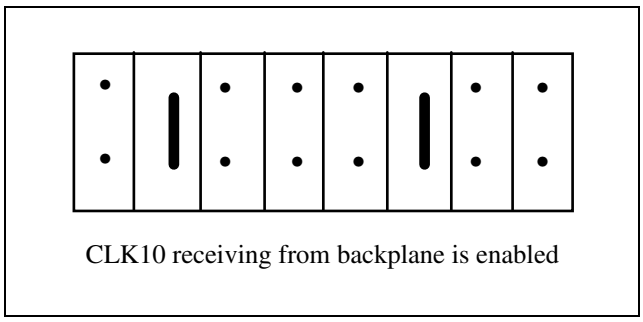


Figure 2-9. CLK10 Jumper Settings for Non-Slot 0 Resource Manager Operation

Non-Slot 0 Resource Manager Operation

The startup sequence for a GPIB-VXI configured for Non-Slot 0 Resource Manager operation is identical to the Slot 0 Resource Manager operation, except that the GPIB-VXI controls the Slot 0 resources remotely.

A VXIbus Slot 0 device must be in the system. It must be either a Register-Based device that implements the MODID Register, or a Message-Based device that supports the Word Serial commands *Read MODID*, *Set Lower MODID*, and *Set Upper MODID*. VXIbus Specification Revision 1.2 Word Serial Slot 0 devices are *not* supported.

Non-Slot 0 Message-Based Device Configuration

You can configure the GPIB-VXI for Non-Slot 0 Message-Based device operation by disabling the VXIbus Slot 0 functions, setting the model code to 8FFh, and setting the logical address to a non-zero value, as shown in Table 2-10.

Table 2-10. Non-Slot 0 Message-Based Device Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
CLK10 jumpers enabled.	See Figure 2-10.	CLK10 receiving from backplane is If S24 is OFF, the GPIB-VXI will source CLK10 at the front panel BNC.
Switch S1	OFF	SYSCLK sourcing is disabled.
Switch S2	OFF	MODID pull-down resistor is disabled.
Switch S5	OFF	Bus arbiter is disabled.
Switch S10	OFF	Model code is 8FFh.
Logical Address Switches 8 through 1	At least one is OFF	Logical address is not equal to 0.

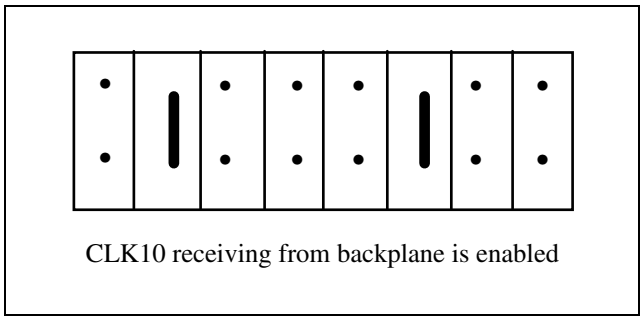


Figure 2-10. CLK10 Jumper Settings for Non-Slot 0 Message-Based Device Operation

Non-Slot 0 Message-Based Device Operation

Setting the logical address to FFh causes the GPIB-VXI to participate in dynamic configuration.

At startup, a GPIB-VXI configured as a Non-Slot 0 Message-Based device performs its self-tests, then waits until it receives its *Device Grant* and *Begin Normal Operation* Word Serial commands. When it responds to the *Begin Normal Operation* command, the GPIB-VXI enters its normal mode of operation.

Front Panel LED Indications for Message-Based Device Operation

The GPIB-VXI indicates the progress of its self-test with the FAILED, TEST, and ONLINE LEDs. The LED indications are shown in Table 2-11. A successful system startup sequences through the first five states. The probable cause of failure shown for each combination of LEDs is valid if the FAILED LED is lit five seconds following system startup. The LED indications are identical for Non-Slot 0 Message-Based device and Slot 0 Message-Based device operation.

Table 2-11. Front Panel LED Indications for Message-Based Device Operation

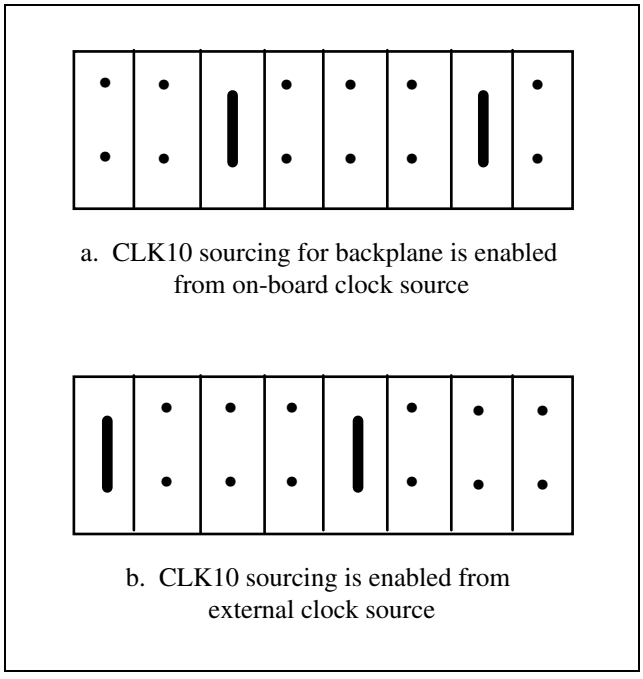
Sequence	FAILED	TEST	ONLINE	State	Point of Failure
1	OFF	OFF	OFF	No power	
2	ON	OFF	OFF	In self-initialization	Failed before self-test
3	ON	ON	OFF	In self-test	Failed self-test
4	OFF	ON	ON	Waiting for BNO	
5	OFF	OFF	ON	Online	
	ON	OFF	ON	Failed	Failed while online

Slot 0 Message-Based Device Configuration

The GPIB-VXI is configured for Slot 0 Message-Based device operation by enabling the VXIbus Slot 0 functions, setting the model code to 0FFh, and setting the logical address to a non-zero value, as shown in Table 2-12.

Table 2-12. Slot 0 Message-Based Device Operation Switch and Jumper Settings

Jumper/Switch	Position	Function
CLK10 jumpers	See Figure 2-11 a.	CLK10 sourcing for backplane is enabled from onboard clock source. If S24 is OFF, the GPIB-VXI will also source CLK10 at the front panel BNC.
	or	
	See Figure 2-11 b.	CLK10 sourcing is enabled from external clock source via front panel BNC (S24 must be ON).
Switch S1	ON	SYSCLK sourcing is enabled.
Switch S2	ON	MODID pull-up resistor is enabled.
Switch S5	ON	Bus arbiter is enabled.
Switch S10	ON	Model code is 0FFh.
Logical Address Switches 8 through 1	At least one is OFF	Logical address is not equal to 0.



a. CLK10 sourcing for backplane is enabled from on-board clock source

b. CLK10 sourcing is enabled from external clock source

Figure 2-11. CLK10 Jumper Settings for Slot 0 Resource Manager Operation

Slot 0 Message-Based Device Operation

At startup, a GPIB-VXI configured as a Slot 0 Message-Based device performs its self-tests, then waits until it receives its *Device Grant* (if any) and *Begin Normal Operation* Word Serial commands. When the GPIB-VXI responds to the *Begin Normal Operation* command, it enters the normal mode of operation.

After the GPIB-VXI Passed bit is set, the RM can manipulate or read the MODID lines by sending the Word Serial queries *Read MODID*, *Set Lower MODID*, or *Set Upper MODID* to the GPIB-VXI.

Chapter 3
Local Command Set

This chapter contains descriptions of the GPIB-VXI local command set. The descriptions of the commands and queries include syntax, format and error handling information, as well as examples of the use of the commands and queries. The local command set supports the following types of operations:

- System configuration and control
  - Help
  - General configuration
  - Resource Manager (RM) information extraction
  - Dynamic system configuration and reconfiguration
  - VXI-defined Common ASCII System Commands
  - GPIB address configuration
  - VXIbus interrupt handler configuration
  - IEEE-488.2 common commands
- Instrument development and test
  - VXIbus access
  - TTL Trigger access
  - Word Serial communication
- Code Instrument (CI) use and development
  - CI configuration

The GPIB-VXI command set consists of commands and queries. *Commands* cause the GPIB-VXI to take some action. A *query* may also cause the GPIB-VXI to take some action, but it also returns a response containing data or other information.

Command Set Access

The local commands can be executed from the following ports:

- RS-232
- GPIB
- VXI Word Serial communication

The three ports are active when the GPIB-VXI is in the Normal Operation sub-state and operate independently of one another. The GPIB-VXI returns query responses only to the port originating the query. The GPIB-VXI also maintains a separate status state for each port. You can use local commands to disable and re-enable each port's access to the local command set. The RS-232 port prompts you to enter a local command with the `GPIB-VXI>` prompt.

Command Syntax

The local command set parser is syntactically compatible with the IEEE-488.2 standard. It will accept numeric parameters in the 488.2 binary, octal, decimal, or hexadecimal formats. 488.2 binary parameters are prefixed with *#b*. Octal parameters are prefixed with *#q*, and hexadecimal parameters are prefixed with *#h*. The most common numeric parameter types are listed in Table 3-1. The ranges given in Table 3-1 apply unless otherwise specified.

Table 3-1. Valid Ranges for Common Numeric Command Parameters

Parameter	488.2 Decimal	488.2 Hexadecimal
`<logical address>`	0 to 254	#h0 to #hFE
`<secondary address>`	0 to 30	#h0 to #h1E
`<handler>`	1 to 3	#h1 to #h3
`<level>`	0 to 7	#h0 to #h7
`<A16 address>`	0 to 65535	#h0 to #hFFFF
`<A24 address>`	2097152 to 14680062	#h200000 to #hDFFFFE
`<word value>`	0 to 65535	#h0 to #hFFFF
`<byte value>`	0 to 255	#h0 to #hFF
`<boolean>`	0 or 1	#h0 or #h1

The logical value of a `<boolean>` parameter is FALSE for the numeric value 0, and TRUE for the numeric value 1.

The first parameter is delimited from the command name by a space ( ). Additional parameters are delimited from one another by a comma (`,`). The command names are not case-sensitive.

In the command descriptions, parameters are enclosed in angle brackets (< >), and optional parameters are also enclosed in square brackets ([ ]). Do not enter the brackets as part of the command.

Multiple commands may be concatenated in a single command line if they are separated with semicolons (for example, `OBRAM?; DPRAM?`<CR>).

Command Line Termination

The serial port command line termination is a carriage return, shown in the subsequent function descriptions as <CR> (ASCII 0Dh). If the command contains a trailing linefeed, shown in the subsequent function descriptions as <LF> (ASCII 0Ah), it is ignored. The GPIB termination is EOI. Commands issued to the GPIB-VXI via VXI Word Serial Protocol are terminated by setting the END bit in the last *Byte Available* command. Responses are terminated by setting the END bit in response to the last *Byte Request* query.

Command and Query Responses

The local commands and queries have two response formats–program mode and console mode. *Program mode responses* have a terse data-only format that is intended for a control program to read and parse. *Console responses* are returned in the form of readable sentences, which are better suited for interactive command entry.

You can enable or disable each mode independently, except that one response mode must be enabled at all times. If both modes are simultaneously enabled, the program response is returned first, followed by the console response.

The response mode configuration is independent for each command source. The default (startup/reset) response mode configurations are given in Table 3-2.

Table 3-2. Default Response Mode Configurations

Port	Response Mode
RS-232	Console mode enabled, program mode disabled
GPIB	Program mode enabled, console mode disabled
VXI Word Serial	Program mode enabled, console mode disabled

Command Response Format

Commands do not have program mode responses. They do not return a response to a port configured for console mode response only, unless the GPIB-VXI detects an error condition.

Console mode command responses are self-explanatory, and are not described in this manual.

Query Response Format

Queries have both program and console mode responses. Program mode query responses are fixed-field formatted, with commas delimiting the fields. For example, the list of logical addresses returned by the `Laddrs?` query is returned as groups of three characters (to allow the field to accommodate the valid range of 0 to 254) separated by commas. The values are right-justified and padded with the ASCII space character ( ) (20h). For example, the logical address 45 would be returned as `( )45`. Unless otherwise noted, all returned values are decimal.

Console mode query responses are self-explanatory, and are not described in this manual.

The query response line termination sequence, shown in the query descriptions as <CRLF>, indicates an ASCII 0Dh followed by 0Ah.

Error Reporting

Command syntax and execution errors are reported to the port where the command originated. If the program response mode is enabled, the GPIB-VXI returns an error message in the following format:

```
$ <error code><CRLF>
```

The distinguishing characteristic of a program mode error message is the leading dollar sign character (`$`). A list of error code descriptions is given in Appendix B, *Error Codes*.

If the console response mode is enabled, the GPIB-VXI returns an error message in the following format:

```
<error description><CRLF>
```

If both response modes are enabled, the program mode error message is returned first, followed by the console mode message.

The Help Query

The `Help?` query is a quick online reference to the syntax and functionality of the GPIB-VXI local command set.

Help?

Purpose: List syntax and descriptions of local command set.

Query Syntax:

```
Help?  [<type>[,<type>,....]]
```

or

```
Help  [<type>[,<type>,....]]
```

`<type>` is the category of command information requested, as follows:

`he`	Help	`ci`	Code Instruments
`al`	All	`sa`	GPIB secondary address
`gc`	General configuration	`ih`	Interrupt handler configuration
`dc`	Dynamic configuration	`ba`	VXIbus access
`rc`	Dynamic reconfiguration	`ws`	Word Serial communication
`rm`	Resource Manager	`tr`	TTL Trigger access
`cc`	Common commands		

The default type is All.

Response: The local command set is displayed in the following format:

```
                GPIB-VXI Local Command Set<CRLF>

      Command/Query Format                Description<CRLF>
-------------------------------------------------------------------
<Command Syntax>                    <Command description><CRLF>
<Command Syntax>                    <Command description><CRLF>
<Command Syntax>                    <Command description><CRLF>
      .                                     .
      .                                     .
      .                                     .
```

Example: List syntax and descriptions of general configuration and secondary address commands.

```
Help?  gc,sa
```

General Configuration Commands and Queries

The general configuration commands and queries are described on the following pages.

- `ConsoleEna`
- `ConsMode`
- `DPRAM?`
- `NVconf?`
- `OBRAM?`
- `ProgMode`
- `WordSerEna`

The `ConsMode` and `ProgMode` commands enable and disable the console and program response modes for the port originating the command.

The `NVconf?` query returns the contents of the onboard nonvolatile memory.

The `ConsoleEna` and `WordSerEna` commands control access to the local command set from the RS-232 and VXI Word Serial ports.

The `OBRAM?` query can be used to determine the amount of GPIB-VXI installed RAM, and the `DPRAM?` query returns the amount of the installed RAM that is dual-ported to VME A24 space.

ConsoleEna

Purpose: Enable or disable the RS-232 port as the console. When the RS-232 port is disabled as the console, a CI can take control of the serial port.

Command Syntax: `ConsoleEna <boolean>`

Action: If `<boolean>` is TRUE, `ConsoleEna` sets the RS-232 port to be a local command set input.

If `<boolean>` is FALSE, `ConsoleEna` disables the RS-232 port connection to the local command set. Notice that once the console has been disabled, it must be re-enabled from another command source (such as the GPIB port).

Examples: Disable console.

```
ConsoleEna 0
```

Enable console.

```
ConsoleEna 1
```

ConsMode

Purpose: Enable or disable the console data mode.

Command Syntax: `ConsMode <boolean>`

Action: If `<boolean>` is TRUE, `ConsMode` enables console format responses for the command source issuing the command.

If `<boolean>` is FALSE, `ConsMode` disables console format responses for the command source issuing the command.

The console response mode applies only to the response path connected to the `ConsMode` command source. For example, disabling the console response mode from the GPIB port does not affect the response mode on the serial port.

Example: Disable console format responses.

```
ConsMode 0
```

Enable console format responses.

```
ConsMode 1
```

DPram?

Purpose: Get the A24 starting address and the size of dual-ported RAM.

Query Syntax: `DPram?`

Response: Program response:

`<A24 starting address>, <dual-ported RAM size>`<CRLF>

Console response:

```
This GPIB-VXI has <dual-ported RAM size>K bytes Dual-Ported to A24
Address <A24 hex starting address><CRLF>
```

where `<A24 starting address>` is the dual-ported RAM base address in decimal integer format.
`<dual-ported RAM size>` is in K bytes.
`<A24 hex starting address>` is in C language hexadecimal format.

NVconf?

Purpose: Display the contents of the nonvolatile (NV) configuration parameter memory.

Query Syntax: `NVconf?`

Response: The contents of the onboard EEPROM are displayed in the following format:

```
=============== Nonvolatile Configuration Information ===============

Serial Number  : 0x68
Region 1 Size  : 0x70000          Number Procs   : 32
Number Exchgs  : 32               Number Msgs    : 384
VXI Interrupt Level to Handler Logical Address (0xFF = free to assign):
  1:0xFF  2:0xFF  3:0xFF  4:0xFF  5:0xFF  6:0xFF  7:0xFF
DC Starting LA : 0x01             For FAILED DEV : DO set Reset bit
A24 Assign Base: 0x200000         A32 Assign Base: 0x20000000
OverRide Srvnt : NO               Servant Area   : N/A
OverRide Prim  : NO               GPIB Primary   : N/A
GPIB Sec Addr  : Default          GPIB Flags     : DMA 7210

CI Block Base  : 0x80000          CI Num Blocks  : 128

          ------ Resident Code Instrument Locations ------
# 0:       0             # 1:       0             # 2:       0
# 3:       0             # 4:       0             # 5:       0
# 6:       0             # 7:       0             # 8:       0
# 9:       0             # a:       0             # b:       0

      ------ CI Nonvolatile User Configuration Variables ------
# 0:      0       # 1:      0       # 2:      0       # 3:      0
# 4:      0       # 5:      0       # 6:      0       # 7:      0
# 8:      0       # 9:      0       #10:      0       #11:      0
#12:      0       #13:      0       #14:      0       #15:      0
#16:      0       #17:      0       #18:      0       #19:      0
#20:      0       #21:      0       #22:      0       #23:      0
#24:      0       #25:      0       #26:      0       #27:      0
#28:      0       #29:      0       #30:      0       #31:      0
```

OBram?

Purpose: Get the amount of RAM installed onboard the GPIB-VXI.

Query Syntax: `OBram?`

Response: Program response:

`<memsize>`<CRLF>

where `<memsize>` is the amount of installed RAM, in K.

Console response:

`This GPIB-VXI has <expression> of RAM installed onboard.`<CRLF>

where `<expression>` is the amount of installed RAM.

ProgMode

Purpose: Enable or disable the program data mode.

Command Syntax: `ProgMode <boolean>`

Action: If `<boolean>` is TRUE, `ProgMode` enables program format responses for the command source issuing the command.

If `<boolean>` is FALSE, `ProgMode` disables program format responses for the command source issuing the command.

The program response mode applies only to the response path connected to the `ProgMode` command source. For example, disabling the program response mode from the GPIB port does not affect the response mode on the serial port.

Examples: Disable program format responses.

```
ProgMode 0
```

Enable program format responses.

```
ProgMode 1
```

WordSerEna

Purpose: Assign control of the GPIB-VXI physical Word Serial registers to an onboard logical address (GPIB-VXI command interpreter or code instrument).

Command Syntax: `WordSerEna <logical address>`

Action: Control of the physical Word Serial registers is passed to `<logical address>`.

The default control of the physical registers is given to the GPIB-VXI local command set parser.

Examples: Pass control of the physical registers to code instrument at logical address 5.

```
WordSerEna 5
```

Pass control of the physical registers back to GPIB-VXI local command parser at logical address 0.

```
WordSerEna 0
```

RM Information Queries

The RM information queries are described on the following pages.

- `A24MemMap?`
- `A32MemMap?`
- `Cmdr?`
- `CmdrTable?`
- `Laddrs?`
- `NumLaddrs?`
- `RmEntry?`
- `Srvnts?`
- `StatusState?`

The `Numladdrs?` query is used to find out how many devices there are in the system. The number of devices could then be used by a control program to determine the allocation size for an array that is to hold the logical addresses of each device.

The `Laddrs?` query returns a list of logical addresses for devices in the system.

The `RmEntry?`, `Srvnts?`, `Cmdr?`, and `StatusState?` queries return RM information for a particular device.

The `CmdrTable?` query returns the system hierarchy table.

The `A24MemMap?` and `A32MemMap?` queries return the A24 and A32 memory configuration lists.

The system information commands (`NumLaddrs?`, `Laddrs?`, `CmdrTable?`, `A24MemMap?`, and `A32MemMap?`) return information about the known system. If the GPIB-VXI is the system RM, it can access information about the entire system. If it is not the RM, it has information only about itself and its immediate servants.

A24MemMap?

Purpose: Get the A24 address space allocation for the system.

Query Syntax: `A24MemMap?`

Response: Program response:

```
<la1>,<A24 memory base>,<A24 memory size><CRLF>
<la2>,<A24 memory base>,<A24 memory size><CRLF>
      •
      •
<laN>,<A24 memory base>,<A24 memory size><CRLF>
```

where `<la1>` through `<laN>` are logical addresses containing A24 address space.

Console response:

```
A24 Memory Map is as follows:<CRLF>

Logical Address <la1> has <A24 memory size>K
(<A24 memory size> bytes) at A24 Address<A24 memory base><CRLF>
      •
      •
Logical Address <laN> has <A24 memory size>K
(<A24 memory size> bytes) at A24 Address<A24 memory base><CRLF>
```

Example: Get A24 address map for the system.

```
A24MemMap?
```

A32MemMap?

Purpose: Get the A32 address space allocation for the system.

Query Syntax: `A32MemMap?`

Response: Program response:

```
<la1>,<A32 memory base>,<A32 memory size><CRLF>
<la2>,<A32 memory base>,<A32 memory size><CRLF>
      •
      •
<laN>,<A32 memory base>,<A32 memory size><CRLF>
```

where <la1> through <la*N*> are logical addresses containing A32 address space.

Console response:

```
A32 Memory Map is as follows:<CRLF>

Logical Address <la1> has <A32 memory size>K
(<A32 memory size> bytes) at A32 Address<A32 memory base><CRLF>
      •
      •
Logical Address <laN> has <A32 memory size>K
(<A32 memory size> bytes) at A32 Address<A32 memory base><CRLF>
```

Example: Get A32 address map for the system.

```
A32MemMap?
```

Cmdr?

Purpose: Get the logical address of a device's commander.

Query Syntax: `Cmdr? <logical address>`

where <logical address> is the logical address of the device.

Response: Program response:

```
<commander's logical address><CRLF>
```

Console response:

```
The Commander of Logical Address <logical address> is Logical
Address <commander's logical address><CRLF>
```

Example: Get the commander's logical address for logical address 15.

```
Cmdr? 15
```

CmdrTable?

Purpose: Get the known system hierarchy table.

Query Syntax: `CmdrTable?`

Response: Program response:

```
<cla0>,<cla1>,<cla2>,<cla3>,<cla4>, . . .,<cla254><CRLF>
```

where <cla*N*> is either the commander's logical address for logical address *N*, or 0 for top-level commanders and unused logical addresses. Notice that no value is returned for logical address 255.

Console response:

```
Known Hierarchy is as follows:<CRLF>
Logical address <la1> has servants: <sa1,1>,...,<sa1,M>  <comment><CRLF>
Logical address <la2> has servants: <sa2,1> ,...,<sa2,M> <comment><CRLF>
       •
       •
Logical address <laN> has servants: <saN,1> ,...,<saN,M> <comment><CRLF>
```

where <la*X*> is a valid logical address with servant addresses <sa*X*,1> through <sa1,M>.

The <comment> field indicates any relevant information about the status and/or capabilities of the device at logical address <la*X*>.

Laddrs?

Purpose: Get a list of the known logical addresses.

Query Syntax: `Laddrs?`

Response: Program response:

`<la1>,<la2>,..., <laN>`<CRLF>

where `<la1>` through `<laN>` are the known logical addresses.

Console response:

`Known logical addresses are <la1>,<la2>,..., <laN>`<CRLF>

CI logical addresses are terminated with an asterisk (`*`) in the console mode response.

NumLaddrs?

Purpose: Get the number of known logical addresses.

Query Syntax: `NumLaddrs?`

Response: Program response:

`<num las>`<CRLF>

where `<num las>` is the number of known logical addresses.

Console response:

`There are <num las> known Logical Addresses`<CRLF>

RmEntry?

Purpose: Return RM information about a device or all devices. `RmEntry?` does not return the servant list.

Query Syntax: `RmEntry? [<logical address>]`

(If `<logical address>` is omitted, `RmEntry?` returns the RM information for all devices.)

Response: Program response:

```
<la>,<cla>,<sa>,<slot>,<devclass>,<subclass>,<manID>,<modelcode>,
<memspace>,<membase>,<memsize>,<state>,<line status><CRLF>
```

Console response:

```
Resource manager entry for logical address <logical address>:
<CRLF>
<CRLF>
Commander's Logical Address  :<cla><CRLF>
Secondary Address            :<sa><CRLF>
Slot                         :<slot><CRLF>
Device class                 :<device class> (class)<CRLF>
Extended Sub Class           :<subclass><CRLF>
Manufacturer's ID            :<manID> (manufacturer's name)<CRLF>
Model code                   :<modelcode><CRLF>
Memory space                 :<memspace> (memory space)<CRLF>
Memory Base                  :<membase><CRLF>
Memory Size                  :<memsize>K (<memsize> bytes)<CRLF>
Status State                 :<state> (state)<CRLF>
Forced Offline?              :<line status> (yes/no)<CRLF>
```

The mnemonics have the following meanings:

`la`	device's logical address
`cla`	commander's logical address
`sa`	device's secondary address (255 if not assigned secondary address)
`slot`	slot number (255 if unknown, such as if the device does not have MODID capability)

`devclass` device class; the following values may be used:

0 = Memory
1 = Extended
2 = Message-Based
3 = Register-Based

`subclass` extended device subclass

`manID` manufacturer's ID number

`modelcode` device's manufacturer-assigned model code

`memspace` memory space requirement:

0 = A16 only
1 = A16/A24
2 = A16/A32

`membase` memory base address

`memsize` memory size in bytes

`state` status state:

0 = Failed and not Ready
1 = Passed and not Ready
2 = Failed and Ready
3 = Passed and Ready

`line status` online/offline status:

0 = online
1 = forced offline

The program mode response format is the same for all devices. However, the console mode response returns only the lines that are relevant. For example, memory base address and memory size lines are not returned for A16-only memory space devices.

Example: Get RM information for a device at logical address 78.

```
RmEntry? 78
```

Srvnts?

Purpose: Get a list of a device's servants.

Query Syntax: `Srvnts? <logical address>`

`<logical address>` is the device's logical address.

Response: Program response:

`<sla1>,<sla2>,...,<slaN>`<CRLF>

where `<sla1>` through `<slaN>` are the servant device logical addresses.

Console response:

```
Logical Address <logical address> has servants:
<sla1>, <sla2>,..., <slaN> <comment><CRLF>
```

if the device has servants, or

```
Logical Address <logical address> has servants:
none <comment><CRLF>
```

if the device has no servants.

The `<comment>` field indicates any relevant information about the status and/or capabilities of the device.

Example: Get a list of logical address 15's servants.

```
Srvnts? 15
```

StatusState?

Purpose: Get a device's current self-test status.

Query Syntax: `StatusState? <logical address>`

`<logical address>` is the logical address for the device.

Response: Program response:

`<val>`<CRLF>

The value of `<val>` is equivalent to the value of the field in the device's status register containing the Ready and Passed bits. `<val>` can be interpreted as follows:

- `0` The device is Failed and not Ready.
- `1` The device is Passed and not Ready.
- `2` The device is Failed and Ready.
- `3` The device is Passed and Ready.

Console response:

`Device at logical address <logical address> is FAILED and not Ready`<CRLF>

or

`Device at logical address <logical address> is PASSED and not Ready`<CRLF>

or

`Device at logical address <logical address> is FAILED and Ready`<CRLF>

or

`Device at logical address <logical address> is PASSED and Ready`<CRLF>

Example: Get logical address 48's self-test status.

```
StatusState? 48
```

Dynamic Configuration Commands and Queries

The dynamic configuration (DC) commands and queries are described on the following pages.

- `DCBNOSend`
- `DCGrantDev`
- `DCSystem?`

The DC commands are used to configure the VXI system when all of these conditions are present:

- The GPIB-VXI is the RM.
- At least one DC device is present in the system.
- The system is still in the startup Configure sub-state.

The DCSystem? query response indicates whether the system contains a DC device. If the system is found to be a DC system, the DCGrantDev command is used to configure the commander/servant hierarchy. The DCBNOSend command is used to end the DC phase and to cause the system to enter normal operation.

DCBNOSend

Purpose: Cause a DC system to exit the Configure sub-state and enter the Normal Operation sub-state.

Command Syntax: DCBNOSend

Action: Send the *Begin Normal Operation* command to all top-level commanders.

DCGrantDev

Purpose: Grant a device to a Message-Based commander in a DC system. DCGrantDev can be used only to configure the initial commander/servant hierarchy of a DC system, and before DCBNOSend is used to cause the system to enter the Normal Operation sub-state.

Command Syntax:
```
DCGrantDev  <commander's logical address>,
<servant's logical address>
```

Action: DCGrantDev sends the *Device Grant* command to the commander at <commander's logical address>, granting it the device at <servant's logical address>.

Example: Grant servant at logical address 7 to commander at logical address 5.

```
DCGrantDev 5,7
```

DCSystem?

Purpose: Determine if the system is a DC system. A system is DC if it has at least one DC device.

Query Syntax: DCSystem?

Response: Program response:

`1` <CRLF>

if it is a DC system, or

`0` <CRLF>

if it is not a DC system, or if it is no longer dynamically configurable because the *Begin Normal Operation* command has already been sent to the top-level commanders through the `DCBNOSend` local command.

Console response:

`This IS a Dynamic Configured system.`<CRLF>

if it is a DC system, or

`This is NOT a Dynamic Configured system.`<CRLF>

if it is not a DC system.

Dynamic Reconfiguration Queries

The dynamic reconfiguration queries are described on the following pages.

- `Broadcast?`
- `GrantDev?`
- `RelSrvnt?`

The dynamic reconfiguration commands are used to reconfigure the GPIB-VXI's servant subtree after the system has entered the Normal Operation sub-state. If the GPIB-VXI is RM, these commands can be used to reconfigure the entire system.

The `Broadcast?` query can be used to make the system or subtree enter the Configure sub-state by broadcasting the *End Normal Operation* Word Serial query, or the *Clear* Word Serial command followed by the *Abort Normal Operation* Word Serial query.

The `RelSrvnt?` and `GrantDev?` queries can then be used to restructure the commander/servant hierarchy. Dynamic reconfiguration could be directly performed by using the `WSCmd` and `WSCmd?` local commands, but the GPIB-VXI's RM table would not be updated. By using the `RelSrvnt?` and `GrantDev?` queries to reconfigure the system, you ensure that the GPIB-VXI's system hierarchy and secondary address link records do not become corrupted.

The system or subtree can be returned to the Normal Operation sub-state by using the `Broadcast?` query to broadcast the *Identify Commander* and *Begin Normal Operation* Word Serial commands.

Broadcast?

Purpose: Broadcast dynamic reconfiguration initialization or termination Word Serial commands to the GPIB-VXI's Message-Based servants or to all top-level commanders in the system.

Query Syntax: `Broadcast? <boolean>,<ws cmd>`

If `<boolean>` is 1, the GPIB-VXI broadcasts `<ws cmd>` to all top-level commanders. If `<boolean>` is 0, it broadcasts `<ws cmd>` to its Message-Based servants. Notice that the GPIB-VXI should only broadcast to top-level commanders when it is RM.

`<ws cmd>` is a mnemonic as follows:

<ws cmd>	**Word Serial Command Name**	**Type**
ANO	*Abort Normal Operation*	Query
BNO	*Begin Normal Operation*	Query
CLR	*Clear*	Command
ENO	*End Normal Operation*	Query
IDN	*Identify Commander*	Command

The `Broadcast?` query can fail due to inability to complete a Word Serial operation, or because an invalid code was returned from a device in response to ANO or ENO.

Response: Program response:

<CRLF>

if the command was successful, or

`<la>,<cmd val>,<ws response>,<ws error code>`

if the command failed.

Console response:

```
Done broadcasting Word Serial command:  <Word Serial command
name>.
```

if the command was successful, or

```
Logical address <la> returned <ws response> from ENO (Unable to
halt)
```

or

```
Logical address <la> returned <ws response> from ANO (Invalid
response)
```

or

```
Error sending logical address <la> word serial command <hex cmd
val><CRLF><space><space><ws error><CRLF>
```

if the command failed.

<la> is the logical address of the device to which the broadcast failed.

<cmd val> is the value of the Word Serial command, in decimal. <hex cmd val> is the value in hexadecimal.

For Word Serial queries, <ws response> is the Word Serial response of the device at logical address <la>. For Word Serial commands <ws response> is 0.

<Word Serial command name> is the name of the command name as shown in the previous table.

<ws error code> is a decimal value that can be interpreted by converting it to a binary bit pattern. A value of 1 in the bit positions shown in the following table indicates that an error occurred during the attempt to broadcast the Word Serial command:

Bit	Word Serial Error
0	Word Serial command completed successfully (no Word Serial error)
1	Timeout waiting to send Word Serial command to device at <la>
2	Timeout waiting for Word Serial response from device at <la>
3	Device at <la> did not recognize the command
6	Multiple query error
10	Read Protocol error not supported
13	Read Ready (RR) violation
14	Write Ready (WR) violation

None of the other bits has significance in this context.

<ws error> is a string explaining the Word Serial error as shown in the previous table.

Example: Broadcast the *Identify Commander* Word Serial command to all top-level commanders.

```
broadcast? 1,IDN
```

GrantDev?

Purpose: Grant a servant to a commander.

Query Syntax:
```
GrantDev? <commander's logical address>, <servant's logical address>
```

Action: Grants the device at <servant's logical address> to device at <commander's logical address>.

The GPIB-VXI must own the device at `<servant's logical address>`. The GPIB-VXI can get ownership of any device with the `RelSrvnt?` command.

Notice that before the `GrantDev?` query is used, the Word Serial *End Normal Operation* query, or a *Clear* command followed by the *Abort Normal Operation* query should have been broadcast with the `Broadcast?` query.

Response: Program response:

`0`<CRLF>

indicates that the command was successful.

Console response:

```
Logical Address <commander's logical address> granted device at
Logical Address <servant's logical address>.
```

Example: Grant device 16 to commander at logical address 8.

```
Grantdev? 8,16
```

RelSrvnt?

Purpose: Recover a servant from a commander.

Query Syntax: `RelSrvnt? <commander's logical address>, <servant's logical address>`

Action: Commands device at `<commander's logical address>` to release ownership of the device at `<servant's logical address>`. The GPIB-VXI assumes ownership of the device.

Response: Program response:

`254`<CRLF>

if the commander released the servant. Any other response indicates that an error occurred.

Console response:

```
Logical Address <commander's logical address> released device at
Logical Address <servant's logical address>.
```

Example: Recover servant at logical address 16 from commander at logical address 8.

```
Relsrvnt? 8,16
```

VXI-Defined Common ASCII System Commands

The VXI-defined Common ASCII System Commands and Queries are described on the following pages.

- DCON?
- DINF?
- DLAD?
- DNUM?
- DRES?
- RREG?
- WREG

These commands and queries can be used to retrieve device information/configuration, perform a soft reset, and peek/poke a device's registers.

The DNUM? query is used to find out how many devices are in the system. The DLAD? query returns a list of logical addresses for devices in the system.

The DINF? query returns static information about a device. The DCON? query returns configuration information about a device.

The DRES? query is used to perform a soft-reset sequence on a device.

The RREG? query and WREG command are used to peek and poke registers on a VXI device.

DCON?

Purpose: Return system configuration information about a device or all devices.

Query Syntax: DCON? [<logical address>]

(If <logical address> is omitted, DCON? returns the configuration information for all devices.)

Response: Program response:

```
<la1>,<cla>,<IHANS>,<INTS>,<status>,<sstate>,<com><CRLF>
```

Console response:

```
Device configuration at Logical Address <la>:<CRLF>
<CRLF>
Commander's Logical Address   :<cla><CRLF>
Interrupt Handlers            :<IHANS><CRLF>
Interrupters                  :<INTS><CRLF>
Passed/Failed/Ready           :<status><CRLF>
Device Substate               :<sstate><CRLF>
Manufacturer Specific Comment :<com><CRLF>
```

`la` device's logical address

`cla` commander's logical address

`IHANS` Interrupt handler levels used by this device where `IHANS` is a 7-digit binary representing the seven VXI interrupt levels and a one in each position, meaning Interrupt Handler present

`INTS` Interrupter levels used by this device where `INTS` is a 7-digit binary representing the seven VXI interrupt levels and a one in each position, meaning Interrupter present.

`status` the status state of the device:

PASS
FAIL
IFAIL
READY

`sstate` the substate of the device

NOP
CONF
NONE

`com` not used; always returns `""`

Example: Get device configuration information for logical address 6.

```
DCON? 6
```

DINF?

Purpose: Return static system information about a device.

Query Syntax: `DINF? [<logical address>]`

(If `<logical address>` is omitted, `DINF?` returns static information for all devices.)

Response: Program response:

```
<la1>,<manID>,<modelcode>,<devclass>,<memspace>,<membase>,
<memsize>,<slot>,<slot0>,<ext>,<attr>,<com><CRLF>
```

Console response:

```
Device configuration at Logical Address <la>:<CRLF>
<CRLF>
Manufacturer ID Number        :<manid> (manufacturer name)<CRLF>
Model Code :<modelcode><CRLF>
Device Class                  :<devclass><CRLF>
A16/A24/A32 Memory Space      :<memspace><CRLF>
A16/A24/A32 Memory Base       :<membase><CRLF>
A16/A24/A32 Memory Size       :<memsize><CRLF>
Slot                          :<slot><CRLF>
Slot 0 Logical Address        :<slot0><CRLF>
Extended Subclass             :<ext><CRLF>
Attribute                     :<attr><CRLF>
Manufacturer Specific Comment :<com><CRLF>
```

`la` device's logical address

`manid` manufacturer's ID number

`devclass` device class; the following values may be used:

REG = Register-Based device
MSG = Message-Based device
EXT = Extended-Class device
MEM = Memory-Based device

`memspace` memory space requirement

A16
A16/A24
A16/A32

`membase` memory-based address for A16, A24, A32

	"HHHH, HHHHHH, HHHHHHHH"
`memsize`	memory sizes for A16, A24, A32
	"HHHH, HHHHHH, HHHHHHHH"
`slot`	slot number (-1 if unknown)
`slot0`	slot 0 Logical Address (-1 if unknown)
`ext`	extended device's subclass
`attr`	memory device's attributes
`com`	not used, always `" "`

DLAD?

Purpose: Get a list of the known logical addresses.

Query Syntax: `DLAD?`

Response: Program response:

`<la1>,<la2>,..., <laN>`<CRLF>

where `<la1>` through `<laN>` are the known logical addresses.

Console response:

`Known logical addresses are <la1>,<la2>,..., <laN>`<CRLF>

CI logical addresses are terminated with an asterisk (`*`) in the console mode response.

Example: Get a list of the known logical addresses.

```
DLAD?
```

DNUM?

Purpose: Get the number of the known logical addresses.

Query Syntax: `DNUM?`

Response: Program response:

`<num las>`<CRLF>

where `<num las>` is the number of known logical addresses.

Console response:

`There are <num las> known Logical Addresses.`<CRLF>

Example: Get the number of the known logical addresses.

```
DNUM?
```

DRES?

Purpose: Perform a soft-reset sequence on a device.

Query Syntax: `DRES? <logical address> [, <sysfail flag>]`

Note: If the device stays failed for five seconds after the soft-reset sequence, `<sysfail flag>` determines whether or not the device is kept sysfail-inhibited.

Response: Program response:

`<status>`<CRLF>

Console response:

`Logical Address <logical address> is <status>.  SYSFAIL Inhibit is <state>.`<CRLF>

where `<status>` is one of the following:

PASS
FAIL

IFAIL
READY

and <state> is one of the following:

ON
OFF

Example: Soft-reset device at logical address 3.

```
DRES? 3
```

RREG?

Purpose: Read a 16-bit VXI register from a device.

Query Syntax:

```
RREG? <logical address>, <reg offset>
```

where <logical address> is the device to read from and <reg offset> is the number of bytes to offset from the base of the VXI registers for that device.

Response: Program response:

```
<hex word value><CRLF>
```

Console response:

```
Value 0x<hex word value> (<word value>) read from Logical Address
<logical address>, Register offset 0x<reg offset><CRLF>
```

Example: Read Device Type register from logical address 12.

```
RREG? 12,2
```

WREG

Purpose: Write a 16-bit VXI register on a particular device.

Query Syntax: `WREG <logical address>, <reg offset>, <value>`

where `<logical address>` is the device to write, `<reg offset>` is the register offset to write to, and `<value>` is the 16-bit value to write.

Action: Write `<value>` to `<logical address>`, register offset `<reg offset>`.

Example: Write the Data Low register for logical address 4 with the value 65535.

```
WREG 4,14,65535
```

GPIB Address Configuration Commands and Queries

The GPIB address configuration commands are described on the following pages.

- `LaSaddr`
- `LaSaddr?`
- `Primary?`
- `SaddrLa?`
- `Saddrs?`
- `SaDisCon`

These commands and queries configure and report the relationships between VXI logical addresses and GPIB addresses.

The GPIB-VXI's primary address can be determined by using the `Primary?` query from the serial port. The relationships between GPIB secondary addresses and VXI logical addresses can be determined by using the `Saddrs?` query followed by `SaddrLa?` queries, or by using the RM information query `Laddrs?` followed by `LaSaddr?` queries.

Secondary address links to Message-Based servants of the GPIB-VXI can be assigned with the `LaSaddr` command. The `SaDisCon` command deletes all secondary address links except the link to the GPIB-VXI local commands.

LaSaddr

Purpose: Attach or detach a secondary address to a logical address.

Command Syntax: `LaSaddr <logical address>, <secondary address>`

Action: If `<secondary address>` is not equal to 255, attach `<secondary address>` to <logical address>.

If `<secondary address>` is equal to 255, release `<secondary address>` currently attached to `<logical address>`.

Attaching a secondary address to a logical address that already has a secondary address will cause the first secondary address to be replaced by the new secondary address.

Attempting to release or change a secondary address will result in a Delete I/O Link error if any of the following conditions is true:

- The secondary address does not exist.
- The secondary address is addressed to talk or listen.
- There is still data in the secondary address input or output queue.

Examples: Attach secondary address 6 to logical address 4.

```
LaSaddr 4,6
```

Release secondary address currently attached to logical address 8.

```
LaSaddr 8,255
```

LaSaddr?

Purpose: Get the secondary address attached to a logical address.

Query Syntax: `LaSaddr? <logical address>`

Response: Program response:

`<secondary address><CRLF>`

where `<secondary address>` is the secondary address attached to the logical address. A value of 255 indicates that no secondary address is attached to the logical address.

Console response:

```
Logical Address <logical address> is attached to GPIB
Secondary Address <secondary address><CRLF>
```

for logical addresses with attached secondary addresses, or

```
Logical Address <logical address> is NOT attached to a
GPIB Secondary Address<CRLF>
```

for logical addresses without attached secondary addresses.

Example: Get the secondary address attached to logical address 9.

```
LaSaddr? 9
```

Primary?

Purpose: Get a GPIB primary address.

Query Syntax: `Primary?`

Response: Program response:

```
<primary address><CRLF>
```

where `<primary address>` is the GPIB primary address of GPIB-VXI.

Console response:

```
The GPIB primary address of the GPIB-VXI is <primary
address><CRLF>
```

SaddrLa?

Purpose: Get the logical address that a secondary address is attached to.

Query Syntax: `SaddrLa? <secondary address>`

Response: Program response:

```
<logical address><CRLF>
```

where <logical address> is the logical address that the secondary address is attached to. A value of 255 indicates that the secondary address is not attached to a logical address.

Console response:

```
GPIB Secondary Address <secondary address> is attached
to Logical Address <logical address><CRLF>
```

for a secondary address that is attached to a logical address, or

```
GPIB Secondary Address <secondary address> is NOT
attached to a Logical Address<CRLF>
```

for a secondary address that is not attached to any logical address.

Example: Get the logical address attached to secondary address 9.

```
SaddrLa? 9
```

Saddrs?

Purpose: Get a list of used secondary addresses.

Query Syntax:

```
Saddrs?
```

Response: Program response:

```
<sa1>,<sa2>, . . .,<saN><CRLF>
```

where <sa1> through <sa*N*> are the secondary addresses currently attached to logical addresses.

Console response:

```
Current Secondary Addresses used:
Secondary Address <sa1>:  connected to Logical Address <la1>.
Secondary Address <sa2>:  connected to Logical Address <la2>.
      •
      •
Secondary Address <saN>:  connected to Logical Address
<laN><CRLF>
```

SaDisCon

Purpose: Detach *all* secondary address links except the secondary address link to the GPIB-VXI command set.

Command Syntax: `SaDisCon`

Action: Detaches all secondary address links from servants of the GPIB-VXI.

VXIbus Interrupt Handler Configuration Commands and Queries

The interrupt handler configuration commands and queries are described on the following pages.

- `AllHandlers?`
- `AssgnHndlr`
- `HandlerLine?`
- `RdHandlers?`

The interrupt handler commands and queries configure and report the relationships between the GPIB-VXI interrupt handlers and VXIbus interrupt levels.

The GPIB-VXI has three programmable interrupter handlers. An application program can confirm this with the `RdHandlers?` query. The `AllHandlers?` and `HandlerLine?` queries return the current VXI interrupt level assignments for the handlers. The `AssgnHndlr` command can be used to change the level assignments.

AllHandlers?

Purpose: Get the VXIbus interrupt level assigned to all GPIB-VXI interrupt handlers.

Query Syntax: `AllHandlers?`

Response: Program response:

`<level1>,<level2>,<level3>`<CRLF>

where `<level1>` is the interrupt level assigned to handler 1, `<level2>` is the interrupt level assigned to handler 2, and `<level3>` is the interrupt level assigned to handler 3.

If <level*N*> equals 0, then interrupt handler <handler*N*> is not assigned to an interrupt level.

Console response:

```
VXI interrupt Handler 1 assigned to interrupt level
<level1><CRLF>
VXI interrupt Handler 2 assigned to interrupt level
<level2><CRLF>
VXI interrupt Handler 3 assigned to interrupt level
<level3><CRLF>
```

if all handlers are assigned to levels, or

```
VXI interrupt Handler <handler> NOT assigned to any interrupt
level.<CRLF>
```

if <handler*N*> is not assigned to a level.

Example: Get the interrupt level assigned to all interrupt handlers.

```
AllHandlers?
```

AssgnHndlr

Purpose: Assign a VXIbus interrupt level to a GPIB-VXI interrupt handler.

Command Syntax: `AssgnHndlr <handler>, <level>`

where <handler> is a numeric integer quantity in the range 1 to 3, and <level> is a numeric integer quantity in the range 0 to 7.

Action: If <level> is in the range 1 to 7, VXIbus interrupt line <level> is assigned to interrupt handler <handler>.

If <level> is 0, then the current VXIbus interrupt line held by interrupt handler <handler> is released.

Examples: Assign the interrupt level 6 to the GPIB-VXI interrupt handler 2.

```
AssgnHndlr 2,6
```

Release the interrupt level currently held by the GPIB-VXI interrupt handler 1.

```
AssgnHndlr 1,0
```

HandlerLine?

Purpose: Get the level assigned to a GPIB-VXI interrupt handler.

Query Syntax: `HandlerLine? <handler>`

Response: Program response:

`<level>`<CRLF>

Console response:

`VXI interrupt handler <handler> assigned to interrupt level <level>`<CRLF>

`<level>` is the interrupt level assigned to handler `<handler>`. If `<level>` equals 0, then the interrupt handler `<handler>` is not assigned an interrupt level.

Example: Get the interrupt level assigned to interrupt handler 3.

```
HandlerLine? 3
```

RdHandlers?

Purpose: Get the number of assignable GPIB-VXI interrupt handlers.

Query Syntax: `RdHandlers?`

Response: Program response:

`3` <CRLF>

Console response:

`This GPIB-VXI has 3 configurable VXI interrupt handlers.`<CRLF>

Example: Get the number of assignable GPIB-VXI interrupt handlers.

```
RdHandlers?
```

IEEE-488.2 Common Commands and Queries

The IEEE-488.2 commands and queries are as follows:

- `*CLS`
- `*ESE`
- `*ESE?`
- `*ESR?`
- `*IDN?`
- `*OPC`
- `*OPC?`
- `*RST`
- `*SRE`
- `*SRE?`
- `*STB?`
- `*TRG`
- `*TST?`
- `*WAI`

These commands provide minimal conformance to the 488.2 requirements for a DT1 device. Many of these 488.2 commands have limited meaning in the VXI environment, but are included for compatibility.

*CLS

488.2
Intent: Clear the device status data structures, and force it to the Operation Complete Query Idle state.

Command Syntax: `*CLS`

Action: None.

---

*ESE

488.2
Intent: Set the GPIB-VXI's Standard Event Status Enable (ESE) Register bits.

Command Syntax: `*ESE <byte value>`

where `<byte value>` is the new value of the ESE register.

Action: Sets ESE to `<byte value>`.

Example: Set the ESE register to 45.

```
*ESE 45
```

---

*ESE?

488.2
Intent: Get the contents of the ESE Register.

Query Syntax: `*ESE?`

Response: `<ESE val>`<CRLF>

where `<ESE val>` is the current value of the ESE register. The default value is FFh.

---

*ESR?

488.2
Intent: Read and clear the Standard Event Status Register (ESR).

Query Syntax: `*ESR?`

Response: `<ESR val>`<CRLF>

`<ESR val>` is the current value of the ESR.

---

*IDN?

488.2
Intent: Get the GPIB-VXI's manufacturer, model, serial number, and firmware level.

Query
Syntax: `*IDN?`

Response: `"National Instruments","GPIB-VXI",<serial number>,<firmware version>`<CRLF>

---

*OPC

488.2
Intent: Cause the GPIB-VXI to generate the operation complete message in the ESR when all pending selected device operations have been finished.

Command
Syntax: `*OPC`

Action: None.

Notice that since the GPIB-VXI only parses and routes commands, there are never any pending commands on the GPIB-VXI.

---

*OPC?

488.2
Intent: Cause the GPIB-VXI to place an ASCII 1 in its output queue when all pending operations have completed.

Query
Syntax: `*OPC?`

Response: `1` <CRLF>

---

*RST

488.2
Intent: Return a device to a known initial state.

Command Syntax: `*RST`

Action: None.

Other than the response mode configuration, the GPIB-VXI does not depart from its initial state.

---

*SRE

488.2
Intent: Set the device's Service Request Enable (SRE) Register bits.

Command Syntax: `*SRE <byte value>`

where `<byte value>` is the new value of the SRE register.

Action: Sets the SRE to `<byte value>`.

Example: Set the SRE register to 120.

```
*SRE 120
```

---

*SRE?

488.2
Intent: Get the contents of the SRE Register.

Query Syntax: `*SRE?`

Response: `<SRE val>`<CRLF>

`<SRE val>` is the current value of the SRE Register. The default value is FFh.

---

*STB?

488.2
Intent: Get the contents of a device's Status Byte.

Query
Syntax: `*STB?`

Response: `<STB value><CRLF>`

where `<STB value>` is the current status of the path to the GPIB-VXI local command parser.

---

*TRG

488.2
Intent: Cause a device to execute a stored trigger sequence.

Command
Syntax: `*TRG`

Action: None.

---

*TST?

488.2
Intent: Perform self-test and return passed or failed status.

Query
Syntax: `*TST?`

Response: `0 <CRLF>`

Failure to complete the self-test is indicated by a failure to respond to this query. If the response is received, the self-test was successful.

---

*WAI

488.2
Intent: Prevent device from executing any further commands until the No-Operation Pending flag is TRUE.

Command Syntax: `*WAI`

Action: None.

VXIbus Access Commands and Queries

The VXIbus access commands and queries are described on the following pages.

- `A16`
- `A16?`
- `A24`
- `A24?`
- `SYSRESET`

The `A16` and `A24` commands can be used to *poke*, or write, locations in VME A16 and A24 memory space. The `A16?` and `A24?` queries can be used to *peek*, or read, locations in VME A16 and A24 memory space.

The `SYSRESET` command can be used to remotely reset the system.

A16

Purpose: Write a 16-bit value into VXI A16 space.

Command Syntax: `A16 <A16 address>, <word value>`

Action: Write `<word value>` to `<A16 address>`.

Example: Write A502h to VXI A16 address 4305h.

```
A16 #h4305, #hA502
```

A16?

Purpose: Read word value from VXI A16 address space.

Query Syntax: `A16?  <A16 address>`

Response: Program response:

`<word value>`<CRLF>

Console response:

`Value <hex word value> (<word value>) read from A16 address <A16 hex address> (<A16 address>)`<CRLF>

where `<word value>` is in decimal integer format, `<hex word value>` is in C language hexadecimal format, `<A16 hex address>` is in C language hexadecimal format, and `<A16 address>` is in decimal integer format.

Example: Read the ID register of logical address 16.

```
A16? #hc400
```

A24

Purpose: Write a 16-bit value into VXI A24 space.

Command Syntax: `A24  <A24 address>, <word value>`

Notice that `<A24 address>` has a valid range of 2097152 to 14680062 (`#h200000` to `#hDFFFFE`).

Action: Write `<word value>` to `<A24 address>`.

Example: Write the value A502h to VXI A24 address 504305h.

```
A24  #h504305, #hA502
```

A24?

Purpose: Read a word value from VXI A24 address space.

Query Syntax: `A24?  <A24 address>`

Response: Program response:

`<word value>`<CRLF>

Console response:

```
Value <hex word value> (<word value>) read from A24 address <A42
hex address (<A24 address>)<CRLF>
```

where `<word value>` is in decimal integer format, `<hex word value>` is in C language hexadecimal format, `<A24 hex address>` is in C language hexadecimal format, and `<A24 address>` is in decimal integer format.

Example: Read the word at A24 address 205634h.

```
A24?  #h205634
```

SYSRESET

Purpose: Remotely reset system.

Command Syntax: `SysReset`

Action: Asserts the VME backplane signal SYSRESET.

TTL Trigger Access Commands

The TTL Trigger Access commands are described on the following pages.

- `SetTrigOutFP`
- `SetTrigSrc`
- `SourceTrig`

These commands can be used to directly manipulate the VXI TTL Trigger lines and the front panel Trigger connectors of the GPIB-VXI.

The `SetTrigSrc` command is used to set up the trigger line and protocol to use. `SetTrigOutFP` routes sourced triggers out the GPIB-VXI's front panel. `SourceTrig` is used to generate a TTL trigger.

SetTrigOutFP

Purpose: Set up GPIB-VXI trigger output on the front panel.

Command Syntax: `SetTrigOutFP <enable>`

where `<enable>` is a boolean value.

Action: If `<enable>` is TRUE (1), the GPIB-VXI is set up to source Trigger out the front panel. If `<enable>` is FALSE (0), the GPIB-VXI Trigger output is disabled.

Example: `SetTrigOutFP 1`

SetTrigSrc

Purpose: Set up a trigger line to source on.

Command Syntax: `SetTrigSrc <enable>, <line>, <protocol>`

where `<enable>` is a boolean value, `<line>` is 0-7 corresponding to TTL lines 0-7, and `<protocol>` is 0-4 where:

0: External In from front panel
1: Start/Stop
2: Sync
3: Semi-Sync
4: Asynch

Action: GPIB-VXI is set up to source on TTL Trigger line `<line>` using protocol `<protocol>`.

Example: Set up to source Sync protocol on TTL line 3.

```
SetTrigSrc 1, 3, 2
```

SourceTrig

Purpose: Source a TTL trigger.

Command Syntax: `SourceTrig`

Action: Source a TTL trigger on VXIbus backplane and/or our front panel depending upon current configuration by `SetTrigSrc` and `SetTrigOutFP` commands.

Example: `SourceTrig`

Word Serial Communication Commands and Queries

The Word Serial communication commands and queries are described on the following pages.

- `ProtErr?`
- `RespReg?`
- `WScmd`
- `WScmd?`
- `WSresp?`
- `WSstr`
- `WSstr?`

These commands can be used to directly generate Word Serial communication operations with any Message-Based device, including the GPIB-VXI itself, regardless of whether or not it is the GPIB-VXI's servant.

Note: The Word Serial communication commands and queries are intended for debugging purposes. National Instruments does not guarantee that these commands will work when other Word Serial paths are open, such as the GPIB Secondary Address link.

Some of the Word Serial commands as defined in the VXIbus specification require a response from the Message-Based device, while other commands do not. To distinguish between the two types of Word Serial commands, and to avoid confusion between Word Serial commands and GPIB-VXI local commands and queries, the following terminology will be used in this section:

Word Serial command–A VXI-defined Word Serial command that does not require a response from the Message-Based device.

Word Serial query–A VXI-defined Word Serial command that requires a response from the Message-Based device.

Command–A GPIB-VXI command, as defined in this chapter.

Query–A GPIB-VXI query, as defined in this chapter.

The `WScmd` command is used to send a Word Serial command to a Message-Based device. The `WScmd?` query is used to send a Word Serial query to the Message-Based device, and to automatically read and return the device's response.

`WScmd` can also be used to send a Word Serial query to a Message-Based device. Because `WScmd` does not read the query response, the intermediate state of the device can be examined using the `RespReg?` query, after which the response can be read using the `WSresp?` query.

The WSstr command can be used to send device-dependent commands and queries to a device. If the string sent to the device was a device-dependent query, the WSstr? query can be used to read the device's response.

The ProtErr? query sends a *Read Protocol Error* Word Serial query to a device and reports the error response. The RespReg? query returns the value of a device's response register.

ProtErr?

Purpose: Send a *Read Protocol Error* Word Serial query to a Message-Based device.

Query Syntax: `ProtErr? <log addr>`

Action: *Read Protocol Error* query is sent to a Message-Based device. Response is read and reported.

Response: Program response:

`<hex value><CRLF>`

where `<hex value>` is the hexadecimal value of the Data Low Register response.

Console response:

```
Read Protocol Error for Logical Address <log addr> returned 0x<hex
value>:
  <description>
```

where `<description>` is text explaining the error response.

Example: `ProtErr? 3`

RespReg?

Purpose: Get the Response Register contents of a Message-Based device.

Query Syntax: `RespReg? <log addr>`

Action: Returns the contents of the device's Response Register at logical address `<log addr>`.

Response: Program response:

```
<hex value><CRLF>
```

where <hex value> is the hexadecimal value of the Response Register contents.

Console response:

```
Logical Address   <log addr>'s Response register:<CRLF>
 [0x<hex value>]: <dor> <dir> <err> <rr> <wr> <fhs>
<locked><CRLF>
```

where <dor>, <dir>, <err>, <rr>, <wr>, <fhs>, and <locked> are text flags that interpret the state of the Response Register bit flags. Capitalized text in a text flag indicates that the corresponding bit flag is in the logic TRUE state. Lowercase text indicates that the corresponding bit flag is in the logic FALSE state.

WScmd

Purpose: Send a 16-bit Word Serial command or query to a Message-Based device.

Command Syntax: `WScmd  <log addr>, <WS cmd>`

Action: Sends the Word Serial command <WS cmd> to the device at <log addr>.

Example: Write the *Begin Normal Operation* Word Serial query (FCFFh) to a device at logical address 3.

```
WScmd 3, #hFCFF
```

WScmd?

Purpose: Send a 16-bit Word Serial query to a Message-Based device.

Query Syntax: `WScmd? <log addr>, <WS cmd>`

Action: Sends the Word Serial query `<WS cmd>` to the device at `<log addr>`. Reads and returns the device's response.

Response: Program response:

`<hex value>`<CRLF>

where `<hex value>` is the hexadecimal value of the Data Low Register response.

Console response:

```
Logical Address <log addr> Word Serial Query 0xceff returned
0x<hex value>.<CRLF>
```

Example: Write the *Read Servant Area* Word Serial query (CEFFh) to a device at logical address 4.

```
WScmd? 4, #hCEFF
```

WSresp?

Purpose: Read a 16-bit Word Serial response to a previously sent query.

Query Syntax: `WSresp? <log addr>`

Action: Reads and returns the response of the device at `<log addr>`.

Response: Program response:

`<hex value>`<CRLF>

where `<hex value>` is the hexadecimal value of the Data Low Register response.

Console response:

```
Logical Address <log addr> returned response 0x<hex value><CRLF>
```

Example: Read the 16-bit response to a previously sent Word Serial query from logical address 3.

```
WSresp? 3
```

WSstr

Purpose: Send a device-dependent command string to a Message-Based device.

Command Syntax:

```
WSstr  <log addr>, <string>
```

where `<string>` is an ASCII character sequence enclosed by double quotation marks (`"`).

The following sequences of characters within the `<string>` parameter are special cases and will be interpreted as follows:

`\n`	line feed (LF)
`\r`	carriage return (CR)
`\\`	backslash (\)
`\xHH`	any binary 8-bit value where `HH` is the ASCII hexadecimal representation of that value

Action: Writes the string `<string>` to the device at `<log addr>` as a series of *Byte Available* commands.

Example: Write the string `"start"` to a device at logical address 8.

```
WSstr  8, "start"
```

WSstr?

Purpose: Read a device-dependent response string from a Message-Based device.

Query Syntax: `WSstr? <log addr>, <max cnt>`

Action: Reads and returns a string, up to a maximum character count of `<max cnt>`, using a series of *Byte Request* commands.

Response: Program response:

```
<resp string>
```

where `<resp string>` is the response string returned by the device.

Console response:

```
Logical address <log addr> read <# bytes> (<hex # bytes>) through
word serial: <CRLF><CRLF> <resp string>
```

where `<# bytes>` and `<hex # bytes>` are the number of bytes in `<resp string>`, in decimal and hexadecimal, respectively.

Example: Read a device-dependent response up to 20 characters long from a device at logical address 10.

```
WSstr? 10, 20
```

CI Configuration Commands and Queries

The CI configuration commands and queries are described on the following pages.

- `CIAddr?`
- `CIArea`
- `CIArea?`
- `CIBlocks?`
- `CIDelete?`
- `CIList?`
- `DCIDownLdPI`
- `DCIDownLoad`

- DCISetup?
- DCISetupPI?

These commands and queries manipulate CIs and their related resources, and extract information about the CI configuration.

The query CIList? returns the list of code instrument logical addresses. The RM information queries that access information for physical devices (Cmdr?, RmEntry?, Srvnts?, StatusState?) can be used to retrieve the equivalent information for a CI. The CIDelete query deletes a CI.

The amount of RAM reserved for all CIs is set by the GPIB-VXI, depending upon its nonvolatile configuration, the amount of RAM installed, and the use of the command CIArea. The CI RAM area is partitioned into blocks of 4K. The current location and size of the CI RAM area can be determined by using the CIArea? query. The CIBlocks? query returns the allocation state of each block in the CI RAM area. The base address of a particular CI's RAM area can be determined by using the CIAddr? query.

Static Downloaded CIs (DCIs) are downloaded to the GPIB-VXI and initialized with the local commands DCISetUp? and DCIDownLoad.

Position Independent DCIs are downloaded to the GPIB-VXI and initialized with the local commands DCISetupPI? and DCIDownLdPI.

CIAddr?

Purpose: Get the local base RAM address of a CI.

Query Syntax: `CIAddr? <logical address>`

Response: Program response:

`<base address>`<CRLF>

where `<base address>` is the CI's base address in decimal.

Console response:

`CI at Logical Address <logical address> base Local Address is <hex base address>`<CRLF>

where `<hex base address>` is the CI's base address in C language hexadecimal notation.

Example: Get the local address of the CI at logical address 9.

```
CIAddr? 9
```

CIArea

Purpose: Change the location and size of the CI RAM area.

Command Syntax: `CIArea <Base Address>, <Number of blocks>`

Action: Sets the CI global RAM to start at `<Base Address>`, and span `<Number of blocks>` blocks of 4096 bytes each.

The default base address and size of the CI RAM area are set by the nonvolatile configuration parameters `CI Block Base` and `CI Num Blocks`.

`<Base Address>` is the new base address of the CI RAM area. It must be a multiple of 4096 decimal (1000h), and must be in the region above the top of pSOS Region 1 and below the top of memory. pSOS Region 1 starts at 10000h, and its size is determined by the nonvolatile configuration parameter `Region 1 Size`. For example, if `Region 1 Size` = 60000h, then the lowest allowed value for `<Base Address>` is as follows:

10000h + 600000h = 70000h

`<Number of blocks>` is the number of 4096 (1000h) byte blocks in the CI RAM area. The size of the CI RAM area is limited by the amount of physical RAM on the GPIB-VXI, so the maximum allowed value for `<Number of blocks>` is as follows:

(`<RAM size>` - `<Base Address>`) / 1000h

For example, if the GPIB-VXI is configured with 512K (80000h) of RAM, and `<New Base Address>` is 70000h, the maximum allowed value for `<Number of blocks>` is given by the following formula:

(80000h - 70000h)/ 1000h = 10h = 16

If `<Number of blocks>` is set to 0, CI's are disabled.

Example: Set the base of CI RAM area to 80000h, and the size to 128 blocks of 4K.

CIArea #h80000, 128

CIArea?

Purpose: Return the base address and size of CI global memory area.

Query Syntax: `CIArea?`

Response: Program response:

`<base address>, <number of blocks>`<CRLF>

where `<base address>` and `<number of blocks>` are the current base address and size of the CI RAM area in blocks of 4K, respectively.

Console response:

`CI Global Base is local Address <hex base address> with <number of blocks> 4K blocks`<CRLF>

`<hex base address>` is the base address of the CI RAM area in C language hexadecimal notation. `<number of blocks>` is the size of the CI RAM area (in blocks of 4K) in decimal.

CIBlocks?

Purpose: Return a listing of used and unused CI memory area blocks.

Query Syntax: `CIBlocks?`

Response: Program response:

`<b0>, <b1>, . . ., <bL-1>`<CRLF>

where `<bJ>` is a boolean value that indicates whether the Jth block is unused (0) or used (1). `L` is the number of blocks of 4K in the CI global memory area.

Console response:

`Blocks Used of the <L> Blocks of CI Global Memory:` <CRLF>
`<r0start> - <r0stop>, <r1start> - <r1stop>, . . .,<r`*N*`-1start> - <r`*N*`-1stop>`<CRLF>

where `<rMstart>` and `<rMstop>` are the start and stop block numbers for the Mth occupied memory region.

CIDelete?

Purpose: Delete a CI.

Query Syntax: `CIDelete? <code instrument logical address>`

`<code instrument logical address>` is the logical address of the CI to be deleted.

Response: Program response:

`<error code>`<CRLF>

where `<error code>` is a decimal value that indicates the result of the attempt to delete the CI. If `<error code>` is equal to 0, the attempt was successful.

Console response:

```
Code Instrument at Logical Address <code instrument logical
address> successfully deleted<CRLF>
```

if the attempt was successful, or

```
Error Deleting Code Instrument (Error code = <hex error code>)
```

if the attempt was unsuccessful.

`<hex error code>` is a value in C language hexadecimal format that indicates the result of the attempt to delete the CI.

`<error code>` and `<hex error code>` can be interpreted by converting them to a binary bit pattern. A value of 1 in any bit position indicates that the error shown in the following table occurred during the attempt to delete the CI:

Bit	Error condition
0	The GPIB-VXI was unable to delete the CI's secondary address link.
1	The GPIB-VXI was unable to delete the CI's message exchange.
2	The GPIB-VXI was unable to delete the CI's Async process.
3	The GPIB-VXI was unable to delete the CI's Worker process.
4	The GPIB-VXI was unable to delete the CI's Word Serial I/O structures.
5	The GPIB-VXI was unable to free the PI CI's dynamic memory.

Any error encountered is unrecoverable in the sense that the CI is not restored. Any further attempts to communicate with it will have undetermined results, and may adversely affect the behavior of the GPIB-VXI.

CIList?

Purpose: Get a list of logical addresses for CIs running on the GPIB-VXI.

Query Syntax: `CIList?`

Response: Program response:

`<ci la1>,<ci la2>, . . .,<ci laN>`<CRLF>

where `<ci laJ>` is the logical address of the Jth local CI. *N* is the total number of local CIs.

Console response:

`Local CI Logical Addresses are: <ci la1>,<ci la2>, . . ., <ci laN>`<CRLF>

DCIDownLdPI

Purpose: Download a Position Independent (PI) CI to RAM and start running it.

Command Syntax: `DCIDownLdPI [<boolean>]`

Action: Bytes of code and data (up to the number requested in the `DCISetupPI?` command) are downloaded from the command source. When the download is complete, the pSOS processes associated with the PI DCI are initiated.

DCIs can only be downloaded from the GPIB port or via Word Serial Protocol, because the download is terminated on GPIB EOI or the Word Serial `END` command. Because there is no analogy for EOI or END for the serial port (that is, a carriage return is a valid binary number), it cannot be used to download PI DCIs.

If `<boolean>` is 1, debug statements are printed to the serial port during the various stages of PI DCI initialization. If `<boolean>` is 0 or if it is omitted, the debug statements are not output. The debug printing mode is only available with the development firmware option.

The `DCIDownLdPI` command should always be immediately preceded by a `DCISetupPI?` command that configures the download parameters. Executing intermediate GPIB-VXI commands between `DCISetupPI?` and `DCIDownLdPI` may invalidate the download setup.

Example: Download and initialize a PI DCI, generating debug statements.

```
DCIDownLdPI 1
```

DCIDownLoad

Purpose: Download a CI to RAM and start running it.

Command Syntax: `DCIDownLoad [<boolean>]`

Action: Blocks of code and data (up to the number requested in the `DCISetup?` command) are downloaded from the command source. When the download is complete, the pSOS processes associated with the DCI are initiated.

DCIs can only be downloaded from the GPIB port or via Word Serial protocol, because the download is terminated on GPIB EOI or the Word Serial `END` command. Because there is no analogy for EOI or END for the serial port, it cannot be used to download DCIs.

If `<boolean>` is 1, debug statements are printed to the serial port during the various stages of DCI initialization. If `<boolean>` is 0 or if it is omitted, the debug statements are not output. The debug printing mode is only available with the development firmware option.

The `DCIDownLoad` command should always be immediately preceded by a `DCISetup?` command that configures the download parameters. Executing intermediate GPIB-VXI commands between `DCISetup?` and `DCIDownLoad` may invalidate the download setup.

Example: Download and initialize a DCI, generating debug statements.

```
DCIDownLoad 1
```

DCISetup?

Purpose: Setup parameters for a DCI download.

Query Syntax:

```
DCISetup? <logical address>, <commander's logical
address>, <start block>, <number of blocks>, <stack
size>, [, <servant1>, [<servant2>,..., <servantN>]]
```

The `DCISetup?` query provides the GPIB-VXI with the information it needs to prepare for executing a `DCIDownLoad` command. This command is provided as a separate operation from the `DCIDownLoad` command so that the download parameters can be validated before the object code download is initiated.

The GPIB-VXI interprets the `DCISetup?` query parameters as follows:

- The DCI is to be assigned logical address `<logical address>`, and granted to the commander at `<commander's logical address>` as a servant.
- Up to `<number of blocks>` of DCI code and data are to be loaded into CI RAM area starting at `<start block>`.
- A stack of size `<stack size>` words is to be allocated for the CI worker process. If `<stack size>` is less than 1024, a stack size of 1024 words (2K) is to be allocated for the CI.
- `<servant1>` through `<servantN>` are to be granted to the CI as servants.

Any GPIB secondary address links to `<servant1>`, `<servant2>` through `<servantN>` must be disconnected before the DCI is downloaded. The links may be deleted by using the `SaDisCon` or `LaSaddr` commands.

Response: Program response:

`0` <CRLF>

`0` is returned to report the successful completion of the `DCISetup?` command. Any errors are reported in the form of an error code.

Console response:

`DCI parameters OK, Ready to download.`<CRLF>

Example: Set up to download a DCI at logical address C0h, to be a servant of the device at logical address 2 and commander of device at logical address 50. Set up to download to the first 20 blocks of DCI memory area, and allocate a 2048-word stack.

DCISetup? #hC0,2,0,20,#h800,50

DCISetupPI?

Purpose: Set up parameters for a PI DCI download.

Query Syntax:

```
DCISetupPI? <logical address>, <commander's logical
address>, <dynamic RAM size>, <stack size>,
[, <servant1>, [<servant2>,..., <servantN>]]
```

The `DCISetupPI?` query provides the GPIB-VXI with the information it needs to prepare for executing a `DCIDownLdPI` command. This command is provided as a separate operation from the `DCIDownLdPI` command so that the download parameters can be validated before the object code download is initiated.

The GPIB-VXI interprets the `DCISetupPI?` query parameters as follows:

- The PI DCI is to be assigned logical address `<logical address>`, and granted to the commander at `<commander's logical address>` as a servant.
- Up to `<dynamic RAM size> bytes` of PI DCI code and data are to be loaded into a pSOS dynamic RAM segment allocated when the `DCIDownLoadPI` command is sent.
- A stack of size `<stack size>` words is to be allocated for the CI worker process. If `<stack size>` is less than 1024, a stack size of 1024 words (2K) is to be allocated for the CI.
- `<servant1>` through `<servant`*N*`>` are to be granted to the CI as servants.

Any GPIB secondary address links to `<servant1>`, `<servant2>` through `<servant`*N*`>` must be disconnected before the PI DCI is downloaded. The links can be deleted by using the `SaDisCon` or `LaSaddr` commands.

Response: Program response:

`0` <CRLF>

`0` is returned to report the successful completion of the `DCISetupPI?` command. Any errors are reported in the form of an error code.

Console response:

`PI DCI parameters OK, Ready to download.`<CRLF>

Example: Set up to download a PI DCI at logical address C0h, to be a servant of the device at logical address 2 and commander of device at logical address 50. Set up to download up to 10000 bytes of code and data to a pSOS dynamic memory segment, and allocate a 2048-word stack.

```
DCISetupPI?  #hC0,0,10000,#h800,50
```

Chapter 4
Nonvolatile Configuration

The GPIB-VXI nonvolatile (NV) memory is a 256-byte EEPROM that is accessible as 64 longword locations. The first half of the NV memory (32 longwords) is reserved for National Instruments usage. The second half of NV memory is allocated for storing Code Instrument (CI) configuration variables.

The NV memory holds configurable constants that are accessible from the onboard software. The configurable National Instruments-reserved configuration parameters include the following:

- pSOS configuration
- VXI interrupt line assignment
- Resource Manager (RM) A24 and A32 address assignment base
- Servant area size
- DC starting logical address
- Device failure mode
- GPIB configuration
- Default CI configuration
- CI RAM area configuration
- Resident CI locations

To enter NV configuration mode, set the startup mode switches to the nonvolatile configuration mode as described in the *GPIB-VXI Startup Mode Configuration* section of Chapter 2. Connect a terminal to the serial port as described in the *System Configuration* section of Chapter 2. The NV configuration mode can also be entered from pROBE (on development modules) through the `Conf` command. The NV memory can also be configured via the GPIB port through the pROBE `Conf` command, although this method is more difficult to use manually.

The EEPROM is connected to the microprocessor via a serial bus. Because it takes from five to ten seconds to write the contents of the memory, the GPIB-VXI creates a copy of the contents of the EEPROM in RAM, which can be quickly edited. When the editing is complete, the entire contents of the RAM copy can be written back at once to the EEPROM.

Notice that some of the changes (such as the pSOS parameters) do not take effect until the system is restarted. This can be accomplished by the pROBE commands `IN` or `BO`, by resetting the system, or by cycling the system power.

The GPIB-VXI Nonvolatile Configuration Main Menu

When you enter the NV configuration mode, the GPIB-VXI displays the menu shown in Figure 4-1.

```
   GPIB-VXI Nonvolatile Configuration Main Menu
            (C) 1989  National Instruments
================================================
 1). Read In Nonvolatile Configuration
 2). Print Configuration Information
 3). Change Configuration Information
 4). Set Configuration to Factory Settings
 5). Write Back (Save) Changes
 6). Quit Configuration

     Choice (1-6):
```

Figure 4-1. The GPIB-VXI Nonvolatile Configuration Main Menu

From the main menu, you can select the NV memory editing function you want to perform. To select an item in the menu, enter its number at the prompt. The effect of selecting each item is described below.

Read in Nonvolatile Configuration

The item `Read In Nonvolatile Configuration` reads the contents of the EEPROM into RAM.

Print Configuration Information

The item `Print Configuration Information` displays the Nonvolatile Configuration Information as shown in Figure 4-2.

```
=============== Nonvolatile Configuration Information ===============

Serial Number  : 0x00000056       Firmware Type  : User
Region 1 Size  : 0x060000         Number Procs   : 0x20
Number Exchgs  : 0x20             Number Msgs    : 0x180
VXI Interrupt Level to Handler Logical Address (0xFF = free to assign)
  1:0xFF  2:0xFF  3:0xFF  4:0xFF  5:0xFF  6:0xFF  7:0xFF
A24 Assign Base: 0x200000         A32 Assign Base: 0x20000000
DC Starting LA : 0x01             For FAILED Dev : DO set Reset Bit
OverRide Srvnt : NO               Servant Area   : N/A
OverRide Prim  : NO               GPIB Primary   : N/A
GPIB Sec Addr  : Default          GPIB Flags     : DMA 7210

CI Block Base  : 0x070000         CI Num Blocks  : 0x10

          ------ Resident Code Instrument Locations ------
0x00: 00000000            0x01: 00000000            0x02: 00000000
0x03: 00000000            0x04: 00000000            0x05: 00000000
0x06: 00000000            0x07: 00000000            0x08: 00000000
0x09: 00000000            0x0A: 00000000            0x0B: 00000000

     ------ CI Nonvolatile User Configuration Variables ------
0x00:00000000      0x01:00000000      0x02:00000000      0x03:00000000
0x04:00000000      0x05:00000000      0x06:00000000      0x07:00000000
0x08:00000000      0x09:00000000      0x0A:00000000      0x0B:00000000
0x0C:00000000      0x0D:00000000      0x0E:00000000      0x0F:00000000
0x10:00000000      0x11:00000000      0x12:00000000      0x13:00000000
0x14:00000000      0x15:00000000      0x16:00000000      0x17:00000000
0x18:00000000      0x19:00000000      0x1A:00000000      0x1B:00000000
0x1C:00000000      0x1D:00000000      0x1E:00000000      0x1F:00000000
```

Figure 4-2. The Nonvolatile Configuration Information Display

The first three sections display the National Instruments-reserved variables. The last section displays hexadecimal values representing the contents of the user-defined variables. In this example, no user-defined variables have been initialized.

Change Configuration Information

The item `Change Configuration Information` displays the GPIB-VXI Nonvolatile Configuration Changer as shown in Figure 4-3.

```
      GPIB-VXI Nonvolatile Configuration Changer
            (C) 1989  National Instruments
 ================================================
  1). Edit pSOS Configuration
  2). Edit Default VXI Interrupt Handler Levels
  3). Edit Resource Manager A24/A32 Assign Bases
  4). Edit Servant Area and DC Starting LA
  5)  Edit FAILED Device Handling Mode
  6). Edit GPIB Configuration
  7). Edit Default CI Configuration
  8). Edit Resident CI Base Locations
  9). Edit CI User Configuration Variables
  Q). Quit Editor

      Choice (1-9,Q):
```

Figure 4-3. The GPIB-VXI Nonvolatile Configuration Changer

You can edit the National Instruments-reserved configuration parameters and CI user configuration variables by selecting the corresponding menu item. In each case, you are prompted to enter constants for the new values, with default values supplied where appropriate. For the pSOS configuration parameters, the GPIB-VXI prints a formula for calculating an appropriate value for each parameter if you type in `0` in response to the prompt requesting the value.

The Default CI Configuration and Resident CI Base Locations options are only important when installing a Resident CI. Refer to Appendix D, *Using the CDS-852 Adapter Code Instrument*, for instructions on installing the Resident CIs.

Note: The `Change Configuration Information` editor only modifies the RAM copy of the NV memory contents. You must update the NV memory with the `Write Back (Save) Changes` command in the main menu to retain the changes after the GPIB-VXI has been reset or powered-down.

You can override the GPIB primary address and servant area size switch settings and the default secondary address assignment of the GPIB-VXI by editing the GPIB configuration and servant area size parameters.

Selecting `Quit Editor` returns the display to the main menu.

Set Configuration to Factory Settings

The item `Set Configuration to Factory Settings` sets the contents of the RAM copy of the NV memory to the default (original) factory settings. Notice that only the RAM copy is affected. The NV memory must be written back using the `Write Back (Save) Changes` command in the main menu to retain the changes after the GPIB-VXI has been reset or powered-down.

Write Back (Save) Changes

The item `Write Back (Save) Changes` writes the modified copy of the NV memory back to the EEPROM. The write-back procedure takes from five to ten seconds.

Quit Configuration

The item `Quit Configuration` prompts you to select a different startup configuration, or re-enters pROBE, depending upon how the diagnostics were entered.

Chapter 5
Diagnostic Tests

This chapter contains information for executing the GPIB-VXI diagnostic self-tests. The diagnostics test each GPIB-VXI sub-circuit, so they are useful in detecting and localizing problems.

To access the diagnostics mode, start the system with startup mode switches 8 and 7 set to *ON* and *OFF*, respectively, as described in the *GPIB-VXI Startup Mode Configuration* section of Chapter 2. Development modules can enter the diagnostics mode from pROBE using the pROBE `Diag` command. Diagnostic execution is controlled through a terminal connected to the serial port.

Configuration for Diagnostic Testing

The diagnostic tests require the GPIB-VXI to be disconnected from all other GPIB devices to prevent interference with the GPIB tests.

Diagnostic Test Structure

A total of 126 diagnostic routines, or *steps*, are organized as nine tests, as shown in Table 5-1. Each step is composed of one or more subroutines called *commands*.

Table 5-1. Diagnostic Tests

		Step Numbers	
Test Name	**Test Number**	**From**	**To**
EPROM	1	1	2
RAM	2	3	6
68070 CPU	3	7	21
VXI Configuration Registers	4	22	33
Local Interrupts	5	34	86
GPIB	6	40	86
DIP Switch and Trigger (Interactive)	7	113	114
DMA	8	115	125
68881 Coprocessor	9	126	126

Each test is designed to verify that a specific part of the GPIB-VXI circuitry is functioning correctly. The diagnostic steps can be invoked individually or as test groups.

Diagnostic Test Description

The EPROM Test

The EPROM test performs a checksum on the EPROM to verify that it is not corrupted.

The RAM Test

The RAM test performs tests on RAM to ensure that the CPU can correctly read and write from all RAM addresses.

The 68070 CPU Test

The 68070 CPU test performs tests on the 68070 I^2C interface, UART interface, and timer to determine that they are functioning properly.

The VXI Configuration Register Test

The VXI Configuration Register test performs tests on the VXI configuration registers to verify their content and functionality.

The Local Interrupt Test

The Local Interrupt test performs interrupt tests on the GPIB, Trigger, Write Data, Signal, Read Data and SYSFAIL local interrupts.

The GPIB Test

The GPIB test performs tests on the 7210 and Turbo488 GPIB interface ICs to ensure that they are functioning properly.

The DIP Switch and Trigger Test

The DIP Switch and Trigger test interactively tests the operation of the onboard DIP switches and trigger circuitry.

The DMA Test

The DMA test performs tests on DMA Channel 2 to ensure that DMA memory-to-memory transfers are functional.

The 68881 Coprocessor Test

The 68881 Coprocessor test tests the numeric coprocessor operation. If the 68881 is not installed, the GPIB-VXI skips this test.

Diagnostics Mode Selection

Three hierarchical levels of menus control execution of the diagnostic tests. The highest-level menu is the Diagnostics Mode menu, which you can use to select whether to execute a test group or step group, and the mode in which to run them. The Diagnostics Mode menu is shown in Figure 5-1 and described in Table 5-2.

```
GPIB-VXI DIAGNOSTICS:

       Default              ===> 0
       Stepping Steps       ===> 1
       Looping Tests        ===> 2
       Looping Steps        ===> 3
       Slow Tests           ===> 4
       Slow Steps           ===> 5
       Slow Looping Tests   ===> 6
       Slow Looping Steps   ===> 7
       Display Step Lines   ===> 8
       Display/Hold Lines   ===> 9
       Over Night Loop      ===> a
       Quit                 ===> q

    REPORT LAST ERROR       ===> r
    CLEAR LAST ERROR        ===> c
    PRINT TOGGLE            ===> p
    ABORT ERROR TOGGLE      ===> e
    (Current settings:  PRINT(ON), ABORT ERROR(ON) )

  Enter Selection    :
```

Figure 5-1. The Diagnostics Mode Menu

Table 5-2. Diagnostics Mode Menu Option Descriptions

Selection	**Description**
Default	Runs the selected subset of tests (1-9).
Stepping Steps	Runs the selected subset of steps (1-126).
Looping Tests	Runs the selected subset of tests (1-9) repeatedly.
Looping Steps	Runs the selected subset of steps (1-126) repeatedly.
Slow Tests	Runs the selected subset of tests (1-9) with a pause between the execution of each test.
Slow Steps	Runs the selected subset of steps (1-126) with a pause between the execution of each step.
Slow Looping Tests	Runs the selected subset of tests (1-9) repeatedly with a pause between the execution of each test.
Slow Looping Steps	Runs the selected subset of steps (1-9) repeatedly with a pause between the execution of each step.
Display Step Line	Runs the selected subset of steps (1-126) and displays the subroutine commands executed for each step.
Display/Hold Lines	Runs the selected subset of steps (1-126) and displays the subroutine commands executed for each step. With this mode you must input a keystroke to continue after each command is executed.
Over Night Loop	Runs the selected subset of steps (1-126) repeatedly until an error is encountered or the module is reset. If you select the `All` option from the second-level menu, all steps but the DIP switch and trigger steps are executed. Since the DIP switch and trigger steps are interactive, they are not included in the Over Night Loop tests.

By default, test result messages are printed to the serial port as each test is executed, and diagnostics execution terminates if an error occurs. Using the Diagnostics Mode menu, you can suppress test result message printing and the abort-on-error logic. Suppress printing with the `p` toggle selection, and the abort-on-error logic with the `e` toggle selection.

You can recall the last error at any time (before the module is powered-down or restarted) with the `r` selection. The last detected error is retained even if the diagnostics mode is exited and re-entered. This is a convenient feature that you can use under these conditions:

- Test result message printing is suppressed.

- Abort-on-error logic is suppressed.
- The terminal connected to the serial port is disconnected during diagnostic execution. For example, a PC is used with a terminal emulator to initiate the tests, but is used for other applications while the tests are executing.

You can clear the last logged error with the `c` selection.

Diagnostic Test Selection

When you select a diagnostics mode, the GPIB-VXI displays the Diagnostic Test Selection menu. This menu selects whether the whole set or a subset of the tests or steps are to be run.

```
GPIB-VXI DIAGNOSTICS:  TEST/STEP OPTIONS

      All        ===> 1
      Selection  ===> 2
      Quit       ===> q

Enter Your Selection   :
```

Figure 5-2. The Diagnostic Test Selection Menu

If you specify the `Selection` option, the GPIB-VXI prompts you to select which tests or steps to execute with the lowest level menu.

When you select the tests or steps, the GPIB-VXI begins executing the selected diagnostics. The diagnostics run until an error is encountered when abort-on-error is enabled; or until the indicated test or step subset has successfully completed.

If an error is encountered, you can repeat the offending step in the Display Step Lines mode to pinpoint the problem. Contact National Instruments for an interpretation of diagnostics mode error messages.

The `q` selection always returns to the next higher menu in the hierarchy. At the Diagnostics Mode menu, the diagnostics environment exits to the environment from which it was entered. If you entered the diagnostic environment from pROBE, it will re-enter pROBE. If the GPIB-VXI was powered-on or reset in the diagnostics startup mode, you will be prompted to select a different startup mode and to re-start the GPIB-VXI.

Appendix A
Specifications

Power Requirements

Source	Typical	Maximum
+5 VDC	4.7 A	9.5 A
+12 VDC	9.0 mA	15 mA
-12 VDC	8.0 mA	15 mA
-5.2 VDC	250 mA	400 mA
-2 VDC	50 mA	100 mA

Physical

C size VXIbus board

Board dimensions	9.187 in. by 13.386 in.
Front Panel connectors	One IEEE-488 connector One BNC connector for TTL trigger input One BNC connector for TTL trigger output One BNC external connector for 10-MHz clock One 9-pin DSUB connector for RS-232

Operating Environment

Component temperature	0° to 70° C
Relative humidity	0% to 95% noncondensing
Emissions	FCC Class A

Storage Environment

Temperature	-40° to 85° C
Relative humidity	0% to 100% noncondensing

Appendix B
Error Codes

Table B-1 describes the error associated with each local command set error code.

Table B-1. Error Codes

Error Number	Type	Description
0	Format	Command format error
1	Syntax	Command was not found
2	Syntax	Illegal identifier after <Program Data Separator>
3	Syntax	Missing <Program Data Separator>
4	Syntax	Maximum <Program Mnemonic> length is 12 characters
5	Syntax	Illegal command: Expecting upper or lower case alpha
6	Syntax	Illegal command
7	Syntax	Illegal non-numeric
8	Syntax	Illegal <Decimal Numeric Program Data>
9	Syntax	Illegal <Suffix Program Data>
10	Syntax	Maximum <Suffix Program Data> length is 12 characters
11	Syntax	Illegal <String Program Data>
12	Syntax	Illegal <Arbitrary Block Program Data>
13	Syntax	Illegal <Expression Program Data>
14	Syntax	Illegal <Character Program Data>
15	Syntax	Illegal character on input
16	Syntax	Illegal identifier after command
17	Syntax	Illegal identifier after <Program Separator>
18	Syntax	Missing <Program Separator>
19	Syntax	Too much data
30	Device	No error
31	Device	Logical address is out of range 0 through 254
32	Device	No device is at that logical address
33	Device	GPIB secondary address is out of range 0 through 30
34	Device	VXI interrupt handler number is out of range 1 through 3
35	Device	VXI interrupt level is out of range 0 through 7
36	Device	A16 address is out of range 0000h through FFFEh
37	Device	Address must be even
38	Device	Word write value is out of range 0000h through FFFFh
39	Device	A bus error occurred during the access
40	Device	A24 address is out of range 200000h through DFFFFEh
41	Device	488.2 register is out of range 0 through 255

(continues)

Table B-1. Error Codes (continues)

Error Number	Type	Description
42	Device	Console mode is disabled: must have one output mode enabled
43	Device	Logical device has no secondary address link
44	Device	Unable to delete secondary address link
45	Device	Unable to create secondary address link
46	Device	Secondary address is already attached to a logical address
47	Device	Device is not a Message-Based device
48	Device	Device is not a servant of this GPIB-VXI
49	Device	Device does not have commander capability
50	Device	Not Dynamically Configured
51	Device	Commander did not accept *Device Grant* command
52	Device	Servant did not accept BNO or *Identify Commander* command
53	Device	Logical address cannot be this GPIB-VXI
54	Device	Word Serial command is out of range 0 through FFFFh
55	Device	Logical address is not physical VXI device
56	Device	Unable to create I/O buffer
57	Device	Commander did not accept *Release Device* command
58	Device	Unable to grant CI to physical device
59	Device	Device is already a servant
60	Device	Device is not commander of servant
61	Device	Register offset out of range 0 through 3Eh
100	CI	DCI functionality is inactive
101	CI	Logical address conflict
102	CI	Logical address is out of range
103	CI	Block(s) requested are used
104	CI	Block(s) requested do not exist
105	CI	Servant(s) requested do not exist
106	CI	Servant(s) requested are not servants of the GPIB-VXI or another DCI
107	CI	Commander requested does not exist
108	CI	Servant(s) requested do not have the same commander
109	CI	Requested zero blocks
110	CI	Logical address referenced is not a DCI
111	CI	DCI base address is out of range
112	CI	DCI area new base address is not a multiple of 4096
113	CI	DCI area request exceeds available memory
114	CI	Attempted to change DCI area while DCIs were installed
116	CI	DCI was not found
117	CI	Logical address referenced is not a DCI
120	CI	Error encountered while spawning DCI process(es)
121	CI	Error encountered while creating Asynch process exchange
122	CI	No DCI initialization information (need to do `DCISetup?` query)
123	CI	Download timed out
125	CI	Download overflowed requested blocks
126	CI	Servant(s) requested has secondary address link

(continues)

Table B-1. Error Codes (continued)

Error Number	Type	Description
127	CI	Memory requested for DCI Word Serial structures is unavailable
128	CI	Logical address referenced is not the GPIB-VXI or local DCI
129	CI	Logical address referenced is not GPIB-VXI's or CI's servant
130	CI	Stack size requested for worker process exceeds FFFFh words

Appendix C
Code Instrument Overview

This appendix contains an overview of the functions, applications, and implementations of software modules known as Code Instruments (CIs), and presents comparisons and illustrations of GPIB-VXI operation with and without CIs.

CIs are a National Instruments GPIB-VXI proprietary feature. They are capable of a wide variety of application-specific translation and control functions. A CI is a set of software routines running on the GPIB-VXI, but the system sees a CI as a physical Message-Based device. As with physical Message-Based devices, an external controller can communicate directly with the CI via a GPIB secondary address.

CIs perform special functions in the VXI environment. Typical applications of CIs include the following:

- Parsing and interpreting command languages
- Creating virtual (hierarchical) instruments
- Creating a Message-Based interface for Register-Based or non-VXI devices

A CI is more than a CPU process that replaces another VXI device's communication path; it has all of the capabilities of a physical Message-Based commander. These capabilities include the following:

- Having servant(s) assigned to it
- Having Word Serial communication with its commander and servant(s)
- Handling VXI interrupts and signals
- Having communication with the GPIB System Controller through a secondary address
- Having bus mastership for direct access to Register-Based servants and non-VXI devices

CIs improve the system structure for the following reasons:

- GPIB traffic is greatly reduced.
- System performance is greatly increased.
- The System Controller can treat all devices as if they were Message-Based.

GPIB-VXI Operation without CIs

Typical commander/servant relationships for GPIB-VXI operation without CIs are illustrated in Figure C-1. A GPIB System Controller communicates with the local command set and the GPIB-VXI's Message-Based servants through GPIB secondary addresses. This is a complete interface solution for Message-Based devices. Although the GPIB and serial controllers are not commanders of the command parser in the VXI sense, they are its *master* in the sense that it will respond to their commands as if they were its commander. The GPIB-VXI maintains independent control paths to the local command set parser from the GPIB, the serial controller, and the GPIB-VXI's commander.

The GPIB-VXI has four ports for communicating with other devices. Each port consists of its electrical interface and the associated system software. The GPIB-VXI communicates with its commander through the Word Serial servant port, and with its servants through the Word Serial commander port. The GPIB System Controller communicates with the GPIB-VXI and its Message-Based servants through the GPIB port, which maps GPIB secondary addresses to VXI logical addresses. A serial controller can access the local command parser through the RS-232 port, and therefore can communicate with the GPIB-VXI's Message-Based servants through the Word Serial communication commands.

The System Controller can also directly control Register-Based and non-VXI devices through the VXIbus Access local commands. This solution to the general problem of controlling Register-Based and non-VXI devices is relatively ineffective for high-performance applications, however, because of the low-level functions that the System Controller must perform and the resulting heavy GPIB traffic.

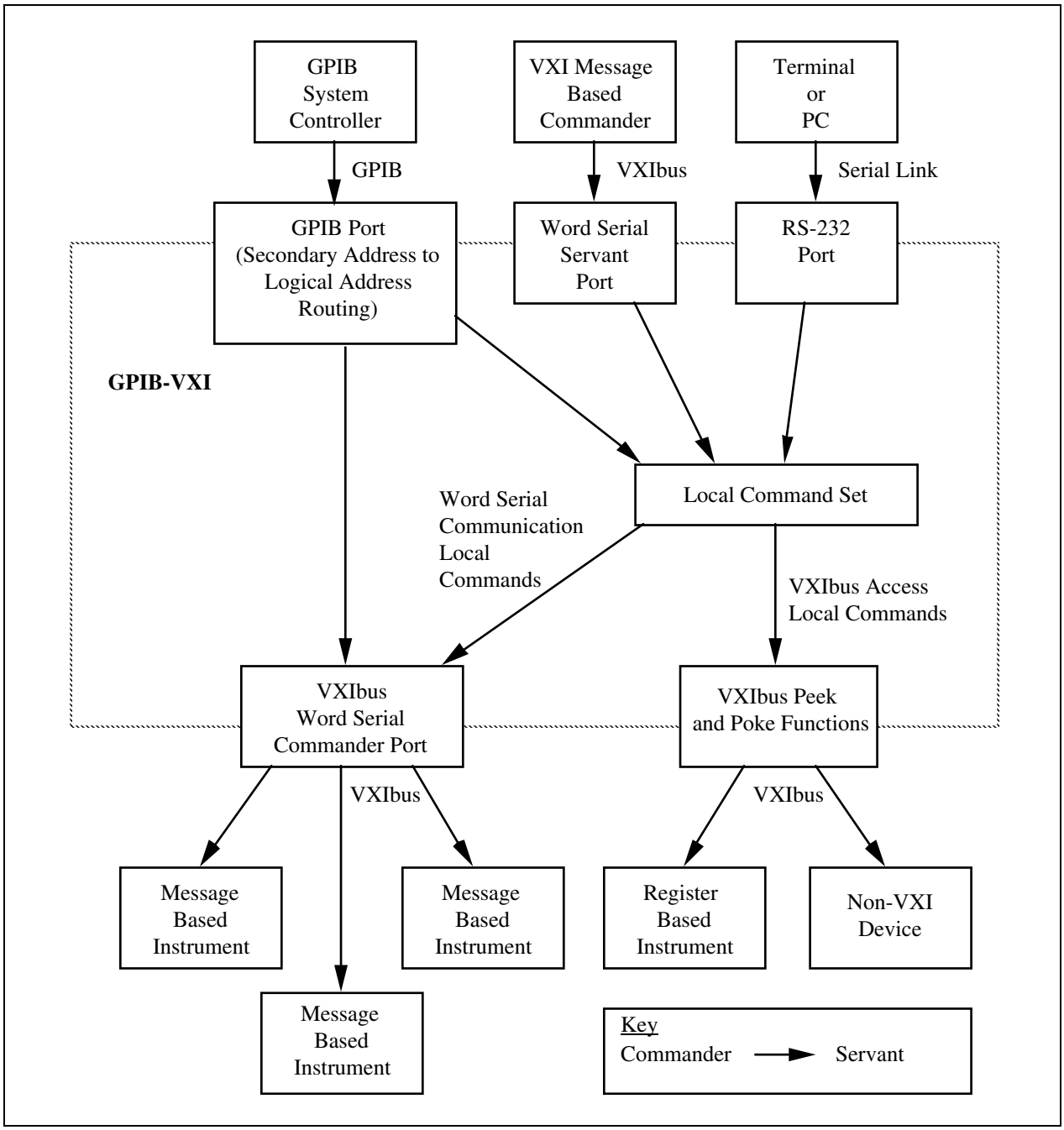


Figure C-1. GPIB-VXI Operation Without Code Instruments

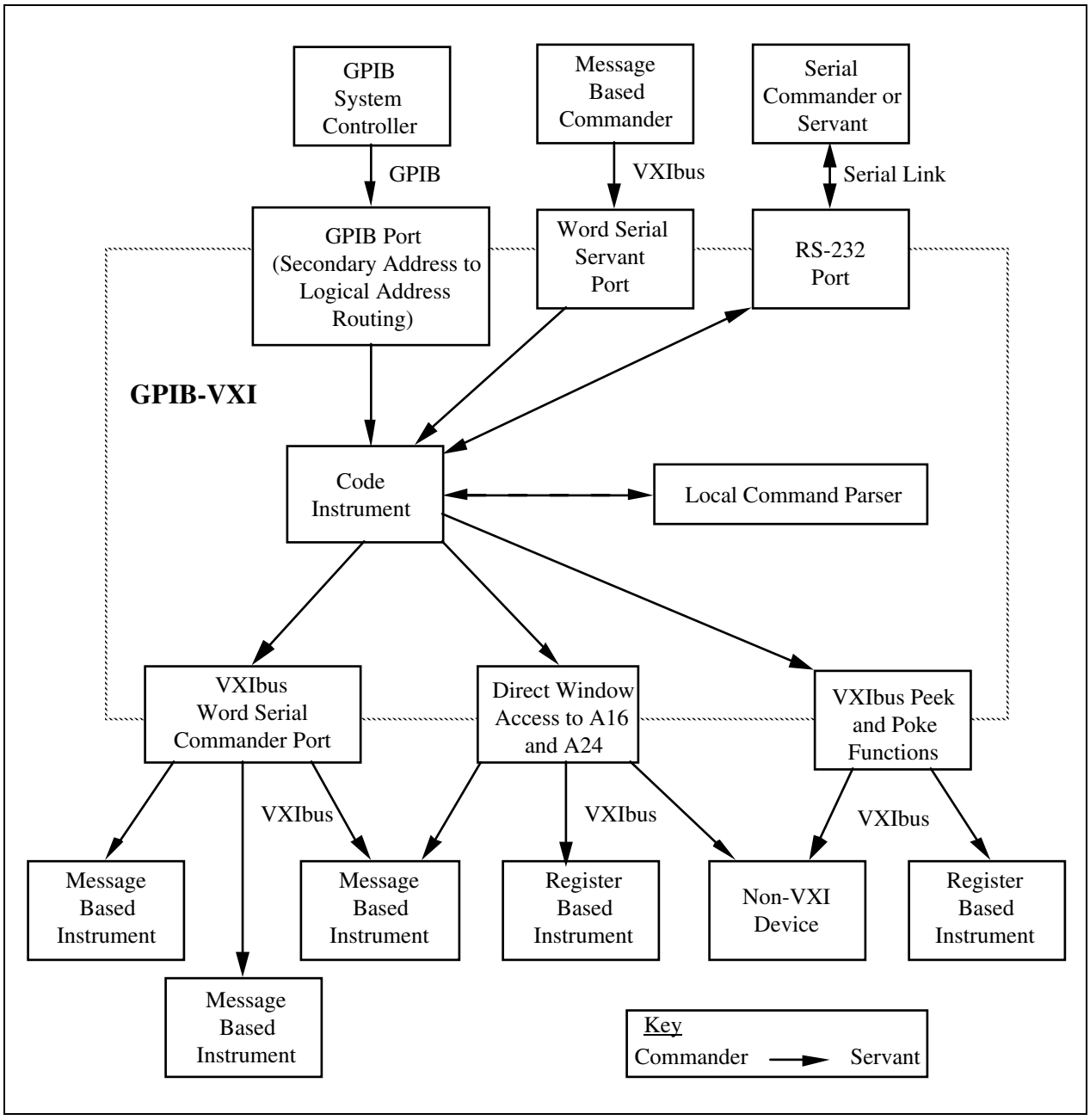


Figure C-2. Code Instrument Operation

CI Operation

A CI is a set of software routines that can perform the functions of a physical VXI Message-Based device. These CI capabilities are illustrated in Figure C-2. CIs coexist with the IEEE-488 VXI translation and local command set functions shown in Figure C-1, with the exception that the Word Serial servant and RS-232 ports support only one command/servant connection (either to the local command set parser or to a CI, but not both) at one time.

CI Characteristics

A CI has all of the abilities of a physical Message-Based commander. A CI can do the following:

- Set up its own set of configuration registers
- Have servants assigned to it
- Engage in Word Serial communication with its commander and servants
- Receive VXI interrupts
- Send and receive VXI signals
- Directly access the A16 or A24 registers/memory of a VXI or non-VXI device
- Communicate with the GPIB System Controller through a secondary address
- Perform memory-to-memory DMA operations using 68070 DMA channel
- Source or accept a trigger on any one TTL trigger line

As with physical devices, a CI must be an immediate servant of the GPIB-VXI in order to have a GPIB secondary address assigned to it. In addition to these VXIbus device capabilities, CIs can also communicate directly with the local command set parser and the serial port.

The GPIB-VXI emulates the physical capabilities of a Message-Based device for each CI. Because a CI can be a commander or a servant, multilevel hierarchies of CIs and physical Message-Based devices can be constructed. The only restriction is that a CI cannot be mapped out of the hierarchy of devices within the GPIB-VXI. In other words, a CI can be any of the following:

- The commander of any number of CIs and/or physical VXI devices
- A top-level commander
- A servant of another CI
- A servant of the GPIB-VXI's commander

A CI cannot, however, be the servant of a physical VXI device that is not the GPIB-VXI's commander.

A CI appears to the GPIB and to other CIs to be a physical device, since it performs all of the functions that a physical Message-Based device performs. If the CI takes control of the physical Word Serial registers on the GPIB-VXI, it becomes a physical VXI device. Most applications, however, do not require a CI to take control of the physical Word Serial registers, because most CIs will function as commanders to drive other servants in the system rather than as servants to higher level commanders.

A non-VXI device does not have VXI configuration registers, so it does not appear in the VXI device hierarchy. A CI that provides a Message-Based interface for a non-VXI device (together with that non-VXI device) is viewed by the system as a single device. Typical examples of non-VXI devices include:

- VME cards (CPU, Register-Based, memory, and so on)
- CDS 73A-852 adapter module

Downloaded CIs and EPROMed CIs

You can download CIs in the form of binary code into the GPIB-VXI's RAM. The downloaded modules are called *Downloaded CIs*, or *DCIs*. The CI Configuration local commands download and initialize CIs. You can use the DCI form to develop CIs without programming EPROMs, or to create disk-loadable CI applications.

The GPIB-VXI also has an interface for installing CIs in onboard EPROMs, including a mechanism for automatically initializing them at system startup. CIs stored in the EPROMs are called *EPROMed CIs*, or *ECIs*. You can use the ECI form to create stand-alone CI applications.

Resident CIs

A Resident CI (RCI) that communicates with a CDS 73A-852 adapter is supplied by National Instruments as part of the GPIB-VXI firmware. The 852 adapter is a non-VXI device that requires a special code module somewhere in the system with a Message-Based interface. Appendix C, *Using the CDS-852 Adapter Code Instrument*, contains information about installing and using the 852 adapter CI.

Summary

With these capabilities, a CI can emulate or replace any existing VXI or VME device, or extend a device's native capabilities to new levels of functionality, as a disk-loadable or stand-alone solution. To summarize, CIs improve the system structure for the following reasons:

- GPIB traffic is greatly reduced.
- Register-Based and non-VXI devices can be treated as if they were Message-Based.
  - The GPIB Controller sees one type of instrument (an IEEE-488 instrument).
  - Standard IEEE-488 communication is possible with all types of VXI/non-VXI instruments.
- GPIB control of a VXIbus system can be implemented uniformly at a high level.
- Application software is simplified due to uniformity of control.
- System performance is greatly increased.
  - Direct access results in a tight coupling with its servants.
  - Distributed processing removes burden from outside controller.
  - Access to VXIbus bandwidth is accomplished without GPIB overhead.

Appendix D
Using the CDS-852 Adapter Code Instrument

This appendix contains instructions for installing and using the National Instruments-supplied Code Instrument (CI) for controlling one or more Colorado Data Systems (CDS) 73A-852 adapter modules.

The 73A-852 is a non-VXI device with communication registers located in A24 space rather than in A16 space. To communicate with the 852 adapter as a Message-Based device, the 73A-852 requires special adapter software. The GPIB-VXI performs the Message-Based-to-852 communication translation with a CI.

The GPIB-VXI B.1 firmware includes one 852 Position Independent CI. This CI implements the configuration and translation functions required to communicate with up to twelve 852 adapter modules via the GPIB.

Installing the 852 Adapter CI

The 852 adapter CI is installed by configuring the nonvolatile configuration parameters.

Enter the nonvolatile configuration mode as described in Chapter 4. The following menu is displayed:

```
   GPIB-VXI Nonvolatile Configuration Main Menu
         (C) 1989  National Instruments
================================================
1). Read In Nonvolatile Configuration
2). Print Configuration Information
3). Change Configuration Information
4). Set Configuration to Factory Settings
5). Write Back (Save) Changes
6). Quit Configuration

    Choice (1-6):
```

Enter 3 to change the configuration information. The following menu is then displayed:

```
        GPIB-VXI Nonvolatile Configuration Changer
              (C) 1989  National Instruments
    ================================================
     1). Edit pSOS Configuration
     2). Edit Default VXI Interrupt Handler Levels
     3). Edit Resource Manager A24/A32 Assign Bases
     4). Edit Servant Area and DC Starting LA
     5). Edit FAILED Device handling mode
     6). Edit GPIB Configuration
     7). Edit default CI Configuration
     8). Edit Resident CI Base Locations
     9). Edit CI User Configuration Variables
     Q). Quit Editor

         Choice (1-9,Q):
```

Enter 1 to edit pSOS configuration. The following prompt then appears:

```
      ------pSOS Configuration------

Enter Dynamic RAM Region 1 Size (default 0x70000):

Enter <CR> to keep the present value and continue to the next entry:

Enter Maximum Number of Processes (default 0x20):
```

The following formula is used to calculate the maximum number of processes:

Number of processes = 10h + (# Secondary Address links) + (2* # CI's)

If fewer than six CIs are installed and no other secondary address links exist, the default value of 32 (0x20) is adequate. Increasing the number of processes affects the throughput of the GPIB-VXI. Enter the number of processes in hexadecimal.

The next prompt is then displayed:

```
Enter Maximum Number of Exchanges (default 0x20):
```

The following formula is used to calculate the maximum number of exchanges:

Number of exchanges = 10h + (# CI's)

The default value of 32 (0x20) is adequate even if all 12 CIs are installed. Enter <CR> to select the default value.

The last prompt appears:

```
Enter Maximum Number of Message Buffers (default 0x180):
```

The following formula is used to calculate the maximum number of message buffers:

Number of message buffers = 100h + (25* # CI's)

If fewer than six CIs are installed, the default value of 384 (180h) is adequate. Increasing the number of message buffers affects the throughput of the GPIB-VXI. Enter the number of message buffers in hexadecimal.

When the edit menu reappears, enter `8` to edit the resident CI base locations.

Configure as many CI base locations as there are 852 adapters to be controlled by the GPIB-VXI. For example, to control four 73A-852s, configure CI base locations 0 through 3. The address for the base of the CI should be F5E000h.

Type `Y` to respond yes to the `Debug mode On for Resident CI 0xXX` prompt. This enables debug statement printing to the terminal.

For example, to install two 852 adapter CIs and enable debug statement printing on the second CI, enter the following sequence, which we have highlighted in boldface type for this example:

```
     ----- Resident CI Base Location Configuration -----
Enter Number of Base Location to EDIT (0xff = EXIT): 0
Enter Address for Base 0x00 (default = 0x000000): F5E000
Debug mode ON for Resident CI 0x00 (default NO): <CR>
Enter Number of Base Location to EDIT (0xff = EXIT): 1
Enter Address for Base 0x01 (default = 0x000000): F5E000
Debug mode ON for Resident CI 0x01 (default NO): Y
Enter Number of Base Location to EDIT (0xff = EXIT): <CR>
```

When the edit menu reappears, enter `Q` to exit the configuration editor. When the Nonvolatile Configuration main menu appears, enter `5` to save the configuration changes. When the Nonvolatile Configuration main menu reappears, enter `2` to confirm the configuration information. The CI configuration for the previous example would be displayed as follows:

```
          ------ Resident Code Instrument Locations ------
0x00: 00F5E000            0x01: 00F5E000            0x02: 00000000
0x03: 00000000            0x04: 00000000            0x05: 00000000
0x06: 00000000            0x07: 00000000            0x08: 00000000
0x09: 00000000            0x0A: 00000000            0x0B: 00000000
```

When the main menu reappears, enter `6` to quit the configuration mode. The following message appears:

```
Must Re-Initialize pROBE or re-boot for pSOS changes to take effect.
Other changes made automatically when configuration saved.

*********************************************
*           DONE WITH CONFIGURATION          *
* Change Startup mode Dip Settings to enter  *
* different mode or push RESET to re-configure.*
*********************************************
```

Deleting a CI

To delete a CI, follow the installation procedure but set the CI's base address location to 000000.

Logical Address and A24 Address Assignment

The 852 adapter CIs are assigned logical addresses sequentially, starting with the lowest configured CI base address and logical address 80. For example, if the CIs at base address locations 1 and 3 are installed, the CI at location 1 is assigned logical address 80, and the CI at location 3 is assigned 81.

The default offset where the CI expects to find its 73A-852 registers in VME A24 space is related to the CI's logical address as follows:

73A-852 A24 offset = CI logical address * 10000h

For example, a CI at logical address 80h expects (by default) to find its 73A-852 registers at offset 800000h. The CI's A24 offset can be changed with the CI command `!!L.` The 73A-852 has rotary switches for changing its A24 register locations.

852 Adapter CI Commands

The 852 adapter CI commands are interpreted by the CI itself and do not directly affect the 852 module. If the adapter CI receives a word serial buffer that does not begin with the `!!` character sequence, it assumes that the buffer is for the 73A-852 and writes the buffer to the appropriate A24 register location. The CI command formats were chosen to minimize the possibility of conflict with the command sets of the various plug-in instruments that are compatible with the 852 adapter. Since National Instruments has not had the opportunity to study the command set of all CDS plug-in instruments, the possibility of conflict should be kept in mind while developing applications.

The `!!A` and `!!B` commands set the CI read mode for compatibility with the CDS instrument. Some CDS command responses are in ASCII format, while others are in binary format. Refer to the appropriate CDS manual for the response format of a particular command. The `!!A` command sets the adapter CI mode to be compatible with the ASCII response format. If the expected response format is binary, the `!!B` command must be used to set the CI to the binary read mode.

Two termination conditions can apply to reading data from the 73A-852. Depending upon the 73A-852 configuration and the operation being performed, the 852 may terminate the transmission of data with an end-of-string (EOS) character, or it may assert the END bit (VMEbus data bit 8).

The `!!E`, `!!T`, and `!!t` commands configure the CI termination conditions. The EOS and END bit termination conditions are independent; that is, the CI can be configured to terminate a read from the 73A-852 on one condition, both conditions, or neither condition. The maximum size of binary mode reads can also be limited with the `!!S` command. Notice that with binary responses, there can be no unique EOS character, so the END or transfer size conditions (not EOS) should be used to terminate binary transfers. The default read mode is ASCII. The default read termination condition is the EOS character <LF> (0Ah) with the END bit set.

The `!!Q` command returns information about the CI settings (always to the serial port). The `!!D` and `!!d` commands control the printing of run-time debug information to the terminal connected to the serial port.

!!A

Purpose: Set the adapter CI read mode to ASCII, for compatibility with the CDS instrument response.

Command Syntax: `!!A`

or

`!!a`

Action: Sets the adapter CI read mode to ASCII. The maximum ASCII response size allowed is 512 bytes.

!!B

Purpose: Set the adapter CI read mode to binary.

Command Syntax: `!!B`

or

`!!b`

Action: Sets the adapter CI read mode to binary. Read size is limited to 512 bytes, or as configured by the `!!S` command.

!!D

Purpose: Enable debug message printing to the serial port.

Command Syntax: `!!D`

Action: Enables debug message printing to the serial port.

!!d

Purpose: Disable debug message printing to the serial port.

Command Syntax: `!!d`

Action: Disables debug message printing to the serial port.

!!E

Purpose: Configure CI read termination on an EOS character.

Command Syntax: `!!E <hex number>`

or

`!!e <hex number>`

`<hex number>` is the ASCII value corresponding to the desired EOS character (for example, 0Ah for <LF>).

Action: Enables read termination on an EOS with a value of `<hex number>`. If `<hex number>` is greater than FFh, the EOS termination condition is disabled.

Examples: Set the EOS character to <CR>

```
!!E 0D
```

Disable EOS read termination.

```
!!E 100
```

!!L

Purpose: Set the A24 base address where the adapter CI expects to find the 852 adapter.

Command Syntax: `!!L <val>`

or

`!!l <val>`

`<val>` is a hex value equal to the upper 8 bits of the target adapter's A24 address.

Action: The adapter CI expects to find the target 852 adapter at offset `<val>` * 10000h. The default (initial) value of `<val>` is the adapter DCI's logical address.

Example: Set the adapter CI to operate with an 852 adapter at A24 base address 830000h.

```
!!L 83
```

!!S

Purpose: Set the maximum size of a binary read.

Command Syntax: `!!S <size>`

or

`!!s <size>`

`<size>` is a decimal value.

Action: Sets the maximum binary read size to `<size>` bytes. The default value of the read size is 512 bytes.

!!T

Purpose: Enable read termination when the END bit is set.

Command Syntax: `!!T`

Action: Enables read termination when the END bit (bit 8) is set.

!!t

Purpose: Disable read termination on the END bit.

Command Syntax: `!!t`

Action: Disables read termination on the END bit (bit 8).

Appendix E
GPIB-VXI Hardware and Software Configuration Form

In the event that you have a technical problem, complete the following form and then call National Instruments for technical support. To complete the form, record the settings and revisions of your hardware and software on the line located to the right of each item. If you complete this form accurately, our applications engineers will be able to answer your questions efficiently.

National Instruments Products

- GPIB-VXI Model Number (such as 180715-022): _____
- Firmware Revision Number (see EPROM labels inside module): _____

GPIB Controller

- Manufacturer: _____
- Model: _____
- GPIB Interface Module Installed (if applicable): _____
- Programming Language: _____
- Other Modules in System: _____

Other VXIbus Products

Slot	Manufacturer	Model No.	Function (such as A/D or DMM)	Logical Address
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Glossary

Backplane	An assembly, typically a printed circuit board, with 96 pin connectors and signal paths that bus the connector pins. VXIbus systems have either two sets of bused connectors, designated J1 and J2 backplanes, or three sets of bused connectors, designated J1, J2, and J3 backplanes.
CI	See *Code Instrument.*
Code Instrument	CI; a proprietary National Instruments software structure that uses software to emulate the capabilibites of a VXI Message-Based device.
Command	Causes the GPIB-VXI to take some action.
Commander	A Message-Based device that is also a bus master and can control one or more servants.
Console response	Returned in the form of readable sentences, which is better suited for interactive command entry.
DC device	See *Dynamic configuration device*.
DCI	See *Downloaded CI.*
Diagnostics mode	Mode in which you can perform extensive offline diagnostic tests of the GPIB-VXI.
Downloaded CI	DCI; a form of CI that is downloaded into the GPIB-VXI's RAM memory.
Dynamic configuration device	DC device; a device that initially has a logical address of 255. The RM subsequently assigns it a different, unique logical address.
ECI	See *EPROMed CI.*
EEPROM	Electrically Eraseable Programmable Read Only Memory.
EPROM	Eraseable Programmable Read Only Memory.
EPROMed CI	ECI; a form of CI that is user-installed into EPROMs.

GPIB	General Purpose Interface Bus. The industry standard IEEE-488 bus.
GPIB-VXI local command set	Consists of commands and queries.
Logical address	An 8-bit number that uniquely identifies each VXIbus device in a system. It defines a device's A16 register address and indicates commander/servant relationships.
Message-Based device	An intelligent device that implements the defined VXIbus registers and communication protocols.
MODID lines	VXI backplane signals used by the Resource Manager (through the use of the Slot 0 device) in order to perform slot associations for logical addresses. There are 13 MODID lines, one for each slot in a full-size mainframe.
Module	Typically consists of a board assembly and its associated mechanical parts, front panel, optional shields, and so on. A module contains everything required to occupy a slot in a mainframe. A module can occupy one or more slots.
Nonvolatile configuration mode	Mode in which you can edit the contents of the nonvolatile EEPROM memory.
NV	Nonvolatile memory.
Peek	To read the contents.
PH	Programmable Handlers.
Poke	To write a value.
pROBE	A low-level interactive debugger for use with the pSOS operating system. It is commercially available from Software Components Group, Inc. VXI pROBE is an enhanced version of pROBE supplied on developmental versions of the GPIB-VXI.
Program mode response	Has a terse data-only format that is intended for a control program to read and parse.
pSOS	A small, multitasking operating system kernel used on the GPIB-VXI. It is commercially available from Software Components Group, Inc.
Query	Similar to a command in that it also causes the GPIB-VXI to take some action, but it always returns a response containing data or other information.

RCI	See *Resident CI*.
Register-Based Device	A servant-only device that supports VXIbus configuration registers. Register-Based devices are typically controlled by Message-Based devices via device-dependent register reads and writes.
Resident CI	RCI; a CI that is supplied by National Instrument and resides in the firmware.
RM	See *Resource Manager*.
Resource Manager	A Message-Based commander located at logical address 0 that provides configuration management services such as address map configuration, commander/servant mappings, self-test and diagnostic management.
SC Device	See *Static configuration device*.
Servant	A device that is controlled by a commander. Any device can be a servant.
Static configuration device	SC device; a device that has its logical address set by static means, such as by a DIP switch.
System configuration table	During the execution of the RM and general configuration operations, the GPIB-VXI builds up a table of system configuration information. Each device has an entry in the table containing the
VME	Versa Module Eurocard or IEEE 1014.
VXIbus	VMEbus Extensions for Instrumentation.
VXI pROBE mode	Mode in which you can use the enhanced pROBE debugger. This mode is available only with the GPIB-VXI development firmware option.
VXI system mode	The startup mode for normal operation in a VXI system.
Word Serial communication	The simplest form of communication required by Message-Based devices. It utilizes the A16 communication registers to transfer data using a simple polling handshake method.
Word Serial Protocol	The rules and regulations involved in performing Word Serial communication.

Index

Numbers

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

User Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: **GPIB-VXI User Manual**

Edition Date **April 1990**

Part Number: **320151-01**

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Phone ( )_____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway, MS 53-02
Austin, TX 78730-5039